Zur Benutzbarkeit assoziativer Verknüpfungen in verteilten Informationssystemen:

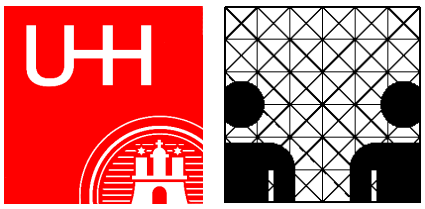
Entwicklung und Evaluation von Konzepten
zur Optimierung der Navigation in
offenen, verteilten Hypertext-Informationssystemen
gezeigt am World Wide Web

Dissertation
zum Erlangen des akademischen Grades
Doktor der Naturwissenschaften

am
Fachbereich Informatik der
Fakultät für Mathematik, Informatik und Naturwissenschaften der
Universität Hamburg

vorgelegt von
Harald W. R. Weinreich

Hamburg 2012



Dissertation zur Erlangung des Grades des Doktors der Naturwissenschaften
genehmigt von der Fakultät für Mathematik, Informatik und Naturwissenschaften,
Fachbereich Informatik an der Universität Hamburg

Gutachter:

Prof. Dr. Winfried Lamersdorf, Universität Hamburg (Betreuer und 1. Gutachter)
Prof. Dr. Horst Oberquelle, Universität Hamburg (2. Gutachter)
Prof. Dr. Susanne Maaß, Universität Bremen (3. Gutachter)

Tag der Disputation: 12. April 2012 in Hamburg

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://portal.dnb.de/ abrufbar.

1. Auflage August 2012

ISBN: 978-1-291-01889-9

© Harald Weinreich, Hamburg, 2012. Alle Rechte vorbehalten.

Druck und Vertrieb: Lulu Enterprises Inc., Morrisville, NC, USA. http://www.lulu.com/de

Printed in Germany

Zusammenfassung

Große verteilte Informationssysteme wie das World Wide Web sind heute für die Kommunikation und den Informationsaustausch sowohl im privaten als auch im professionellen Bereich unverzichtbar. Sie werden von Personen mit unterschiedlichsten kulturellen, sozialen und gesellschaftlichen Hintergründen verwendet, eine gute Benutzbarkeit ist daher entscheidend für ihren Erfolg. Die Digitalisierung trägt wesentlich zum vereinfachten und beschleunigten Zugriff auf die Informationen bei, sofern eine sichere Orientierung und Navigation im System gegeben ist. Diese Arbeit befasst sich mit der Ergonomie verteilter Informationssysteme, identifiziert die Probleme bei ihrer Nutzung und stellt neue Lösungsmöglichkeiten für die Reduzierung der Benutzbarkeitsdefizite sowohl auf Ebene der Benutzungsschnittstelle als auch auf technischer Ebene vor.

Ein zentrales Element für die Navigation in verteilten Informationssystemen sind die assoziativen Verweise zwischen den Objekten, die als *Hyperlinks* bezeichnet werden. Solche im Dokument eingebetteten Verknüpfungen stellen Beziehungen zwischen beliebigen Ressourcen her und können Aktionen auslösen. Die Idee, inhaltlich verwandte Dokumente in einem Informationssystem zur direkten Navigation miteinander zu vernetzen, um die Handhabbarkeit der stetig wachsenden Informationsbestände zu verbessern, stammt bereits aus den 1930er Jahren und wurde in den 1970er Jahren unter dem Begriff *Hypertext* bekannt. Das Web ist das bisher erfolgreichste verteilte Informationssystem, das dieses Konzept einsetzt. Hyperlinks sind das wichtigste Navigationselement im Web, ohne sie wäre es in der heutigen Form undenkbar.

Der Einsatz von Hyperlinks und die Datenverteilung auf viele Systeme brachten jedoch auch neue Benutzbarkeitsprobleme mit sich, bei denen die Benutzungsschnittstelle von Links eine zentrale Rolle spielt. Wesentliche Konzepte zur Reduktion der Schwierigkeiten bei der Link-Navigation sind *typisierte Links* und *Link-Previews*, die Anwendern bereits vor der Auswahl Informationen zum Link und zum Zielobjekt vermitteln. Am Beispiel des Webs wird in dieser Arbeit gezeigt, dass in verteilten Informationssystemen zahlreiche Details zur Art, Funktion und zum Ziel eines Links existieren, die für Nutzer unzugänglich bleiben. Die im Web vorhandenen Link-Informationen werden klassifiziert, und es wird mithilfe von Benutzbarkeitstests und einer Online-Umfrage untersucht, welche von ihnen Anwender als wichtig für die Navigation ansehen.

Neben der Relevanz von Link-Informationen befasst sich diese Arbeit mit den Herausforderungen bei der Darstellung solcher Informationen. Basierend auf den Forschungsergebnissen früherer Hypertext-Projekte und -Standards sowie aktuellen Untersuchungen zur Benutzbarkeit des Webs werden erstmals die Gestaltungsmöglichkeiten der *Benutzungsschnittstelle von Hyperlinks* klassifiziert und systematisch neue Konzepte für eine erweiterte Link-Schnittstelle entworfen und realisiert. Diese Konzepte helfen bei der Identifikation von Link-Ankern und erlauben die Anzeige wichtiger zusätzlicher Link-Informationen in Webseiten.

Zur Analyse der Stärken und Schwächen werden die neu entwickelten Schnittstellenkonzepte in Form der zwei Prototypen *HyperScout I* und *HyperScout II* vorgestellt und jeweils mit Benutzerstudien evaluiert. Die Tests liefern neue Erkenntnisse über die Anforderungen bei der Gestaltung benutzbarer Schnittstellen für Hyperlinks in verteilten Informationssystemen.

Eine Erweiterung der Benutzungsschnittstelle von Hyperlinks erfordert aufgrund der offenen und verteilten Architektur des Webs neue Techniken zur Übertragung der zusätzlich benötigten Daten. Sie sollen gewährleisten, dass Anwendern ergänzende Link-Informationen ohne Verzögerung dargestellt werden. Dies wird durch ein Konzept erreicht, das auf einer Erweiterung des HTTP-Protokolls und einem neuen Modul für die Sprache XHTML beruht. Bei der Konzeption wurde auf Kompatibilität mit aktuellen Web-Standards, eine gute Skalierbarkeit und eine Minimierung der zusätzlich benötigten Bandbreite geachtet.

Sowohl die Realisierung neuer Konzepte zur Erweiterung der Benutzungsschnittstelle als auch Anpassungen der Protokolle und Sprachen des Webs bringen eine erhebliche technische Komplexität mit sich. Hierzu trägt bei, dass Server und Browser erweitert werden müssen und dass, bedingt durch die schnelle technische Evolution, viele untereinander inkompatible Schnittstellen bedient werden müssen. Zur Vereinfachung der prototypischen Realisierung solcher Konzepte wurde ein Framework namens *Scone* konzipiert und entwickelt, das sich durch Plattformunabhängigkeit und Flexibilität auszeichnet. Es weist eine komponentenbasierte Architektur auf und bietet eine Plug-in-Schnittstelle zur Integration und Konfiguration neuer Prototypen. Das Framework beruht auf einem Web-Intermediary zur Analyse und Modifikation übertragener Daten und einem programmierbaren Web-Crawler, der regelbasiert zusätzliche relevante Informationen für Anwender zusammenstellen kann. Komponenten zur Verarbeitung und Persistierung von Web-Ressourcen sowie zur Anbindung an Browser und Server vereinfachen die Umsetzung vieler unterschiedlicher Konzepte zur Erweiterung des Webs. Das Scone-Framework war Grundlage für die prototypische Realisierung und Evaluation der in dieser Arbeit entwickelten Konzepte und wurde bereits in zahlreichen weiteren internationalen Forschungsprojekten eingesetzt.

Danksagungen

Die vorliegende Arbeit hat sehr von der Kooperation mit meinen Kollegen an der Universität Hamburg profitiert. Hervorheben möchte ich die exzellente Zusammenarbeit mit Hartmut Obendorf und Matthias Mayer. Mein Dank gilt in erster Linie meinem Doktorvater, Herrn Prof. Dr. Winfried Lamersdorf, der mir die Freiheit gab, ein Thema meiner Wahl zu bearbeiten und mich und diese Arbeit mit wertvollem Rat, viel Umsicht und Geduld unterstützt hat. Ebenfalls bin ich Herrn Prof. Horst Oberquelle für seine Motivierung und Anregungen zu vielen meiner Projekte und Veröffentlichungen sehr verpflichtet.

Besonderen Dank schulde ich meiner Lebensgefährtin Verena Franzen, meiner Familie und meinem Onkel Reinhardt Weinreich, ohne deren Beistand und Unterstützung diese Arbeit in der ausführlichen Form nicht möglich gewesen wäre. Vielen meiner Freunde und Bekannten bin ich für ihre Mitwirkung bei Umfragen und Studien im Rahmen dieser Arbeit sehr verbunden. Abschließend möchte ich allen Beteiligten meinen herzlichen Dank für ihre Geduld und ihr Vertrauen ausdrücken.

Inhaltsverzeichnis

1 Einleitung und Motivation 1

1.1 Forschungsproblem und Zielsetzung 1

1.2 Forschungsansatz und Vorgehen 3

1.3 Aufbau der Arbeit 5

2 Theoretische Grundlagen verteilter Hypertext-Informationssysteme 9

2.1 Information, Assoziation, Hypertext 9

2.1.1 Der Informationsbegriff 9

2.1.2 Assoziatives Denken 11

2.1.3 Informationssysteme 12

2.1.4 Hypertext 14

2.1.5 Anwendungsfelder von Hypertext 17

2.2 Hypertext-Informationssysteme: Paradigmen und Begriffe 19

2.2.1 Das assoziative Hypertext-Paradigma: Knoten und Links 21

2.2.2 Die Konzepte offener, verteilter Hypertext-Informationssysteme 25

2.2.2.1 Verteilter Hypertext 26

2.2.2.2 Offener Hypertext 27

2.2.3 Vom offenen, verteilten Hypertext zum World Wide Web 30

2.2.4 Spezifische Potenziale und Herausforderungen des Webs 31

2.2.5 Resümee der Neuerungen des Webs 37

3 Zur Benutzbarkeit von verteilten Informationssystemen und Hypertext 39

3.1 Benutzbarkeit von Navigationswerkzeugen in verteilten Informationssystemen 40

3.1.1 Die Suchfunktion 41

3.1.2 Hierarchische Navigation 43

3.1.3 Der Index 45

3.1.4 Übersichtskarten 46

3.1.5 Die „Guided Tour“ 46

3.1.6 Filtermechanismen, Empfehlungssysteme und soziale Navigation 47

3.1.7 Navigationswerkzeuge zum Wiederfinden von Objekten 48

3.1.8 Die Bedeutung von Hyperlinks in verteilten Informationssystemen 50

3.2 Grundlegende Benutzbarkeitsprobleme von Hypertext 51

3.2.1 Orientierung und Navigation im Hyperspace 51

3.2.2 Cognitive Overhead ... 52
3.3 Besondere Herausforderungen verteilter Hypertext-Informationssysteme ... 54
3.3.1 Performanz ... 54
3.3.2 Information Overload ... 61
3.3.3 Qualität und Vertrauen ... 63
3.3.4 Konsistenz ... 68
3.4 Die Rolle von Links für die Benutzbarkeit verteilter Hypertext-Systeme ... 71

4 Typisierung als Grundlage benutzbarer Hyperlinks ... 75
4.1 Konzepte zur Verbesserung der Benutzbarkeit von Links ... 75
4.1.1 Link-Previews ... 76
4.1.2 Typisierte Links ... 77
4.2 Grundlagen und Eigenschaften typisierter Links ... 79
4.2.1 Konzepte zur expliziten Typisierung von Links ... 80
4.2.2 Frei typisierte Links ... 80
4.2.3 Streng typisierte Links ... 82
4.2.4 Semi-streng typisierte Links mit adaptivem Typschema ... 85
4.3 Grenzen der Typisierung von Hyperlinks ... 85
4.4 Link-Techniken und typisierte Links im World Wide Web ... 88
4.4.1 Technische Grundlagen assoziativer Links im Web ... 88
4.4.2 Methoden zur Typisierung assoziativer Links in HTML ... 89
4.4.3 Informationen zum Zielobjekt des Links ... 91
4.4.4 Link-spezifische Darstellung von Link-Markern mit CSS ... 92
4.4.5 Definition des Verhaltens von Links im Web ... 94
4.4.6 Programmatische Links im Web ... 94
4.4.7 Strukturelle Links im Web ... 95
4.4.8 „Transklusionen“: Verknüpfungen zum Einbinden von Objekten ... 97
4.4.9 Zusammenfassung der Defizite der Link-Typisierung im Web ... 97
4.5 Eine Klassifikation impliziter Link-Informationen im Web ... 98
4.5.1 Kategorien impliziter Link-Typen im Web ... 99
4.5.1.1 Der semantische Link-Typ ... 100
4.5.1.2 Die Verteilungscharakteristik des Links ... 103
4.5.1.3 Die Link-Aktion ... 104
4.5.2 Kategorien inhaltlicher Informationen zu Ressourcen im Web ... 106
4.5.2.1 Der Titel des Dokuments ... 106

4.5.2.2 Autor und Anbieter ... 107
4.5.2.3 Datum der Erstellung und Aktualisierung ... 107
4.5.2.4 Zusammenfassung des Inhalts ... 108
4.5.2.5 Sprache des Dokuments ... 109
4.5.2.6 Weitere standardisierte Meta-Informationen in Webseiten ... 110
4.5.2.7 Eignung inhaltlicher Informationen für Link-Previews im Web ... 111
4.5.3 Typisierung der Ressourcen im Web ... 111
4.5.3.1 Der technische Typ: das Dateiformat des referenzierten Objekts ... 111
4.5.3.2 Der Medientyp ... 113
4.5.3.3 Der topologische Typ des Dokuments ... 114
4.5.4 Informationen zum Zugriff auf Ressourcen im Web ... 116
4.5.4.1 Die Verfügbarkeit der Ressource ... 116
4.5.4.2 Die Latenzzeit bei der Navigation ... 117
4.5.5 Benutzungsinformationen im Web ... 119
4.5.5.1 Personal History: die persönliche Benutzung des Informationssystems ... 119
4.5.5.2 Social Navigation: Aktionen und Bewertungen anderer Personen ... 120
4.5.6 Bedeutung des URI als Element der Benutzungsschnittstelle ... 122
4.6 Zusammenfassung ... 123

5 Konzeption einer erweiterten Benutzungsschnittstelle für Hyperlinks ... 125
5.1 Die Benutzungsschnittstelle von Hyperlinks ... 125
5.1.1 Ziele bei der Gestaltung von Link-Markern für eingebettete, assoziative Links ... 126
5.1.2 Anforderungen an die Link-Schnittstelle in offenen, verteilten Hypertext-Informationssystemen ... 127
5.1.3 Der gegenwärtige „Standard“ für Link-Marker ... 129
5.1.4 ISO 9241-151: Norm zur Gestaltung von Schnittstellen im Web ... 131
5.2 Link Presence: Kategorisierung der Methoden zur Kennzeichnung von Link-Ankern 132
5.2.1 Typografische Kennzeichnung ... 132
5.2.2 Farbliche Hervorhebungsmethoden ... 133
5.2.3 Link-Symbole und Icons ... 134
5.2.4 Hervorhebung durch Umrahmung ... 135
5.2.5 Overlays ... 136
5.2.6 Laterale Link-Marker ... 137
5.2.7 Unsichtbare Link-Anker ... 138

5.2.8 Links-on-Demand 139
5.2.9 Link-Marker für Grafiken 140
5.2.10 Multimediale Link-Marker 141
5.2.11 Zusammenfassung 142
5.3 Link-Destination: Darstellungstechniken für erweiterte Informationen zum Link-Ziel 142
5.3.1 Ergänzender Link-Text 143
5.3.2 Link-Hinweise durch Symbole und Icons 145
5.3.3 Farbig codierte Link-Marker 147
5.3.4 Link-Hinweise in Übersichtskarten 148
5.3.5 Der reservierte Anzeigebereich 150
5.3.6 Der adaptive Mauszeiger 151
5.3.7 Tooltips 152
5.3.8 Weitere Techniken 154
5.3.9 Zusammenfassung 155
5.4 Verwandte Forschungsprojekte zur Erweiterung der Link-Schnittstelle im Web 157
5.5 Konzeption einer erweiterten Hyperlink-Schnittstelle: HyperScout I 162
5.5.1 Anforderungen für Link-Previews im Web 163
5.5.2 Gestaltung strukturierter Tooltips für Link-Previews 164
5.5.3 Architektur und Funktionsweise von HyperScout I 166
5.5.4 Klassifikation der in den Tooltips dargestellten Inhalte 169
5.5.4.1 Der semantische Link-Typ 169
5.5.4.2 Die Verteilungscharakteristik des Links 170
5.5.4.3 Die Link-Aktion 172
5.5.4.4 Informationen zum Inhalt des Link-Ziels 174
5.5.4.5 Der technische Typ des Zielobjektes 175
5.5.4.6 Der Medientyp des Link-Ziels 177
5.5.4.7 Der topologische Typ des Zielobjektes 177
5.5.4.8 Die Verfügbarkeit des Ziel-Objektes 178
5.5.4.9 Die Latenzzeit bei der Navigation 179
5.5.4.10 Informationen zur Benutzung 180
5.5.4.11 Der URI des Link-Ziels 180

6 Evaluation und Optimierung der erweiterten Hyperlink-Benutzungsschnittstelle .. 181
6.1 Empirische Evaluation von HyperScout I 181

6.1.1 Studienziele und Studienhypothese....182
6.1.2 Methoden zur Evaluation der Benutzbarkeit von Software....183
6.1.3 Thinking Aloud Studie: Methodik und Teilnehmerzahl....185
6.1.4 Aufbau und Ablauf der Evaluation von HyperScout I....188
6.1.5 Ergebnisse der Pilottests: besondere Herausforderungen bei der Evaluation....190
6.2 Ergebnisse der Evaluation von HyperScout I....191
6.2.1 Probleme bei der Interaktion mit Tooltips und Links....191
6.2.2 Herausforderungen bei der Informations-Präsentation in Tooltips....193
6.2.3 Bedeutung der angebotenen Informationen für die Navigation im Web....195
6.2.3.1 Der semantische Typ des Links....195
6.2.3.2 Die Verteilungscharakteristik des Links....196
6.2.3.3 Die Link-Aktion....197
6.2.3.4 Informationen zum Inhalt des Zielobjektes....198
6.2.3.5 Der technische Typ des Zielobjekts....199
6.2.3.6 Der Medientyp des Link-Ziels....200
6.2.3.7 Der topologische Typ des Link-Ziels....200
6.2.3.8 Die Zugreifbarkeit des Ziel-Objekts....200
6.2.3.9 Die Latenzzeit bei der Navigation....201
6.2.3.10 Informationen zur Benutzung....201
6.2.3.11 Der URI des Link-Zieles....202
6.2.4 Zusammenfassende Beurteilung der Bedeutung impliziter Link-Informationen für die Navigation....202
6.2.5 Innovative Vorschläge der Teilnehmer zur Vereinfachung der Link-Navigation....203
6.2.6 Fazit der Evaluation von HyperScout I....204
6.3 Eine Umfrage zur Problematik der Link-Typisierung und -Navigation im Web....205
6.3.1 Studienziele und Vorgehen....205
6.3.2 Ergebnisse der Umfrage....206
6.3.3 Probleme der Teilnehmer bei der Navigation mit Hyperlinks....207
6.3.4 Zur Bedeutung von Tooltips und typisierten Links im Web....209
6.3.5 Zur Bedeutung einzelner Eigenschaften von Links und Zielobjekten....210
6.4 Optimierung der erweiterten Link-Schnittstelle: HyperScout II....212
6.4.1 Farbig codierte Link-Marker für das Web: Link-Overlays-on-Demand....213
6.4.2 Multiple Maus-Icons für das Web....216

6.4.2.1 Erfahrungen mit partizipativen Verfahren zur Entwicklung von Link-Icons 217
6.4.3 Optimierung der Link-Tooltips für HyperScout II 221
6.5 Empirische Evaluation von HyperScout II 224
6.5.1 Ergebnisse der Evaluation zu den Link-Overlays-on-Demand 226
6.5.1.1 Die Bedeutung von Link-Overlays für die Identifikation von Link-Ankern 226
6.5.1.2 Die Eignung von Farben zur Visualisierung von Link-Attributen 227
6.5.2 Ergebnisse zu den multiplen Maus-Icons 231
6.5.2.1 Zur Gestaltung von Icons für die Darstellung von Link-Attributen 231
6.5.3 Ergebnisse zu den optimierten Tooltips 232
6.5.3.1 Erkenntnisse zur Interaktion mit Link-Tooltips im Web 233
6.5.3.2 Zur Präsentation von Link-Informationen in Tooltips 233
6.5.3.3 Bedeutung der Informationen in den Tooltips für die Navigation 234
6.5.3.4 Vorschläge der Teilnehmer zur Erweiterung der Link-Tooltips 239
6.5.4 Anforderungen an die Konfigurierbarkeit von Link-Previews 239
6.5.5 Die drei Visualisierungstechniken im Vergleich 241
6.5.6 Fazit der Evaluation der Konzepte zur Erweiterung der Link-Benutzungsschnittstelle 242

7 Konzeption von Protokollen und Diensten zur Erweiterung der Link-Schnittstelle 245
7.1 Technische Anforderungen für die Erweiterung der Link-Schnittstelle im Web 245
7.2 Grundlegende Konzepte zur Bereitstellung zusätzlicher Link-Informationen im Web 247
7.2.1 Konzepte zur clientseitigen Bereitstellung zusätzlicher Link-Daten 247
7.2.2 Ein Dienst zur Bereitstellung erweiterter Link-Daten 250
7.2.3 Konzepte zur serverseitigen Bereitstellung zusätzlicher Link-Daten 253
7.2.4 Diskussion der drei Konzepte 255
7.3 Konzepte zur effizienten Bereitstellung erweiterter Link-Informationen im Web 257
7.3.1 Funktionsweise des clientseitigen Agenten für Link-Daten 258
7.3.2 Funktionsweise des serverseitigen Dienstes für Link-Metadaten 261
7.4 Protokolle und Sprachen zur Übertragung erweiterter Link-Daten 264
7.4.1 RDF und REST 265
7.4.2 XLink 267
7.4.3 XHTML und HTTP 267

7.5 Die Protokoll- und Spracherweiterungen für das HyperScout-Konzept....268
7.5.1 Definition der neuer Attribute für Link-Metadaten....268
7.5.2 Deklaration des HyperScout-Moduls in XHTML 1.1....272
7.5.3 Definition einer Extension für das HTTP-Protokoll....282
7.6 Diskussion des vorgestellten technischen Konzepts....287

8 Ein generischer Ansatz zur Realisierung von Web-Erweiterungen....289
8.1 Voraussetzungen für die Realisierung von Navigationserweiterungen des Webs....290
8.2 Anforderungen an ein Framework zur Erweiterung der Navigationsmöglichkeiten im Web....291
8.2.1 Filtern und Ändern von Ressourcen....292
8.2.2 Autonomer Zugriff auf Ressourcen im Internet....294
8.2.3 Interpretation und Analyse von Web-Ressourcen....294
8.2.4 Ermittlung von Status und Ereignissen im Browser....294
8.2.5 Steuerung des Browsers....295
8.2.6 Mehrbenutzerunterstützung....295
8.2.7 Aufzeichnen von Benutzeraktionen....295
8.2.8 Repräsentation und Persistierung von Ressourcen....296
8.2.9 Anpassung der Benutzungsschnittstelle des Browsers und der dargestellten Objekte....296
8.2.10 Unterstützung der Systemevaluation....297
8.3 Techniken zur Erweiterung des Webs....297
8.3.1 Technische Möglichkeiten zur Erweiterung von Webbrowsern....297
8.3.2 Technische Möglichkeiten zur Erweiterung von Webservern....301
8.3.3 Diskussion der verfügbaren Erweiterungsmöglichkeiten....302
8.4 Intermediaries zur Erweiterung der Funktionalität des Webs....303
8.4.1 Charakteristika von Intermediaries....303
8.4.2 Einsatzbereiche von Intermediaries: Eine Klassifikation....304
8.4.2.1 Zwischenspeichern von Daten....305
8.4.2.2 Filtern von Daten....305
8.4.2.3 Aggregieren von Informationen....306
8.4.2.4 Transkodieren von Daten und Protokollen....306
8.4.2.5 Personalisieren von Inhalten....307
8.4.2.6 Privatisieren und Anonymisieren....307
8.4.2.7 Evaluation der Benutzung und Benutzbarkeit von Online-Systemen....307

8.4.3 Programmierbare Intermediaries: Ein Überblick ... 308
8.4.3.1 V6 Web Engine ... 309
8.4.3.2 Muffin ... 310
8.4.3.3 eRACE ... 310
8.4.3.4 Pluxy ... 311
8.4.3.5 WBI ... 312
8.4.4 Bewertung der existierenden Systeme ... 312
8.5 Ein Framework zur Erweiterung des Webs: Scone ... 313
8.5.1 Charakteristika von Software-Frameworks und Einordnung von Scone ... 313
8.5.2 Plug-ins: Architekturen zur Erweiterung und Konfiguration von Software-Systemen ... 315
8.6 Die Architektur des Frameworks Scone ... 316
8.6.1 Eine erweiterte Intermediary-Architektur: *Scone Proxy* ... 317
8.6.1.1 Integration und Erweiterung des Intermediary WBI ... 319
8.6.2 Ein persönlicher, programmierbarer Web-Crawler: *Scone Robot* ... 321
8.6.2.1 Persönliche Web-Crawler ... 322
8.6.2.2 Architektur und Funktionsweise des Crawlers ... 323
8.6.2.3 Programmierung und Steuerung des Crawlers ... 325
8.6.2.4 Ein grafisches Interface für persönliche Crawler: *RobotMonitor* ... 326
8.6.3 Repräsentation und Persistierung von Web-Ressourcen: *Scone NetObjects* ... 327
8.6.3.1 Anforderungen an Persistenzmechanismen für Web-Erweiterungen ... 328
8.6.3.2 Konzepte zur Objektrepräsentation von Web-Ressourcen ... 328
8.6.3.3 Herausforderungen der technischen Identifikation und Adressierung von Web-Ressourcen ... 330
8.6.3.4 Erfahrungen mit unterschiedlichen Konzepten zur Persistierung von Web-Ressourcen in Java ... 331
8.6.3.5 Erfahrungen mit OODBs für verteilte Hypertext-Informationssysteme ... 334
8.6.3.6 Ein objektrelationaler Datenbankadapter zur leichtgewichtigen Datenpersistierung ... 335
8.6.3.7 Caching und Konsistenzsicherung von Daten in Web-Erweiterungen ... 338
8.6.3.8 Potenziale und Grenzen der Scone NetObjects für große verteilte Hypertext-Informationssysteme ... 339
8.6.4 Erfassung von Benutzeraktionen im Web: *Scone AccessTracking* ... 339
8.6.4.1 Möglichkeiten zur Erfassung von Benutzeraktionen mit Intermediaries ... 340
8.6.4.2 Architektur und Funktionsweise des *Scone AccessTracking* ... 342

8.6.4.3 Identifikation und Verwaltung von Benutzern: *Scone UserHandling* 346
8.6.4.4 Grenzen der Erfassung von Benutzeraktionen mit Intermediaries 346
8.6.5 Hilfskomponenten für das Framework zur Erweiterung des Webs 347
8.6.5.1 Ein leichtgewichtiger Dienst zur verbindungsorientierten, asynchronen Kommunikation im Web: *Scone RAS* 348
8.6.5.2 Ein performanter, fehlertoleranter HTML-Parser: *HTMLStreamTokenizer* .. 349
8.6.5.3 Ein Interpreter für Web-Dokumente: *Scone DocumentParser* 350
8.6.5.4 Entfernte Steuerung des Browsers: *Scone BrowserControl* 351
8.6.5.5 Zur grafischen Repräsentation von Webseiten: *Thumbnail-Generator* 351
8.6.5.6 Der Intermediary als Reverse Proxy: *Scone ServerSide-Plug-in* 353
8.6.5.7 Evaluation von Websites und Scone Plug-ins: *TEA UserTestTool* 354
8.6.6 Anwendung der Plug-in-Schnittstelle des Frameworks 359
8.6.6.1 Konzepte zur Konfiguration von Plug-ins 361
8.7 Beispiele zum Einsatz des Frameworks: entwickelte Prototypen.....363
8.8 Möglichkeiten und Grenzen von Intermediary-Frameworks zur Erweiterung der Benutzungsschnittstelle des Webs.....366
8.9 Alternative Techniken zur Entwicklung von Browser-Erweiterungen 368
8.10 Fazit.....370

9 Resümee, Diskussion und Ausblick 373
9.1 Beiträge der Arbeit.....374
9.2 Diskussion.....376
9.2.1 Diskussion aktueller Entwicklungen 376
9.2.2 Zur Einführung erweiterter Link-Benutzungsschnittstellen in der Praxis.....380
9.2.3 Browser-Erweiterungen zur Vereinfachung der Link-Navigation.....382
9.3 Ausblick.....385
9.3.1 Bedeutung der Ergebnisse der Arbeit für den mobilen Zugriff auf das Web.....385
9.3.2 Erweiterung der Link-Schnittstelle für neue Geräteklassen.....386
9.3.3 Barrierefreiheit und erweiterte Link-Schnittstellen im Web.....388
9.3.4 Zukünftige Schritte.....390

10 Literaturverzeichnis.....393

Abbildungsverzeichnis.....445

Glossar.....453

Anhang: Konzepte, Schnittstellen und Standards für Hypertext und Hyperlinks A-1

A Die Konzepte der Hypertext-Pioniere A-1

A.1 Vannevar Bushs Visionen zur Erweiterung des menschlichen Geistes A-2

A.2 Doug Engelbarts Konzepte zur Unterstützung der kooperativen kreativen Arbeit .. A-3

A.3 Ted Nelsons Vision eines revolutionären globalen Hypertext-Systems für alle A-5

A.4 Tim Berners-Lee: Globaler, plattformunabhängiger Hypertext zum Wissensaustausch A-6

A.5 Vier Visionen – ein Konzept A-10

B Benutzungsschnittstellen von Hyperlinks in assoziativen Hypertext-Systemen... B-1

B.1 Memex: Auf assoziativen Pfaden durch die persönliche Bibliothek B-1

B.2 NLS / Augment: Kollaborativer Hypertext für den Knowledge Worker B-3

B.3 HES / FRESS: Textverarbeitung mit bidirektionalen Links und Annotationen B-5

B.4 ZOG / KMS: Hypertext auf Karten mit hierarchischen und assoziativen Links B-7

B.5 Xanadu: Ein globales Informationssystem mit Versionsverwaltung und Micro-Payments B-9

B.6 TextNet: Typisierte Links zur Unterstützung des wissenschaftlichen Diskurses..... B-11

B.7 Guide: Dokumentenverwaltung mit Stretchtext und adaptivem Mauszeiger B-13

B.8 HyperTies: Embedded Menues, Link-Previews und viele Navigationswerkzeuge. B-15

B.9 NoteCards: Übersichtskarten und programmierbare Links B-17

B.10 Intermedia: Ein Mehrbenutzer Client-Server-Framework für Hypermedia B-20

B.11 HyperCard: Ein flexibles Hypertext-Autorensystem B-23

B.12 Sun's Link Service: Hypertext als integrierbarer Dienst B-24

B.13 Microcosm/DLS: Offener Hypertext, generische Links und multiple Link-Datenbanken B-26

B.14 Hyper-G: Ein offenes, verteiltes Hypertext-Content-Management-System B-28

B.15 Das Web: verteilt, skalierbar, universell B-32

B.16 Weitere Hypertext-Systeme B-38

B.17 Resümee B-39

C Hyperlinks und Link-Anker in Hypertext-Standards C-1

C.1 Das Dexter-Referenzmodell C-1

C.1.1 Die Schichten des Dexter-Modells C-2

C.1.2 Komponenten im Dexter-Modell C-3

C.1.3 Link-Anker im Dexter-Modell C-4

C.1.4 Hyperlinks im Dexter-Modell ... C-5
C.1.5 DeVise / DHM – Erfahrungen mit dem Dexter-Modell ... C-5
C.1.6 Erweiterungen am Link-Modell ... C-6
C.1.7 Die Benutzungsschnittstelle des DHM-Systems ... C-6
C.1.8 Das Amsterdam Hypermedia Model ... C-7
C.1.9 Die Benutzungsschnittstelle des AHM ... C-8
C.1.10 Zusammenfassende Betrachtung zum Dexter-Modell ... C-9
C.2 HyTime ... C-9
C.2.1 Zusammenfassende Betrachtung zu HyTime ... C-12
C.3 Die XML Linking Language ... C-13
C.3.1 XLink ... C-14
C.3.2 XPath ... C-16
C.3.3 XPointer ... C-17
C.3.4 Die Benutzungsschnittstelle von XML Linking ... C-17
C.3.5 XLinks in der Praxis: Das Xspect-System ... C-18
C.3.6 Zusammenfassende Betrachtung von XLink ... C-19
C.3.7 Weitere Hypertext-Standards ... C-21
C.4 Fazit: Hypertext-Standards aus Sicht der Benutzbarkeit ... C-22

Anmerkung zu den gewählten Schreibweisen

Diese Arbeit verwendet die aktuelle „neue“ Rechtschreibung von August 2006, die einem zahlreiche Freiräume lässt. Bezüglich der Interpunktion wird, sofern die Kommasetzung optional ist, immer dann ein Komma gesetzt, wenn es der Lesbarkeit des Textes zuträglich ist.

Aufgrund des Themas findet man in dieser Arbeit viele englischsprachige Begriffe. Wort-Kombinationen mit englischen Lexemen wie „Link“, „Hypertext“ oder „Server“ wurden meist mit einem Bindestrich zusammengefasst und nicht in einem Wort geschrieben (z. B. Link-Anker, Hypertext-Dokument, Client-Server-Architektur), sofern sie noch keinen Einzug in den aktuellen Duden gefunden haben (z. B. Webseite, Webbrowser, Webserver).

In dieser Arbeit wird keine geschlechtsspezifische Schreibweise verwendet, da sie aus Sicht des Autors für die Lesbarkeit häufig nicht förderlich ist. In der Regel wird die kürzere männliche Form gewählt (z. B. *Benutzer* statt *Benutzerin*, *BenutzerIn, Benutzer/in oder BenutzerInnen*). Dennoch sind weibliche Benutzerinnen mit dieser Arbeit genauso angesprochen und wurden ebenso berücksichtigt wie männliche.

1 Einleitung und Motivation

Informationen sind zu einer zentralen Ressource geworden. Der schnelle Zugang zu ihnen ist von kaum zu unterschätzender Bedeutung. Gleichzeitig haben die Menge der zugreifbaren Informationen und die Geschwindigkeit ihrer Verbreitung in den letzten Jahrzehnten stetig zugenommen. Dieser Fortschritt ist entscheidend durch die Entwicklung des Internets mitgeprägt; insbesondere das World Wide Web hat innerhalb der letzten 20 Jahre den globalen Wissensaustausch revolutioniert.

Der gewaltige Umfang zugreifbarer Informationen ist mit zahlreichen Problemen verbunden. Um die Handhabbarkeit der stetig wachsenden Informationsbestände zu vereinfachen, wurde bereits in den 1930er Jahren ein Konzept erdacht, das den menschlichen Zugang zu und den Umgang mit Informationen vereinfachen soll, indem dem Leser *assoziative Verweise* zwischen den Dokumenten angeboten werden (s. Kapitel 2.1.2). Dieses Konzept wurde in den 1970er Jahren als *Hypertext* bekannt (s. Kapitel 2.1.4). Die charakteristischen *Hyperlinks* ermöglichen es, beliebige Ressourcen miteinander zu verknüpfen, sodass Leser von einem Dokument direkt auf weitere inhaltlich verwandte Informationen zugreifen können.

Das World Wide Web ist das bislang größte und erfolgreichste Informationssystem, das basierend auf dem Hypertext-Konzept entwickelt wurde. Seine Dominanz hat es zum De-facto-Standard für Informationssysteme gemacht. Es wird inzwischen von über 70% der Deutschen regelmäßig genutzt (Destatis 2009; ZDF 2011) und ist für Jugendliche zum Informationsmedium Nummer eins avanciert (Bagemihl, Kühte et al. 2010). Weltweit hat es über zwei Milliarden Anwender (comScore 2009; Acharya 2010), und ein Großteil nutzt das Web mehrfach die Woche (Fennah & Botterill 2010). Es ist aus vielen Bereichen des privaten, akademischen und beruflichen Lebens nicht mehr wegzudenken. Dies verleiht der *Benutzbarkeit* des Webs eine entscheidende Bedeutung, zumal ergonomische Softwaresysteme die Effizienz und Zufriedenheit bei der Arbeit steigern können und bei großen Anwenderzahlen bereits kleine Benutzbarkeitsdefizite potenziell volkswirtschaftlich relevante Auswirkungen haben (Nielsen 2002). Interessanterweise gibt es einige grundlegende Probleme bei der Benutzung von Hypertext, die seit Langem bekannt sind und ebenfalls im Web auftreten, für die bislang aber keine Lösungen gefunden wurden (s. Abschnitt 1.1).

Diese Situation war Motivation dafür, sich im Rahmen einer Dissertation mit der Ergonomie von Hypertext-Informationssystemen am Beispiel des Webs auseinanderzusetzen und neue Konzepte zur Reduktion der vorhandenen Defizite zu entwickeln.

1.1 Forschungsproblem und Zielsetzung

Diese Arbeit versucht, grundlegende Benutzbarkeitsfaktoren von großen verteilten Hypertext-Informationssystemen am Beispiel des Webs zu ermitteln und möglichst allgemeingültige Konzepte zur Verbesserung ihrer Gebrauchstauglichkeit zu entwickeln.

Hyperlinks sind das am häufigsten verwendete Navigationsmittel (Weinreich, Obendorf et al. 2006b; Dubroy & Balakrishnan 2010), das insbesondere im Web von essenzieller Bedeutung ist, da das Web im Gegensatz zu den meisten anderen Hypertext-Systemen keine weiteren systemimmanenten Navigationsmöglichkeiten bietet, wie hierarchische Strukturen oder integrierte Suchsysteme. Sogar die für den Zugriff auf neue Informationen und Angebote unverzichtbaren globalen Suchmaschinen benötigen die Verlinkung der Ressourcen des Webs zu ihrer Indexierung und Relevanzbewertung.

Hyperlinks tragen nicht nur wesentlich zur Vereinfachung beim Umgang mit Informationen bei, mit ihnen sind auch zahlreiche Benutzbarkeitsprobleme verbunden. Hierzu gehören Navigations- und Orientierungsprobleme wie das *„Lost-in-Hyperspace"-Phänomen* und der *Cognitive Overhead*, der sich aus den vielen zusätzlichen Navigationsentscheidungen beim Lesen von Hypertext ergibt (Conklin 1987). Die globale Verteilung der Informationen und die mangelnde Konsistenzsicherung des Webs führen zu weiteren Herausforderungen bei der Link-Navigation.

Obwohl Hyperlinks das maßgebliche Navigationsmittel im Web darstellen, wurde ihre Benutzungsschnittstelle weder systematisch entwickelt noch explizit evaluiert. So wurde bisher nur unzureichend erforscht, wie Links dargestellt werden sollten, welche Informationen Benutzer über Links und die erreichbaren Zielobjekte benötigen und welche Kriterien für eine ergonomische Interaktion mit Hyperlinks zu beachten sind. Darüber hinaus bleiben im Web viele Faktoren der Link-Gestaltung den Autoren überlassen; eine inkonsistente Benutzungsschnittstelle ist die Konsequenz.

Diese Arbeit hat das Ziel, basierend auf Ergebnissen der Hypertext- und der Web-Forschung und ausgehend von aktuellen Techniken für grafische Benutzungsoberflächen, systematisch neue Konzepte zu entwickeln, mit denen sich die Schwächen der Link-Navigation reduzieren lassen. Sie sollen die Benutzbarkeit von Links im Allgemeinen verbessern, also potenziell für jeden Benutzer und beliebige Hypertext-Informationssysteme einsetzbar sein. Da dies am Beispiel des Webs erfolgt, muss die Kompatibilität mit dem jetzigen Web berücksichtigt werden. Gleichzeitig dürfen die neu zu entwickelnden Konzepte den Web-Autoren keine neuen, zusätzlichen Herausforderungen stellen, sondern sollten unmittelbar und einfach einsetzbar sein und gegenwärtige Web-Angebote von sich profitieren lassen.

Dieses weit gefasste Ziel bedarf einiger Eingrenzungen, um als Dissertationsprojekt durchführbar und Erfolg versprechend zu sein.

Erstens steht das World Wide Web als verteiltes Hypertext-System zur *Informationsrecherche* im Fokus, bei dem Anwender mittels gezielter Navigation versuchen, auf bestimmte Informationen zuzugreifen. Das Web ist inzwischen zwar weitaus mehr als eine reine Informationsquelle: Beispielsweise dient es auch als Kommunikationsmedium und als E-Commerce-Plattform (Fennah & Botterill 2010). Hieraus ergeben sich weitere Herausforderungen bezüglich der Benutzungsschnittstelle von Webseiten als Online-Applikationen, die jedoch auf-

grund ihrer kaum eingrenzbaren Vielgestaltigkeit in dieser Arbeit keine Berücksichtigung finden (s. Kapitel 2.2.3 und Kapitel 9.2).

Zweitens bezieht sich diese Arbeit primär auf Benutzer ohne situative oder physische Einschränkungen, die einen *grafischen Webbrowser* mit einer *Desktop-Oberfläche* wie Microsoft Windows, Apple OSX oder KDE einsetzen. Obgleich die Online-Nutzung mit Mobilgeräten, Tablet-PCs und Smartphones ständig an Bedeutung gewinnt, bleiben bis heute Einzelplatzrechner, Laptops und Netbooks mit Desktop-Interface und Mausbedienung die bedeutendsten Plattformen für den Einsatz des Webs (Fennah & Botterill 2010). Zahlreiche Ergebnisse der Arbeit lassen sich dennoch auf andere Geräteklassen übertragen und können gleichzeitig zu einer besseren Barrierefreiheit des Webs beitragen (s. Kapitel 9.3.3).

1.2 Forschungsansatz und Vorgehen

Ein methodisches Vorgehen für eine Verbesserung der Benutzbarkeit von Hyperlinks in großen verteilten Informationssystem erfordert als Erstes die Identifikation der wesentlichen Benutzbarkeitsprobleme bei der Interaktion mit Links und ihrer Ursachen. Diese Arbeit berücksichtigt dabei nicht nur Forschungsergebnisse aus dem Bereich der Web-Usability, sondern auch Publikationen und Projekte aus der Hypertext-Forschung, da sich Experten bereits lange vor der Entwicklung des Webs intensiv mit den Problemen der Link-Navigation auseinandergesetzt und Erkenntnisse über die Ergonomie erlangt haben.

Als wesentlicher Ansatz zur Verbesserung der Benutzbarkeit von Hypertext wurde in früheren Arbeiten die Verringerung des *Informationsdefizits* der Anwender bei ihren Navigationsentscheidungen durch eine *Typisierung* der Links identifiziert (Spyridakis 1989; Kopak 2000; Mobrand & Spyridakis 2002): Hierbei spezifiziert der Link-Autor, in welcher inhaltlichen Relation das Zieldokument zum Ausgangsdokument steht und gibt dem Leser so eine genauere Vorstellung über Bedeutung und Funktion des Links (s. Kapitel 4ff; Trigg 1983). Die Link-Typisierung war auch bei der Konzeption des Webs vorgesehen (Berners-Lee 1989), sie hat sich aber weder hier noch in anderen Systemen aufgrund von Autorenproblemen bei der Typ-Definition praktisch durchgesetzt (vergl. Kapitel 4.3).

Eine später entwickelte Alternative sind *Link-Previews*, die dem Leser ausgewählte, bereits vorhandene Informationen zum Zielobjekt vor dem Folgen des Links zugänglich machen. Dieses Konzept lieferte bereits früh bei lokalen Hypertext-Systemen vielversprechende Ergebnisse (Utting & Yankelovich 1989). Dennoch bleiben Nutzern des Webs zumeist nicht nur viele Informationen zum Link-Ziel – wie Inhalt, Dateityp, Anbieter und Zugreifbarkeit –, sondern oft auch zur Link-Aktion (wird beispielsweise ein neues Fenster geöffnet oder ein Download gestartet) verborgen, obgleich solche Eigenschaften Relevanz für die Navigation und Orientierung haben. Diese Arbeit erweitert daher das Konzept der Link-Previews am Beispiel des Webs zum Einsatz in offenen, verteilten Informationssystemen.

Analyse und Klassifikation der Link-Schnittstelle

Der Weiterentwicklung der Link-Previews sind systematische Voruntersuchungen vorausgegangen. Erstens erfolgte eine Analyse der grundlegenden Eigenschaften von Links und Ziel-Objekten im Web. Sie resultierte in einer Kategorisierung der implizit bereits verfügbaren Link-Informationen, die auch die potenzielle Bedeutung der einzelnen Informationen für die Navigation berücksichtigt. Zweitens lieferte eine Analyse und Klassifikation existierender Methoden zur Darstellung von *Link-Ankern* und *Link-Hinweisen* in Hypertext-Systemen die notwendigen Erkenntnisse zur Gestaltung ergonomischer Link-Previews. Die Resultate beider Studien waren Basis für die Konzeption von *HyperScout*, einer erweiterten Link-Benutzungsschnittstelle zur Anzeige von Vorabinformationen direkt beim Link-Anker.

Entwicklung und Evaluation einer erweiterten Link-Schnittstelle

HyperScout wurde prototypisch in zwei Varianten realisiert. Die erste Version HyperScout I nutzt strukturierte Tooltips zur Darstellung der zusätzlichen Link-Hinweise beim Link-Anker. Mithilfe von Benutzerstudien wurde ermittelt, welche der zusätzlichen Informationen für Benutzer hilfreich sind und ob die neue Benutzungsschnittstelle mit den Interaktionsgewohnheiten der Web-Anwender harmoniert.

Die Evaluationsergebnisse von HyperScout I führten zur Entwicklung eines zweiten Prototyps HyperScout II, der über optimierte Tooltips verfügt und zusätzlich multiple Maus-Icons und Link-Overlays zur Darstellung der *Preview-Informationen* einsetzt. HyperScout II wurde ebenfalls partizipativ evaluiert, um die Stärken und Schwächen der einzelnen Darstellungstechniken zu vergleichen und Rückschlüsse für zukünftige Entwicklungen zu ziehen.

Erweiterung der Protokolle und Sprachen des Webs

Der praktische Einsatz von HyperScout im Web setzt neue Techniken zur Datenübertragung voraus, die dafür sorgen, dass Benutzern zu allen Link-Ankern einer Seite zusätzliche Informationen über die Links und Zielobjekte bei Bedarf verzögerungsfrei dargestellt werden. Diese Übertragungstechniken müssen kompatibel mit dem jetzigen Web und unabhängig von Webbrowsern und Servern sein und sollen sich möglichst auch für andere Schnittstellenkonzepte als HyperScout einsetzen lassen. Zu diesem Zweck wurden neue Erweiterungen des HTTP-Protokolls und der Sprache XHTML entwickelt.

Softwaretechnische Realisierung der entwickelten Konzepte

Im Rahmen dieser Arbeit wurden die HyperScout Link-Benutzungsschnittstelle und die Protokoll- und Spracherweiterungen prototypisch implementiert. Die effiziente Realisierung der verschiedenen Prototypen setzte eine gemeinsame technische Basis voraus. Da kein Framework den Anforderungen für die Implementation der geplanten Systeme genügte, wurde das neue Framework *Scone* konzipiert und softwaretechnisch realisiert. Seine

komponentenbasierte Architektur bietet die Funktionalität und Flexibilität, die für das HyperScout-Projekt erforderlich war.

1.3 Aufbau der Arbeit

Diese Dissertation hat neun Kapitel und einen dreiteiligen Anhang.

Das folgende **zweite Kapitel** führt in die für diese Arbeit wesentlichen *Grundlagen von Informationssystemen* und *Hypertext* ein. Es werden die Begriffe Information, Assoziation und Hypertext hergeleitet und die wichtigsten Paradigmen und Konzepte von Hypertext-Informationssystemen definiert.

Kapitel drei befasst sich mit der *Benutzbarkeit verteilter Hypertext-Informationssysteme*. Erst werden die Navigationswerkzeuge in Informationssystemen verglichen und ihre Stärken und Schwächen untersucht. Es folgt eine Analyse der grundlegenden Benutzbarkeitsprobleme von Hypertext und die besonderen Herausforderungen großer verteilter Hypertext-Systeme wie dem World Wide Web. Dabei zeigt sich, dass *Hyperlinks* für alle Aspekte der Gebrauchstauglichkeit eine besondere Bedeutung haben.

Kapitel vier hat den Schwerpunkt *Benutzbarkeit von Hyperlinks*. Die zur Reduzierung der Benutzbarkeitsdefizite von Hyperlinks entwickelten Konzepte zur *Link-Typisierung* werden vorgestellt und mit *Link-Previews* verglichen. Eine Auswertung der existierenden Möglichkeiten zur Link-Typisierung im Web zeigt die Schwächen der aktuellen Standards auf. Die folgende erstmalig vorgenommene *Klassifikation* der vielfältigen vorhandenen *impliziten Link-Informationen im Web* verdeutlicht die Potenziale von Link-Previews im Web.

Das **fünfte Kapitel** befasst sich mit der Konzeption einer *erweiterten Hyperlink-Benutzungsschnittstelle* für die Darstellung von Preview-Informationen im Web. Erst werden Anforderungen und Ziele für die Gestaltung ergonomischer Link-Schnittstellen identifiziert. Dann erfolgt eine Klassifikation der Visualisierungstechniken für Link-Anker und zusätzliche Link-Informationen, und die Potenziale und Grenzen bezüglich ihrer Nutzung in grafischen Webbrowsern werden ausgewertet. Diese Studien führen zum Entwurf neuer Konzepte für erweiterte Link-Schnittstellen im Web, die es erlauben, zusätzliche Link-Informationen als Vorschau darzustellen. Das hierauf basierende System trägt den Namen *HyperScout*.

Kapitel sechs beschreibt die *Evaluation* und *Überarbeitung* der neuen Link-Schnittstelle. Mithilfe des partizipativen Testverfahrens „Thinking Aloud“ werden Stärken und Schwächen des ersten Prototyps *HyperScout I*, der strukturierte Tooltips zur Anzeige von Previews einsetzt, ermittelt. Um die Bedeutung der zahlreichen Kategorien von Preview-Informationen für die Navigation im Web genauer zu bestimmen, erfolgte zusätzlich eine Online-Umfrage mit 138 Teilnehmern. Die Ergebnisse beider Studien führten zur Entwicklung des optimierten Konzepts *HyperScout II*, das optimierte Tooltips, Maus-Icons und Link-Overlays zur Anzeige weiterer Informationen einsetzt. Dieser zweite Prototyp wurde ebenfalls mit

„Thinking-Aloud-Tests“ untersucht. Er weist wesentliche Vorteile gegenüber dem Vorgänger auf.

HyperScout setzt neue Techniken zur effizienten Übertragung zusätzlicher Informationen zu allen Links und Zielobjekten der aktuellen Webseite voraus. In **Kapitel sieben** geht es um *Protokolle* und *Dienste* zur Bereitstellung der Daten für die erweiterte Link-Schnittstelle. Erst werden die grundlegenden Konzepte zur Bereitstellung der zusätzlichen Daten diskutiert, dann wird ein neues Übertragungskonzept entworfen und prototypisch realisiert, das mit dem jetzigen Web kompatibel, flexibel einsetzbar und erweiterbar ist und nur eine geringe zusätzliche Bandbreite erfordert. Es besteht aus einer Erweiterung des HTTP-Protokolls und der Sprache XHTML.

Das **achte Kapitel** handelt von der *softwaretechnischen Realisierung* der entwickelten Konzepte. Es beginnt mit einer Untersuchung der verfügbaren Methoden zur Erweiterung von Webbrowsern und Servern. Das Konzept des Web-Intermediary stellt sich dabei als am Flexibelsten heraus und wird genauer charakterisiert. Der eigentlichen Implementation ging eine Analyse existierender Projekte und Frameworks voraus, die sich als Grundlage für die Implementation eignen könnten. Da keines hiervon die für diese Arbeit benötigte Flexibilität und Funktionalität bot, wurde ein neues Java-Framework mit dem Namen Scone konzipiert und realisiert. Scone besteht aus mehreren programmier- und erweiterbaren Komponenten: Neben einem Intermediary gehören hierzu ein Web-Crawler, Komponenten zur Aufzeichnung von Benutzeraktionen und zur Repräsentation und Persistierung von Web-Daten. Hilfskomponenten zur Analyse und Verarbeitung von Web-Ressourcen ergänzen das Framework. Abschließend werden einige der mit Scone programmierten Systeme vorgestellt und die Potenziale und Grenzen des Frameworks beleuchtet.

Das letzte, **neunte Kapitel** fasst die Beiträge der Arbeit zusammen und diskutiert die Potenziale und Grenzen für die praktische Einführung der entwickelten Konzepte. Dann wird ein Bezug der Ergebnisse zu aktuellen und sich abzeichnenden Entwicklungen hergestellt; hierzu gehören die neuen Anforderungen für Benutzungsschnittstellen bei Mobilgeräten und Tablet-PCs. Ein *Ausblick* auf zukünftige Möglichkeiten zur Weiterführung der in dieser Arbeit entwickelten Konzepte schließt das Kapitel ab.

Es folgt ein dreiteiliger **Anhang**, der sich mit grundlegenden Aspekten der Hypertext-Theorie auseinandersetzt. Die in diesem Rahmen durchgeführten Untersuchungen lieferten wesentliche wissenschaftliche Grundlagen für die Entwicklung der neuen Link-Schnittstellenkonzepte von HyperScout und geben einen fundierten Einblick in frühere Ergebnisse der Hypertext-Forschung. **Teil A** des Anhangs stellt die *Visionen und Konzepte* der vier für das Hypertext-Konzept wegbereitenden *Pioniere* vor. Dabei zeigt sich, dass viele ihrer Ideen und Ziele auch für heutige Entwicklungen noch von großer Aktualität sind. **Anhang B** umfasst eine Analyse der *Benutzungsschnittstelle von 16 wegweisenden Hypertext-Systemen*, die erstmals den Schwerpunkt auf die Hyperlink-Schnittstelle legt. Im letzten **Abschnitt C** wird

auf die *Repräsentation von Hyperlinks und Link-Ankern* in den wichtigsten *Hypertext-Standards* eingegangen und untersucht, welche Konzepte für eine standardisierte Link-Benutzungsschnittstelle sich aus ihnen ableiten lassen.

2 Theoretische Grundlagen verteilter Hypertext-Informationssysteme

Dieses Kapitel gibt einen Überblick über die wesentlichen theoretischen Grundlagen im Bereich verteilter Hypertext-Informationssysteme. Der erste Teil 2.1 geht auf einige für diese Arbeit fundamentale Begriffe wie Information und Assoziation ein und erklärt auf ihrer Basis die grundlegenden Ideen von Hypertext. In Teil 2.2 werden erst die Konzepte und Begriffe des assoziativen Hypertext-Paradigmas beschrieben und dann die Potenziale und Herausforderungen von offenem, verteiltem Hypertext analysiert und in Bezug zu den neuen Entwicklungen im World Wide Web gesetzt.

2.1 Information, Assoziation, Hypertext

Diese Dissertation beschäftigt sich mit der Benutzbarkeit einer Kategorie von Informationssystemen, die für Menschen den Umgang mit großen Informationsbeständen vereinfachen soll, indem sie dem Leser assoziative Verknüpfungen zwischen den Datenobjekten bietet. Das als *Hypertext* bekannte Konzept stellt dabei den Menschen und seinen Umgang mit Informationen in den Fokus. Zur Erläuterung des Hypertext-Konzepts wird nachfolgend in Abschnitt 2.1.1 das wissenschaftliche Verständnis des Begriffes *Information* betrachtet. Teil 2.1.2 knüpft die Beziehung zur Kognitionspsychologie und beschreibt die Rolle der *Assoziation* beim Denken und Erinnern, um so die Potenziale der assoziativen Verweise im Hypertext auszuführen. Abschnitt 2.1.3 grenzt den weitreichenden Begriff *Informationssystem* im Kontext dieser Arbeit ein und stellt unterschiedliche Konzepte zum Benutzerzugriff auf digital gespeicherte Informationen gegenüber. Basierend auf diesen Grundlagen findet in Kapitel 2.1.4 eine Definition von Hypertext und Hypertext-Systemen statt. Es folgt ein kurzer Überblick über charakteristische Anwendungsfelder für Hypertext-Systeme und ihrer konzeptionellen Stärken (Abschnitt 2.1.5).

2.1.1 Der Informationsbegriff

Es ergibt sich als Erstes die Frage, was man unter dem Begriff *Information* genau versteht. Eine Bestimmung des Begriffes ist allerdings nicht so trivial, wie es erst erscheinen mag, da viele Erklärungsansätze ähnliche Begriffe wie „Wissen“, „Nachricht“, „Kommunikation“ voraussetzen. Solche Rückgriffe führen aber schnell zu einer zyklischen Definition.

Unterschiedliche wissenschaftstheoretische Sichtweisen bieten sehr verschiedenartige Erklärungsmodelle (Capurro 1978).

Erstmals wurde Information im Jahre 1948 von Norbert Wiener (*1894; †1964) – Begründer der Kybernetik – mit dem Satz „Information ist Information, weder Materie noch Energie“ als eine *unabhängige Grundgröße der Naturwissenschaft* postuliert (Wiener 1963: 192). Information ist für Wiener die Zunahme von *Ordnung* aufgrund der Wechselwirkung inner-

halb eines Systems oder zwischen Systemen und somit ein gegensätzlicher Prozess zur Entropie. Genau betrachtet ist Information nach dieser Definition allerdings an das Vorhandensein von Materie oder Energie gebunden, da diese als Träger der Information dienen.

Claude E. Shannon (*1916; †2001) hat im selben Jahr mit seinem Aufsatz „*A Mathematical Theory of Communication*“ (Shannon 1948) die *technische Informationstheorie* begründet. Gegenstand dieser Theorie ist die Übertragung von Zeichen, mit denen Nachrichten dargestellt werden. Ausgehend von der Frage, wie viel Information in einer möglichst kurzen Nachricht codiert werden kann, definiert Shannon in seiner Theorie den Begriff Information auf *statistische* Weise. Danach ist die *Ungewissheit* seitens des Empfängers in Bezug auf die von der Quelle ausgewählten und ausgesandten Nachrichten das Maß für die Menge der Information. Entscheidungsprozesse seitens des Empfängers verringern diese Ungewissheit und damit die Menge der zu empfangenden Informationen. Shannon beschränkt sich allerdings auf den nachrichtentechnischen Aspekt des Informationsbegriffs und lässt bewusst semantische und pragmatische Aspekte von Information außer Acht. So ist es für Shannon völlig ohne Belang, ob eine Buchstabenreihe einen sinnvollen Text darstellt oder nicht. Nach seiner Definition enthält dadurch – kaum intuitiv – eine Zufallsfolge von Buchstaben das mögliche Maximum an Informationen, da sie dem Empfänger zuvor völlig unbekannt war (Shannon 1948; Capurro & Hjørland 2003).

Von einer grundsätzlich anderen Sichtweise – die auch im Rahmen dieser Arbeit Anwendung findet – geht die *semiotische* Definition des Informationsbegriffes aus. Hier ist die Interpretation der Daten durch den Empfänger entscheidend. Die Semiotik unterscheidet zwischen unterschiedlichen Evaluationsebenen, wobei in diesem Zusammenhang die drei Ebenen der *Syntaktik*, der *Semantik* und der *Pragmatik* von Belang sind. *Daten* setzen eine *syntaktische Interpretation* voraus, durch die Gruppen von Zeichen erst als eine *Nachricht* erkannt werden können. Um aus einer Nachricht eine bedeutsame Information formen zu können, wird eine *semantische Abbildung* benötigt, mit der der Mensch die Daten einer Nachricht auf eine interpretierbare Information abbilden kann. Hierbei werden die Daten für den Menschen erst mit einem Sinn behaftet. Je nach Assoziation des Menschen kann er eine andere Sinnhaftigkeit interpretieren, wodurch die semantische Interpretation subjektiv verschieden sein kann. Der Bezug zu *Wissen* lässt sich mithilfe einer *pragmatischen Interpretation* der Informationen herstellen. Auf pragmatischer Ebene ordnet ein Mensch den Sinn einer Nachricht in den jeweiligen *Kontext* ein. So „erkennt“ er den Zweck einer Information, wobei dieses Verständnis nicht nur individuell sehr unterschiedlich sein kann, sondern auch von vielen weiteren Faktoren wie dem situativen Kontext abhängig ist (Eberleh, Oberquelle et al. 1994: 102-120; Nake 2002).

Die semiotische Definition des Begriffes Information setzt für das Verständnis einer Information selbst wieder Informationen voraus. Dies bedeutet konkret für den Menschen, dass er Vorwissen für die Aufnahme und Verarbeitung von Informationen benötigt. Gleichzeitig stellt diese Betrachtungsweise infrage, inwieweit Computer tatsächlich mit

Informationen oder doch nur mit Daten hantieren, da sie bis heute nicht in der Lage sind, eine semantische Abbildung der Daten im menschlichen Sinne zu leisten. Da aber *für den Anwender* diese Daten durchaus Informationen darstellen können und der Computer den Zugang zu diesen Daten ermöglicht und meist sogar vereinfacht, lässt sich aus Anwendersicht durchaus der Standpunkt vertreten, dass Computer dem Menschen Informationen zur Verfügung stellen können (s. 2.1.3).

2.1.2 Assoziatives Denken

Ein zweiter zentraler Begriff dieser Arbeit ist die *Assoziation*. Am stärksten verankert ist er heute in der Psychologie, wo die Assoziation als ein grundlegendes Konzept der Bewusstseins-, Kognitions- und Denkpsychologie angesehen wird (Hofstätter 1974). Die Untersuchung von Bewusstsein und Denken hat eine lange Geschichte. Bereits im Altertum haben Gelehrte über die Funktionsweise des Denkens und der Verarbeitung von Erinnerungen philosophiert. Die ersten systematischen Überlegungen zur Rolle der Assoziationen beim Lernen und Denken werden Aristoteles (384 - 322 v. Chr.) zugeschrieben. Ausgehend von dem im Gedächtnis vor sich gehenden Prozess der Erinnerung, unterscheidet Aristoteles in der Schrift *„Über das Gedächtnis und die Erinnerung"* zwei Basiselemente kognitiver Vorgänge: die Idee (der Gedanke) und die Assoziation (die Verbindung) zwischen Eindrücken. Laut Aristoteles soll der Mensch mithilfe dieser Assoziationen Gedanken erinnern und neue Ideen erschaffen können (Strube 1984). Dies führt er in seiner Schrift weiter aus und identifiziert so die drei Prinzipien der Bildung von Assoziationen im Geiste (s. Abb. 1):

1. das Gesetz des Kontrastes (warm zu kalt);

2. das Gesetz der Ähnlichkeit (Rose zu Tulpe);

3. das Gesetz der räumlichen und zeitlichen *Kontiguität* (Berührung in Raum oder Zeit, z. B. Garten zu Pflanzen oder Blüte zu Frucht).

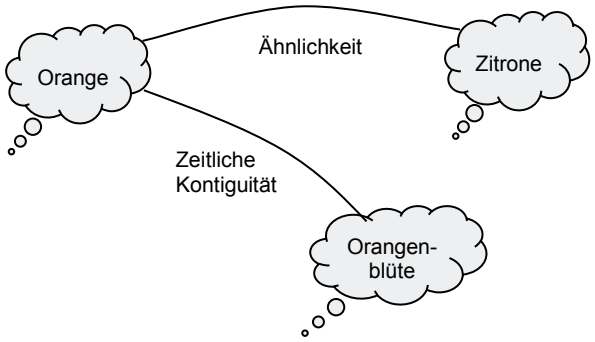


Abb. 1: Assoziationen verknüpfen im Gedächtnis vorhandene Eindrücke.

Diese drei als *Assoziationsgesetze* bekannt gewordenen Regeln haben viele nachfolgende philosophische Überlegungen entscheidend beeinflusst und den frühen systematischen Ausbau der Psychologie geprägt. Die sorgfältige Ausgestaltung dieser Theorien und Modelle zur Funktionsweise des Denkens erfolgte erst im 17. bis 19. Jahrhundert in der sogenannten

„*Englischen Assoziations-Psychologie*" durch viele namhafte Psychologen wie John Locke [*1632; †1704] und David Hume [*1711; †1776] (s. Hofstätter 1974: 29ff). Von kleineren Differenzen abgesehen, verwendet bis heute jede Lerntheorie und jede Lehre vom Gedächtnis den Begriff der Assoziation in entscheidender Funktion, wobei allerdings verschiedene Deutungen der ihm zugrunde liegenden Vorgänge getroffen wurden.

Bahnbrechend waren in diesem Zusammenhang die Arbeiten des deutschen Psychologen Hermann Ebbinghaus (*1850; †1909), dem die ersten systematischen Untersuchungen der höheren mentalen Prozesse im Gehirn zugeschrieben werden. Er hat die Bedeutung assoziativen Denkens beim Lernen und Erinnern erstmals experimentell nachgewiesen (Gorfein & Hoffmann 1987). Das heutige Verständnis des Begriffes Assoziation basiert auf seinen Arbeiten. Im psychologischen Wörterbuch von Dorsch – einem der Standardwerke der Psychologie – findet sich folgende daraus erwachsene Definition:

> Assoziation […], eine Verknüpfung seelischer Inhalte, die sich darin zeigt, daß das Auftreten des einen das Bewußtwerden des anderen (mit ihm assoziierten) nach sich zieht oder wenigstens begünstigt (Dorsch 1994: 60ff).

Von dieser psychologischen Definition wird auch im Rahmen dieser Arbeit ausgegangen, da es beim Hypertext um die Unterstützung des menschlichen Denkens und geistigen Arbeitens durch die Maschine geht.

2.1.3 Informationssysteme

Der Begriff *Informationssystem* ist kaum eindeutig zu definieren, da seine Bedeutung wesentlich vom wissenschaftlichen Standpunkt des Betrachters abhängt. Es gibt Bestrebungen, das Verständnis dieses in vielen Bereichen der Informatik zentralen Begriffs zu standardisieren. Beispielsweise deutet das renommierte amerikanische *Committee on National Security Systems* den Ausdruck folgendermaßen:

Information System (IS)	Set of information resources organized for the collection, storage, processing, maintenance, use, sharing, dissemination, disposition, display, or transmission of information (Hayden 2006: 33).

Diese Definition lässt einen weiten Interpretationsspielraum zu und veranschaulicht die Problematik einer klaren Abgrenzung des Begriffs. Hinzu kommt, dass es diverse andere „offizielle" Definitionen verschiedener Institute gibt, und selbst diese sind häufig vielgestaltig; das ANSI (*American National Standards Institute*) hat beispielsweise gleich drei Definitionen bestätigt (ATIS 2011).

Im Kontext dieser Arbeit bietet sich eine Definition für Informationssysteme an, der eine *semiotische Betrachtung von Informationen* zugrunde liegt (s. Abschnitt 2.1.1). Um ein Datenverarbeitungssystem von einem Informationssystem zu unterscheiden, setzt dies voraus, dass die Daten mithilfe einer semantischen Abbildung interpretiert werden können. Eine Interpretation könnte mittels komplexer Regeln und Grammatiken erfolgen, andererseits kann auch einfach der Mensch als interpretierender Faktor gesehen werden.

Da es bei dieser Dissertation nicht um Mechanismen künstlicher Intelligenz geht, sondern die *Benutzbarkeit* einer bestimmten Klasse von Informationssystemen im Blickfeld steht, spielt der Bezug zum Menschen eine entscheidende Rolle. Die folgende Definition berücksichtigt dies und dient als Begriffsbildung für diese Arbeit. Sie basiert auf einer soziotechnischen Definition, die von Wolfgang Krcmar und anderen geprägt wurde (König 1994; Krcmar 2004: 25; Schwarzer & Krcmar 2004: 11):

> Informationssysteme sind soziotechnische Systeme, die bestehend aus Teilsystemen zum Ziel der optimalen Bereitstellung von Information und Kommunikation eingesetzt werden.

Ein Informationssystem besteht demnach immer aus *Menschen und Maschinen*, die mit Informationen arbeiten und die durch Interaktions- und Kommunikationsbeziehungen miteinander verbunden sind. Diese Sichtweise grenzt das *Informationssystem* gleichzeitig vom *Datenverarbeitungssystem* durch die *Ausrichtung der Schnittstelle auf den Menschen* ab.

Aber selbst diese soziotechnische Eingrenzung des Begriffes Informationssystem lässt recht unterschiedliche Perspektiven zu. Im betrieblichen Umfeld dient das Informationssystem der Abbildung von Prozessen und Austauschbeziehungen zwischen dem Betrieb und seiner Umwelt. Ein rechnergestütztes Informationssystem ist hingegen ein System, das allgemein die Erfassung, Speicherung und Transformation von Informationen unterstützt und teilweise automatisiert. In der Praxis besteht dies typischerweise aus einer Menge von unabhängigen, aber kooperierenden Systemen, die zusammen die angestrebte Leistung erbringen (Krcmar 2004).

Beispiele für Informationssysteme in diesem Sinne sind *Information-Retrieval-Systeme*, die der effizienten Suche nach Informationen in großen Datenmengen dienen, beispielsweise in digitalen Bibliotheken (Belkin & Croft 1987). Dabei ist ein häufiges Problem, dass Benutzer nicht exakt formulieren können, was sie suchen oder sogar nur eine vage Angabe der gesuchten Information machen können (s. Kapitel 3.1.1). Das System soll ihnen dennoch mit möglichst hoher Genauigkeit das Gewünschte liefern (Ferber 2003: 33; Lewandowski & Höchstötter 2007).

Eine weitere Klasse soziotechnischer Informationssysteme sind *Expertensysteme*. Sie versuchen in einem thematisch eingeschränkten Bereich mittels Methoden der künstlichen Intelligenz das Spezialwissen und die Schlussfolgerungsfähigkeit qualifizierter Fachleute nachzubilden, um so Menschen bei Entscheidungen und Problemlösungen zu helfen (Puppe 1991). Man hoffte in den achtziger Jahren, der anwachsenden Informationsflut Herr zu werden, indem Expertensysteme dem Menschen die jeweils gerade benötigten Informationen bereitstellen.

Eine dritte Klasse, die im Mittelpunkt dieser Arbeit steht, sind *Hypertext-Informationssysteme*. Sie zeichnen sich durch die Vernetzung der Informationen und durch eine charakteristische Benutzungsschnittstelle aus, die einen möglichst intuitiven und benutzergerechten Zugriff auf Informationen erlaubt – statt dem Menschen das Denken und die Suche nach

Informationen abzunehmen, soll das Hypertext-System den Menschen bei seinen geistigen Aufgaben optimal unterstützen.

2.1.4 Hypertext

Der Begriff *Hypertext* wurde bereits in der Pionierzeit der digitalen Computertechnik geprägt. Ersonnen wurde er von Ted Nelson, der ihn erstmals 1965 in einer Präsentation auf der Konferenz der *Association for Computing Machinery* in Pittsburgh verwendete (Gillies & Cailliau 2001). In der zugehörigen Publikation definierte er Hypertext folgendermaßen:

> Let me introduce the word "hypertext" to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper. It may contain summaries, or maps of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it.

Und in einer Fußnote ergänzte er:

> The sense of 'hyper-' used here connotes extension and generality ; cf. 'hyperspace.' The criterion for this prefix is the inability of these objects to be comprised sensibly into linear media, like the text string, or even media of somewhat higher complexity (Nelson 1965).

Als Kernidee seines neuen Konzepts sah er damit eine komplexe Verknüpfung des Materials, die eine lineare Wiedergabe auf Papier ausschließt. Er berücksichtigte auch Möglichkeiten von öffentlich zugänglichen Annotationen und Fußnoten sowie die Ergänzung des Materials durch entsprechende Gelehrte.

Ziel dieser Idee war es, den Umgang mit Informationen für alle Menschen einfacher und effizienter zu gestalten. Die assoziativen Verknüpfungen sollten den Leser auf natürliche Art zu weiteren Informationen führen, vergleichbar mit den geistigen Assoziationen des Gehirns, die eine Grundlage des Denkens und Erinnerns darstellen (s. Abschnitt 2.1.2).[1] Nelson griff dabei auf Konzepte von Vannevar Bush zurück (s. Anhang A.1), der bereits über 20 Jahre zuvor den Zugriff auf Informationen mithilfe eines Memex (s. Anhang B.1) genanntem technischen Systems mit assoziativen Verknüpfungen vereinfachen wollte.

Conklin bringt später die Kernidee dieses Konzepts auf den Punkt und beschreibt als die zwei charakterisierenden Eigenschaften von Hypertext die „*node-ness*" und die „*linked-ness*" (Conklin 1987). Konkret drückt er damit aus, dass die Inhalte in relativ kleine ***Knoten*** *(im Folgenden auch Dokument, Objekt und Ressource genannt)* aufgeteilt werden, die durch viele assoziative ***Links*** *(im Folgenden auch Hyperlink und Verknüpfung genannt)* miteinander ver-

[1] Dies soll nicht heißen, dass Hypertext *wie* das Gehirn oder das Denken des Menschen ist. Die Funktionsweise des Gehirns ist bis heute zu wenig verstanden und zu komplex, als dass ein solcher Vergleich sinnvoll wäre. Des Weiteren sind Assoziationen nur ein Aspekt beim menschlichen Denken und Erinnern, es gibt noch viele weitere Mechanismen (siehe beispielsweise die hierarchische Strukturierung von Konzepten, Abschnitt 3.1.2: Hierarchische Navigation). Es wird bei Hypertext lediglich genutzt, dass Menschen beim Denken und Erinnern auf Assoziationen zurückgreifen (McKendree, Reader et al. 1995). Hypertext kann mithilfe inhaltlicher Verknüpfungen diese Vorgänge unterstützen und effizienter zu Informationen führen, die in dem momentanen Zusammenhang ebenfalls von Interesse oder sogar notwendig sind.

bunden sind (s. Abb. 2). Dies hat zur Folge, dass der Leseverlauf nicht mehr vorgegeben ist, sondern der Leser jeweils entscheiden kann, welcher Verknüpfung er folgt und wie er die Informationen aufnehmen will (s. auch Abschnitt 2.2.1).

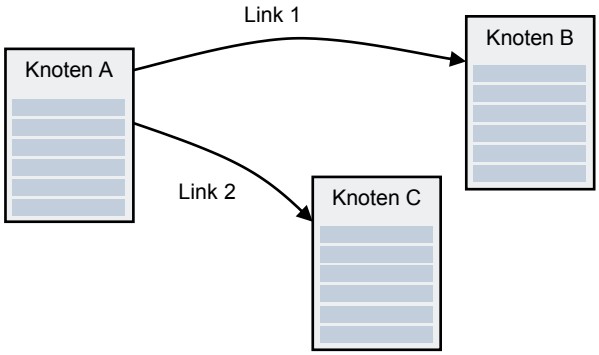
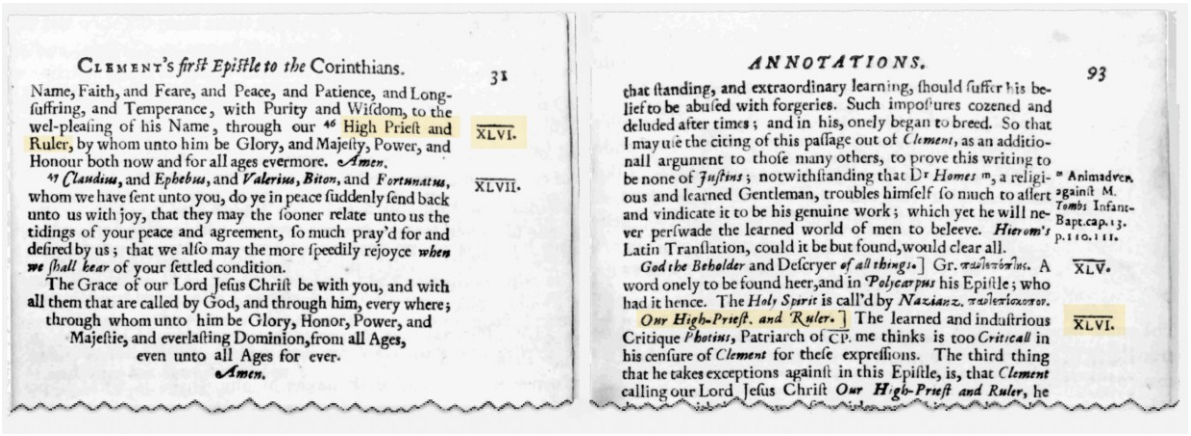


Abb. 2: Drei Knoten (bzw. Dokumente), die durch zwei Hyperlinks miteinander verknüpft sind. Der Leser kann vom Knoten A direkt zu B und C springen.

Die Idee, Informationen durch assoziative Referenzen einfacher zugänglich zu machen, ist bereits wesentlich älter – älter sogar als der Buchdruck. Schon eine Reihe historischer Dokumente besitzt Querverweise wie etwa der Talmud oder die Bibel. Diese Referenzierungssysteme wurden in der heutigen Form allerdings erst ab dem Mittelalter den Schriften hinzugefügt. Dies liegt unter anderem darin begründet, dass ungefähr bis zum fünften Jahrhundert nach Christus *Papyrusrollen* statt *Bücher* mit Seiten verwendet wurden, und danach nochmals viele Jahrhunderte vergingen, bis die ersten Bücher Seitenzahlen hatten und präzise Verweise innerhalb der Dokumente möglich wurden (Chartier & Cavallo 1999).

Ein frühes Beispiel für ein gedrucktes Buch mit einer ganzen Reihe von Hypertext-Merkmalen ist eine Ausgabe des Neuen Testaments von 1647 (DeRose & Durand 1994). Abb. 3 zeigt auf der linken Seite einen Ausschnitt des Briefes an die Korinther. Beispielsweise ist der Satzteil „High Priest and Ruler“ durch die seitliche Referenzangabe „XLVI“ mit einer Annotation verknüpft, die in der rechts abgebildeten Seite dargestellt wird. Hier findet man den referenzierten Text dupliziert wieder, abgeschlossen durch eine eckige Klammer und gefolgt von der eigentlichen Anmerkung. Weiter oben auf Seite 93 findet sich eine Anmerkung, die selbst einen Verweis enthält: Dem Namen „Dr. Homes“ folgt eine Art Fußnotenzeichen „m“, das sich auf den Text in der rechten Marginalspalte der Seite bezieht. Dabei wird ein weiterer Abschnitt der Bibel referenziert.

Ein moderneres Beispiel für gedruckte Bücher mit einem entsprechenden Referenzierungssystem und vielen Querverweisen sind *Enzyklopädien*. Allerdings spricht man bei solchen Druckwerken – auch wenn sie Referenzen aufweisen – in der Regel nicht von Hypertext, da es ihnen an einem weiteren charakteristischen Hypertext-Merkmal mangelt, nämlich inwieweit sie die Informationen für den Anwender in vereinfachter Weise zugänglich machen.



CLEMENT's *first Epiſtle to the* Corinthians. 31

Name, Faith, and Feare, and Peace, and Patience, and Long-ſuffring, and Temperance, with Purity and Wiſdom, to the wel-pleaſing of his Name, through our [46] High Prieſt and Ruler, by whom unto him be Glory, and Majeſty, Power, and Honour both now and for all ages evermore. *Amen.* XLVI.

[47] *Claudius*, and *Ephebus*, and *Valerius*, *Biton*, and *Fortunatus*, whom we have ſent unto you, do ye in peace ſuddenly ſend back unto us with joy, that they may the ſooner relate unto us the tidings of your peace and agreement, ſo much pray'd for and deſired by us; that we alſo may the more ſpeedily rejoyce *when we ſhall hear* of your ſettled condition. XLVII.

The Grace of our Lord Jeſus Chriſt be with you, and with all them that are called by God, and through him, every where; through whom unto him be Glory, Honor, Power, and Majeſtie, and everlaſting Dominion, from all Ages, even unto all Ages for ever. *Amen.*

ANNOTATIONS. 93

that ſtanding, and extraordinary learning, ſhould ſuffer his belief to be abuſed with forgeries. Such impoſtures cozened and deluded after times; and in his, onely began to breed. So that I may uſe the citing of this paſſage out of *Clement*, as an additionall argument to thoſe many others, to prove this writing to be none of *Juſtins*; notwithſtanding that Dr *Homes* [m], a religious and learned Gentleman, troubles himſelf ſo much to aſſert and vindicate it to be his genuine work; which yet he will never perſwade the learned world of men to beleeve. *Hierom's* Latin Tranſlation, could it be but found, would clear all.

[m] Animadver. againſt M. *Tombs* Infant-Bapt. cap. 13. p. 110. 111.

God the Beholder and Deſcryer *of all things.*] Gr. παντεπόπτης. A word onely to be found heer, and in *Polycarpus* his Epiſtle; who had it hence. The *Holy Spirit* is call'd by *Nazianz.* παντεποπτικον. XLV.

Our High-Prieſt, and Ruler.] The learned and induſtrious Critique *Photius*, Patriarch of CP. me thinks is too *Critical* in his cenſure of *Clement* for theſe expreſſions. The third thing that he takes exceptions againſt in this Epiſtle, is, that *Clement* calling our Lord Jeſus Chriſt *Our High-Prieſt and Ruler*, he XLVI.

Abb. 3: Links ein Ausschnitt aus dem Brief an die Korinther, rechts die „Certaine Annotations" (modifiziert nach DeRose & Durand 1994).

Für Nielsen ist das maßgebliche Kriterium, wie sich ein Hypertext-System dem Benutzer präsentiert. Nach seiner Definition soll Hypertext dem Benutzer das Gefühl geben, sich *frei* zwischen Informationen bewegen zu können:

> True hypertext should also make users feel that they can move freely through the information according to their own needs. […] This feeling is hard to define precisely but certainly implies short response times and low cognitive load when navigating (Nielsen 1993a).

Hypertext muss demnach eine *schnelle* Navigation mit *geringer kognitiver* Last zwischen den Einträgen ermöglichen, mittels derer die Informationsaufnahme und das Denken für den Menschen unterstützt werden.

Conklin sah in diesem Sinne *„die Navigationsunterstützung durch die Maschine"*[2] als eine grundlegende Eigenschaft von Hypertext an (Conklin 1987), da seines Erachtens nur Computer den Zugang zu den Informationen in der gewünschten Art weiter beschleunigen könnten. Später wurden allerdings einige Konzepte für gedruckten und geschriebenen Text aufgrund ihrer *Ausrichtung* auch dem Wissenschaftsgebiet Hypertext zugerechnet, auch wenn keine Automaten im Spiel waren.

Ein recht bekanntes Beispiel für nichtdigitalen Hypertext sind „Luhmanns Zettelkästen". Der Systemtheoretiker Niklas Luhmann (*1927; †1998) verwendete über Jahrzehnte mit Zetteln gefüllte Holzkästen, um sein Wissen zu katalogisieren und zu strukturieren. Jeder der über 35000 Zettel enthielt Informationen zu einem Thema und trug eine eindeutige Nummer, wodurch *„Verweisungen"* zwischen den Zetteln möglich wurden und von denen er ausgiebig Gebrauch machte. Als Ergebnis dieser Arbeit entstand über die Jahre laut Luhmann *„eine Art Zweitgedächtnis, ein Alter ego, mit dem man laufend kommunizieren kann"* (Luhmann 1992: 57). Er betrachtete seine Zettelkästen als essenzielle Hilfe zum Ordnen von Gedanken und zum vereinfachten Zugriff auf sein Wissen.[3]

---

[2] Conklin schreibt: "The most distinguishing characteristic of hypertext is its machine support for the tracing of references." (Conklin 1987, S. 33).

[3] In der Dokumentation „Beobachter im Krähennest" erklärt Niklas Luhmann anschaulich die Funktionsweise seiner Zettelkästen (Strauch & Luhmann 1989), siehe auch: http://www.youtube.com/watch?v=7gxXkbEag6k

Eine weitere Art von Hypertext ohne maschinelle Navigationsunterstützung wurde von (Wan, Robertsin et al. 1999) vorgeschlagen. Die Autoren realisierten Hyperlinks in gedruckten Dokumenten mithilfe physikalischer Mittel: Kerben im Rand der Seiten eines Buches (wie man sie teilweise auch in Lexika oder Kinderbüchern kennt) sollten beim Blättern zwischen unterschiedlichen Seiten helfen und es erlauben, im Buch hin und her zu springen und so direkt auf den referenzierten Textabschnitt zuzugreifen (s. Abb. 4).

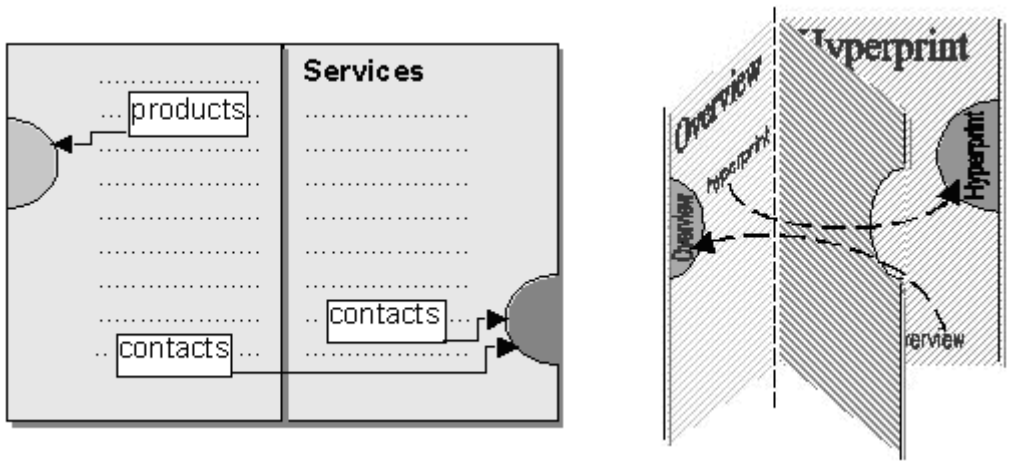


Abb. 4: Bei Wan et al. sorgen Kerben in den Seiten für die schnelle Navigation (nach Wan, Robertsin et al. 1999).

Trotz dieser Gegenbeispiele versteht man in der Regel unter einem Hypertext-System ein computerbasiertes Informationssystem mit einer Schnittstelle, die – neben anderen konventionellen Recherchemechanismen – zusätzlich eine einfache und flexible Navigation entlang von semantischen Querverweisen – den Hyperlinks – erlaubt. Diese Definition ist im Rahmen dieser Arbeit maßgeblich.

Teilweise wird alternativ der Begriff *Hypermedia* als die Verschmelzung von *Hyper*text und Multi*media* gebraucht, um die multimedialen Aspekte eines Projektes hervorzuheben (Garrett, Smith et al. 1986; Nielsen 1993a: 5; Hardman, Bulterman et al. 1994). Zumeist werden heute beide Begriffe aber synonym genutzt, da fast alle jüngeren Systeme multimediale Aspekte aufweisen (Nielsen 1995b; Kuhlen 1997). In dieser Dissertation soll an dem geläufigeren Begriff Hypertext festgehalten werden, zumal viele multimediale Problemfelder, wie die Integration von Animationen oder Audio-Daten, nicht im Mittelpunkt stehen und nur am Rande behandelt werden (s. Kapitel 5.2.9, Anhang C.1.8 und Anhang C.2).

2.1.5 Anwendungsfelder von Hypertext

Die Anwendungsmöglichkeiten von Hypertext-Systemen sind ähnlich weit gestreut wie die für Informationssysteme im Allgemeinen, da im Prinzip jedes Informationssystem um Hypertext-Funktionalität erweitert werden kann (s. Abschnitt 2.2.2.2). Im Folgenden wird ein Überblick über bedeutsame Anwendungsbereiche für Hypertext gegeben und jeweils kurz beleuchtet, welche besondere Rolle den Hyperlinks dabei zukommt, um einen Einblick in die Vielfältigkeit des Konzepts zu geben.

Ein Feld charakteristischer Hypertext-Anwendungen sind digitale Enzyklopädien (z. B. WikiPedia[4]), Online-Handbücher und Hilfesysteme (wie das Windows-Hilfesystem). Dabei verweisen die Hyperlinks jeweils auf semantisch verwandte Dokumente. Anstatt die Suchfunktion oder ein Inhaltsverzeichnis zu bemühen, kann der Benutzer durch das Auswählen entsprechender Links direkt auf weitere erforderliche Informationen zugreifen.

Ein weiteres bedeutendes Einsatzgebiet für Hypertext sind Schulung und Lehre. Hypertext-Systeme lassen sich zum einen für Online-Kurse einsetzen, wobei die assoziativen Links insbesondere das explorative Lernen unterstützen und den Leser zum weiteren Erkunden des Systems motivieren. Der Lernende wird dabei häufig zu unerwarteten und dennoch interessanten Informationen geleitet (s. Abschnitt 3.2.1; Kuhlen 1991). Darüber hinaus lassen sich kollaborative Hypertext-Systeme für eine interaktive Lehr- und Lernunterstützung und die Kommunikation in Lerngruppen einsetzen. Die Studierenden werden dabei selbst zum Autor und können Dokumente, Verweise und Annotationen hinzufügen, wie beispielsweise im webbasierten CommSy.[5] In mehreren Studien haben Studierende in Kursen mit interaktiven Hypertexten, die sie selbst mitgestalten konnten, effizienter und motivierter mitgearbeitet als in vergleichbaren Veranstaltungen ohne Hypertext-Unterstützung (Beeman, Anderson et al. 1987; Rada, Ramsey et al. 1993; Gillies & Cailliau 2001; Finck, Janneck et al. 2005).

Heutzutage nutzen auch Systeme aus dem *E-Commerce-Bereich* mit großer Selbstverständlichkeit Hypertext-Funktionalität, um die angebotenen Waren miteinander zu verknüpfen. Der Kunde wird mithilfe der Hyperlinks auf weitere, für ihn potenziell interessante Artikel hingewiesen, oder ihm werden Alternativen zum ausgewählten Produkt näher gebracht (Hansen, Yndigegn et al. 1999). Ein klassisches Beispiel ist der Online-Händler Amazon, der dem Benutzer auf jeder Produktseite aufgrund seines Profils und dem Verhalten anderer Kunden weitere Angebote darstellt.

In anderen Anwendungsfeldern von Hypertext wird nicht primär versucht, den Zugriff auf Informationen oder Angebote effizienter zu gestalten, sondern es wird die Vernetzung der Dokumente für andere Zwecke eingesetzt. Beispielsweise bricht *literarischer Hypertext* mit traditionellen Lesegewohnheiten und definiert mithilfe der Hyperlinks das Erlebnis des Lesens neu. Der zusätzliche Freiheitsgrad soll es dem Leser ermöglichen, das Werk auf individuelle Weise zu erfahren, indem ihm bei jedem Knoten unterschiedliche Pfade zum Weiterlesen angeboten werden. Dabei wird bewusst ausgenutzt, dass die Wahrnehmung der verknüpften Dokumente durch die Hyperlinks verändert werden kann, indem der Leser eine – möglicherweise vom Autor intentional gewählte – gedankliche Assoziation vornimmt (Burbules 1997). Zwei entsprechende Werke sind Stuart Moulthrops „Victory Garden“

[4] Die populäre webbasierte Enzyklopädie WikiPedia zeichnet sich dadurch aus, dass jeder Benutzer auch als Autor aktiv werden und Einträge ergänzen und überarbeiten kann: http://www.wikipedia.org/.

[5] Das System CommSy wurde an der Universität Hamburg entwickelt: http://www.commsy.net/.

(Moulthrop 1991) und „Sister Stories“ von Rosemary Joyce, Carolyn Guyer und Michael Joyce, das auch im Web zugänglich ist (Joyce, Guyer et al. 2000).

Ein anderes Beispiel für eine weitergehende Verwendung von Hypertext ist die als semantisches Netz zur *Modellierung* des Gedächtnisses. Rao und Turoff haben ein Hypertext-Framework namens „Guilford's ‚Structure of Intellect' Model“ entwickelt, das Knoten zur Repräsentation von Gedanken und Ideen verwendet, wogegen Links die Beziehungen zwischen diesen Ideen herstellen. Das Framework definiert eine Reihe von Knoten- und Link-Typen, die es erlauben sollen, alle menschlichen intellektuellen Möglichkeiten zu repräsentieren (Rao & Turoff 1990).

Dieser kurze Einblick zeigt bereits, dass das Forschungsgebiet Hypertext weitverzweigt ist und es vielfältige Einsatzgebiete der zugrunde liegenden Konzepte gibt. Zur Eingrenzung des Themas wird bei dieser Arbeit der Fokus auf Hypertext-Informationssysteme gelegt, die die Verknüpfungen dazu nutzen, den Umgang mit den Informationen effizienter und menschengerechter zu gestalten. Wie die ersten drei Beispiele dieser Übersicht zeigen, sind aber selbst bei dieser Einschränkung vielfältige Anwendungsgebiete von Hypertext mit unterschiedlichen Ansätzen zu finden. Im folgenden Abschnitt 2.2 werden die unterschiedlichen Hypertext-Paradigmen abgegrenzt und die Konzepte und Begriffe von assoziativem Hypertext genauer spezifiziert.

2.2 Hypertext-Informationssysteme: Paradigmen und Begriffe

Anhand der Methode, mit der ein Hypertext-System die Beziehungen zwischen den Objekten herstellt, lassen sich drei grundlegende Hypertext-Paradigmen charakterisieren: *assoziativer*, *räumlicher* und *taxonomischer Hypertext*. Assoziativer Hypertext zeichnet sich durch die Möglichkeit aus, die Dokumente durch explizite Hyperlinks miteinander zu verknüpfen. Dieses *Paradigma* steht im Blickfeld dieser Arbeit, und seine Konzepte und Begriffe werden im nächsten Abschnitt 2.2.1 dargestellt. Räumlicher und taxonomischer Hypertext als eigenständige Systemkonzepte sind in den letzten Jahren in den Hintergrund getreten, spielen aber in Kombination mit dem assoziativen Hypertext weiterhin eine Rolle.

Räumlicher Hypertext (Spatial Hypertext) hat seine Wurzeln in assoziativen Hypertext-Systemen, die neben der Link-Navigation auch die Navigation mittels einer grafischen Kartenansicht des *Hyperspace* – der Menge von Knoten und Links des Hypertextes – erlauben (s. Abschnitt 3.1.4; Marshall & Shipman III 1995). Beim räumlichen Hypertext dient die Karte als *primäres* Navigations- und Manipulationswerkzeug: Anstatt die Objekte explizit über Links zu verknüpfen, nutzt man räumlich-visuelle Struktur- und Mengenkonzepte. Alle Objekte werden hierzu auf der Karte als grafische Elemente – beispielsweise Kästen – repräsentiert (s. Abb. 5). Die Positionierung und Gruppierung dieser Elemente gibt ihre semantische Relation zueinander wieder. Die Fokussierung auf die Karte im räumlichen Hypertext soll es dem Anwender erleichtern, auf interaktive Weise Ideen und Strukturen zu entwickeln und dabei gleichzeitig den Überblick zu bewahren (Shipman III & Marshall 1999). Weg-

weisende Systeme für das räumliche Hypertext-Konzept waren *Aquanet* (Marshall & Rogers 1992), *VIKI* (s. Abb. 5; Marshall, III et al. 1994), SEPIA und COWFISH (Wang & Haake 1997). An diese Konzepte angelehnte Werkzeuge werden bis heute zur Strukturierung von Informationen und Ideen verwendet.[6]

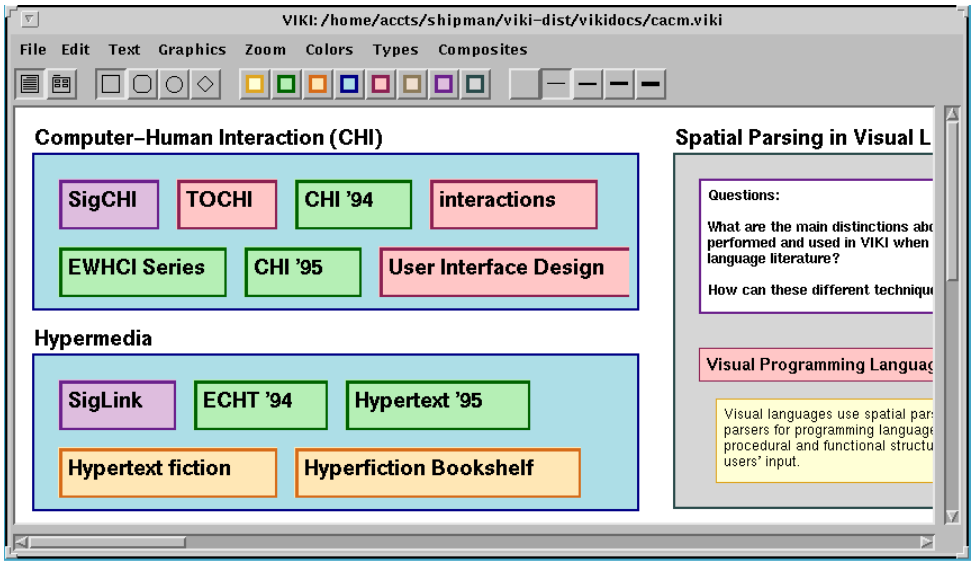


Abb. 5: Bildschirmfoto des VIKI-Systems (nach Marshall & Shipman III 1995). Die Kästen repräsentieren die Knoten des Hypertextes; Attribute wie Anordnung, Schachtelung und Färbung drücken ihre Relation zueinander aus.

Beim *taxonomischen Hypertext* (*Taxonomic Hypertext*) werden die Knoten ebenso wie beim räumlichen Hypertext nicht explizit miteinander verknüpft. Stattdessen wird jeder Knoten einer oder mehreren Mengen zugeordnet, die jeweils ein bestimmtes Thema, Konzept oder Attribut repräsentieren. Diese Mengen stellen Beziehungen zwischen den Knoten her, indem die gemeinsame Zugehörigkeit zweier Objekte zu den gleichen Mengen ihre semantische Nähe ausdrückt (Parunak 1991). Zur Navigation werden statt expliziter Verknüpfungen mengenorientierte Algorithmen verwendet, die aufgrund der Zuordnung der Objekte zu den Mengen automatisch ähnliche Objekte ermitteln. Zwei charakteristische Vertreter dieses Paradigmas sind das System *HyperSets* (Parunak 1991) und *StrathTutor* (Kibby & Mayes 1989). Eine Erweiterung des Konzepts sieht vor, interessante Textabschnitte in den Dokumenten (sogenannte *Hotspots*) zusätzlich weiteren thematischen Mengen zuzuordnen, um sie zu konkretisieren (Kibby & Mayes 1989). Die Hotspots dienen als Ausgangspunkt zu anderen Dokumenten, indem für den Leser bei Anwahl eines Hotspots aus der Kombination aller zugeordneten Mengen automatisch inhaltlich passende Dokumente berechnet werden.

Anwendungsfelder für taxonomischen Hypertext sind zum einen *homogene Informationsmengen*, bei denen die große Ähnlichkeit der Objekte miteinander es schwer macht, sie hierarchisch in Kategorien zu organisieren. Taxonomischer Hypertext bietet dabei den Vorteil, dass die Mengen zur Kategorisierung noch problemlos während der Arbeit mit dem

[6] Ein Beispiel ist das Tinderbox-System von Eastgate Systems, das zusätzlich die Konzepte von Mind Maps (Buzan & Buzan 2002) integriert. Siehe: http://www.eastgate.com/Tinderbox/

System ergänzt werden können. Ein anderes Potenzial sind zudem Informationssysteme mit *großen Objektmengen*, bei denen es für die Autoren aufgrund der Objektzahl unmöglich ist, alle geeigneten Link-Ziele zu ermitteln (Kibby & Mayes 1989). Die Stärken der Prinzipien von taxonomischem Hypertext zeigen sich inzwischen (in Ansätzen) auch im Web: Beim sogenannten *Tagging* klassifizieren Benutzer Web-Ressourcen mit Stichworten und ordnen sie dabei (wie beim taxonomischen Hypertext) bestimmten Mengen zu, innerhalb derer die Leser navigieren können (s. Abschnitt 3.1.6).

2.2.1 Das assoziative Hypertext-Paradigma: Knoten und Links

Assoziativer Hypertext kann als das „klassische" Hypertext-Paradigma angesehen werden. Es folgt den Ideen von Bush (s. Anhang A.5), der mittels *expliziter Pfade* zusammengehörige Dokumente *assoziativ* miteinander verknüpfen wollte (Bush 1945a). Die wichtigsten Konzepte sind die *Knoten* und die *Links*, wobei jeder Link zwei oder mehr Knoten in eine bestimmte inhaltliche Relation zueinander setzt. Man spricht daher auch von *Knoten-Link-Hypertext* („*Node-Link-Hypertext*", siehe: Conklin 1987).

Die besondere Rolle der Links im assoziativen Hypertext ergibt sich daraus, dass sie selbst Nachrichtenträger sind, weil sie von Hypertext-Autoren *explizit* und mit einer bestimmten *Intention* erstellt werden. Diese Information drückt sich dadurch aus, in welcher Art ein Link zwei Knoten verknüpft (s. u.), welche Objekte zueinander in Beziehung gesetzt werden und wie der Verweis typisiert wird (s. auch Abschnitt 4.1.2). Gleichzeitig kann ein Link die Bedeutung der verknüpften Knoten für den Benutzer beeinflussen, da er die Dokumente in einen bestimmten Kontext setzt. Beispielsweise lässt ein Verweis vom Drogenmissbrauch Jugendlicher zur Geschichte der Rockmusik, diese in einem potenziell negativen Licht erscheinen (Burbules 1997). Viele Hypertext-Systeme[7] (s. Anhang B) und alle Hypertext-Standards (s. Anhang C) tragen der Bedeutung von Hyperlinks dadurch Rechnung, dass sie diese – ebenso wie die Knoten – als eigene Objekte realisieren und somit als „*First-Class Citizens*"[8] behandeln.

Eine genauere Analyse zeigt, dass sich die Realisierung der Hyperlinks in den einzelnen Systemen zum Teil gravierend unterscheidet. Dies bezieht sich auf mehrere Aspekte der Verknüpfungen:

- Zum einen unterscheiden sich Links bezüglich ihrer *Funktion*. Links können nicht nur zur Navigation zwischen inhaltlich zusammengehörigen Objekten verwendet werden, sie lassen sich beispielsweise auch einsetzen, um mehrere Knoten zu einem Dokument zu-

[7] Beim World Wide Web sind Links keine eigenen Objekte, sondern in die Knoten eingebettet. Das Web stellt damit eine der wenigen Ausnahmen unter den in den 1990er Jahren entwickelten Hypertext-Systemen dar (s. Anhang B und Tabelle 5).

[8] Diese Formulierung für die Repräsentation von Links und Pfaden als eigene Objekte wurde von Polle Zellweger geprägt (Zellweger 1989). Sie drückt damit aus, dass Links und Pfade als gleichwertig mit den Dokumenten anzusehen sind.

sammenzufügen (Halasz 1988),[9] die Knoten eines Systems hierarchisch zu strukturieren (*strukturelle Links*) oder um andere Dienste und Programme anzusteuern (*programmatische Links*). Abschnitt 4.4.2 geht detailliert auf die unterschiedlichen Kategorien von Link-Aktionen und Funktionen ein.

- Bezüglich der *Direktionalität* sind *unidirektionale* von *bidirektionalen* Links zu unterscheiden. Bidirektionale Links lassen sich im Gegensatz zu unidirektionalen in beide Richtungen verfolgen, folglich sind vom Zielobjekt aus alle hierauf verweisenden Links zugänglich. Bidirektionale Links können zudem gerichtet sein, um den vom Autor intendierten Navigationsweg und die Relation der Knoten zueinander auszudrücken (s. Abb. 7).
- Der *Grad* eines Hyperlinks gibt die Anzahl der möglichen Knoten an, die er miteinander verknüpft. Viele Hypertext-Systeme unterstützen lediglich *binäre Links* mit genau zwei Endpunkten, andere Hypertext-Systeme bieten darüber hinaus *multiple Links* (auch *multivalente Links* genannt (Bernstein, Bolter et al. 1991), die sich auf mehr als zwei Objekte beziehen (s. Abb. 8). Es gibt zahlreiche Anwendungsfelder, beispielsweise kann mit einem multiplen Link auf mehrere Dokumente unterschiedlichen Detailgrades verwiesen werden (Bernard, Crowder et al. 1995) – bei der Auswahl des Links wird dem Benutzer eine Liste der Zielobjekte angeboten (Landow & Kahn 1992), oder das System bestimmt aufgrund des Benutzerprofils das geeignete Link-Ziel automatisch (Miller & Wantz: 1998). Multiple Links können aber auch verwendet werden, um *Pfade* zwischen mehreren Knoten zu definieren oder mehrere Dokumente gleichzeitig zu öffnen („*Fat Links*“, siehe: Nielsen 1993a: 115).
- Es wurden unterschiedliche Konzepte zur *Typisierung* von Links entwickelt. Der Typ eines Links spezifiziert die Relation, in der er die beteiligten Knoten setzt. Link-Typen dienen beispielsweise dazu, dem Leser die Bedeutung eines Links zu vermitteln (s. Abb. 9). Manche Systeme bieten dem Autor eine Auswahl vordefinierter Link-Typen an (*strenge Typisierung*), andere Systeme erlauben es, Links mit beliebigen Attributen oder Namen zu versehen (*freie Typisierung*). In Kapitel 4 wird auf die Potenziale und Grenzen der Link-Typisierung eingegangen, Kapitel 5.3 klassifiziert die Konzepte zur Visualisierung von Typ-Informationen.
- Hyperlinks können von unterschiedlicher *Granularität* sein. Entweder beschränken sie sich darauf, ganze Knoten miteinander zu verknüpfen, oder sie sind in der Lage, Positionen oder Regionen in Dokumenten miteinander in Beziehung zu setzen. Zur Spezifikation dieser Bereiche werden *Link-Anker* verwendet, die je nach Hypertext-System innerhalb des Knotens, des Link-Objektes oder als eigenes Anker-Objekt definiert sein können (s. *Abb. 10*). Einige Systeme bieten zudem die Möglichkeit, neben dem Link-Anker einen *Link-Kontext* anzugeben (auch „*Link Extent*“ genannt, vergleiche Anhang B.10), der

[9] Dieses Konzept ist auch als *Transklusionen* bekannt geworden (Nelson 1999). Siehe auch Kapitel 4.4.8, Anhang A.3 und Anhang B.5.

den Umfang eines erweiterten Textbereiches definiert und zur Verständlichkeit des Links beitragen soll (Nanard & Nanard 1993; Hardman, Bulterman et al. 1994; Weinreich, Obendorf et al. 2001).

Diese vielfältigen Verknüpfungsmöglichkeiten haben Auswirkungen auf die Benutzungsschnittstelle der Hyperlinks. Beispielsweise müssen feingranulare Link-Anker, die sich auf Regionen eines Knotens beziehen, mittels des *Link-Markers* erkenntlich gemacht werden, sodass der Anwender sie auswählen kann: Der Link Marker ist die sinnlich (meist visuell) wahrnehmbare Hervorhebung eines Link-Ankers (s. Abb. 10). Er *kann* sowohl den Start als auch das Ziel eines Links kennzeichnen. Darüber hinaus sollen häufig Informationen über den *Typ*, zur *Richtung* und zum *Ziel* des Links (die Vorschau auf das Zielobjekt heißt auch *Link-Preview*, siehe Anhang B.10) beim Link-Marker dargestellt werden. Eine Analyse der wichtigsten Konzepte für die Benutzungsschnittstelle von Links und Link-Markern ist in den Kapiteln 5.1 bis 5.3 zu finden.

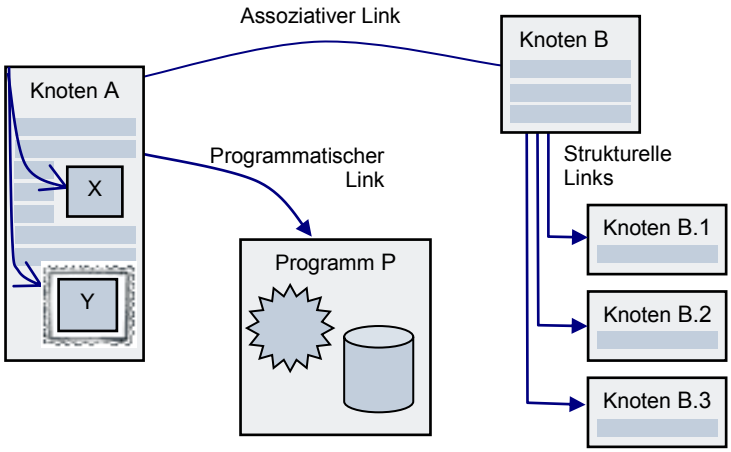


Abb. 6: Schematische Darstellung unterschiedlicher Link-Eigenschaften: Knoten A bindet Objekte X und Y mittels Transklusionen ein. Ein assoziativer Link setzt Knoten A und Knoten B in inhaltliche Beziehung. Ein programmatischer Link startet oder steuert das Programm P. Von Knoten B aus führen strukturelle Links zu den Unterknoten B.1 bis B.3.

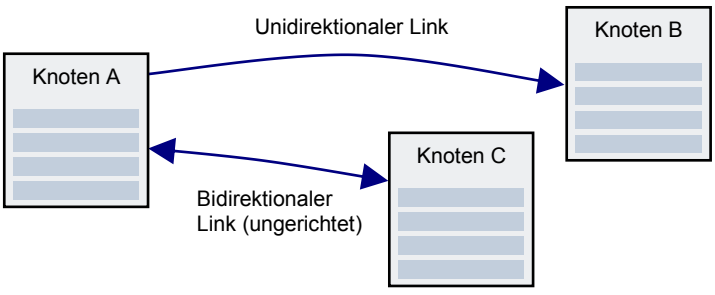


Abb. 7 Zwei Links unterschiedlicher Direktionalität.

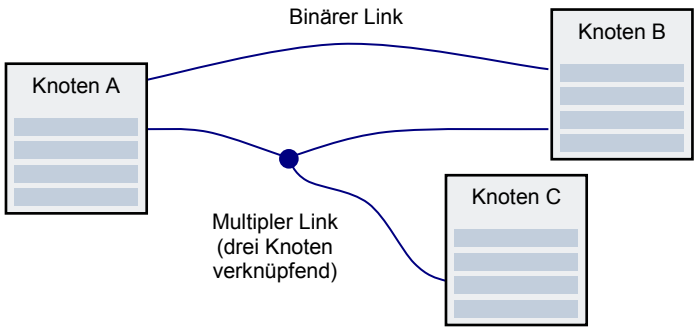


Abb. 8: Zwei Links unterschiedlichen Grades.

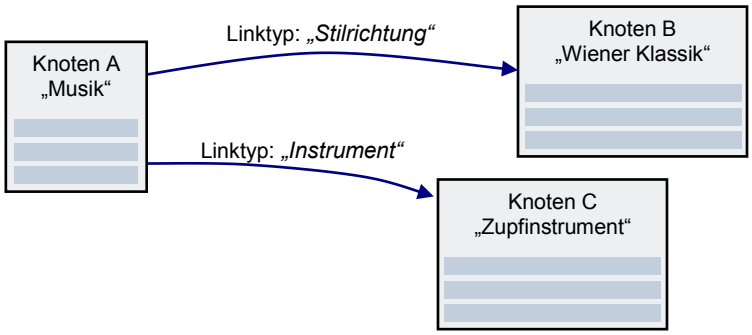


Abb. 9: Typisierte Links spezifizieren die Relation zwischen Link-Start (Knoten A) und Ziel (Knoten B & C).

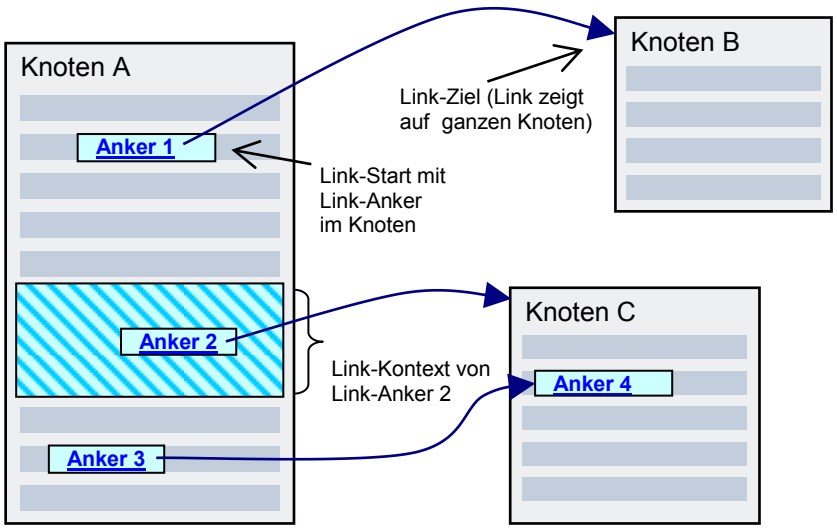


Abb. 10: Knoten A hat drei Link-Anker. Die ersten beiden verweisen jeweils auf die ganzen Knoten B und C, der dritte Anker bezieht sich auf Anker 4 innerhalb von Knoten C. Anker 2 hat zusätzlich einen Link-Kontext, der die Bedeutung des Links genauer spezifiziert. Als Link-Marker dienen in dieser Illustration hellblaue Kästchen mit dunkelblauem Rahmen und blauem, unterstrichenem Text.

Bezüglich der Repräsentation der *Knoten* eines Hypertext-Systems lassen sich zwei wesentliche Ansätze unterscheiden (Neumüller 2001): das der *Karte (Card)* und das der *Schriftrolle (Scroll)*.

Beim *kartenbasierten Hypertext* sind die einzelnen Knoten maximal so groß wie der Bildschirm. Jede Karte hat ein festes Layout; sie wird immer komplett angezeigt, und Links beziehen sich zumeist auf ganze Karten. Objekte, die nicht auf eine Karte passen, werden aufgeteilt, und der Benutzer muss zwischen den Karten wie bei Buchseiten blättern (s. auch

Tabelle 5). Typische Vertreter dieses Genre waren *ZOG/KMS* (vergl. Anhang B.4; Trigg & Weiser 1986), *HyperTies* (s. Anhang B.8; Shneiderman & Kearsley 1989) und *NoteCards* (s. Anhang B.9; Halasz, Moran et al. 1987).

Beim Konzept der Schriftrolle gibt es kein festes Layout der Dokumente. Sie werden stattdessen in Fenstern variabler Größe dargestellt, und der Benutzer muss – wie bei antiken Schriftrollen – zusätzlich innerhalb der Seiten navigieren, indem er den sichtbaren Bereich des Bildschirmfensters verschiebt („scrollt"). Da diese Dokumente beliebig lang sein können, unterstützen solche Systeme zumeist auch die Adressierung von Link-Zielen innerhalb der Knoten (s. o. Granularität der Links, Abb. 10). Das Konzept der Schriftrolle ist flexibler als das der Karte, weist aber den Nachteil auf, dass Navigation und Orientierung innerhalb der Seiten dem Leser zusätzliche Herausforderungen stellen. Beispielsweise kann der Benutzer in der Regel Titel und Anbieter einer Seite nicht mehr sehen, wenn er auf ihr nach unten gescrollt hat, wodurch seine Orientierung erschwert wird. Zudem kann die Adressierung der Link-Anker in den Knoten Probleme bereiten, sofern die Dokumente nachträglich bearbeitet werden. Beispiele für Vertreter dieses Konzepts sind das *Guide*-System (s. Anhang B.7; Brown 1987), *Intermedia* (s. Anhang B.10; Haan, Kahn et al. 1992) und das *World Wide Web* (s. Anhang B.15).

Diese Vielgestaltigkeit von Knoten und Links spiegelt sich in der großen Vielfalt unterschiedlicher Hypertext-Systeme wider (s. Anhang B und Tabelle 5). Sie werden alle dem assoziativen Hypertext-Paradigma zugerechnet, sofern sie Knoten als Informationseinheiten aufweisen, die mittels expliziter Links miteinander verknüpft werden können. Zudem müssen sie über eine entsprechende Benutzungsschnittstelle verfügen, die die einfache, direkte Navigation zwischen den Informationseinheiten entlang der Links erlaubt (Nielsen 1993a).

2.2.2 Die Konzepte offener, verteilter Hypertext-Informationssysteme

Die bisher entwickelten Hypertext-Systeme lassen sich sowohl konzeptionell als auch softwarearchitektonisch unterschiedlichen Generationen zuordnen. Zwei Systemgenerationen gingen den *offenen, verteilten Hypertext-Systemen* voraus, die seit den 1990er Jahren die Forschung und Entwicklung bestimmen.

Die Hypertext-Systeme der **ersten Generation** wurden in den 1960er und 1970er Jahren entwickelt. Sie hatten eine *monolithische* Systemarchitektur – Datenhaltung, Programmlogik und Benutzungsschnittstelle waren nicht getrennt – und sie liefen auf Großrechnern mit Textterminals und speziell für sie konstruierter Hardware. Bei der Entwicklung standen, neben der Unterstützung geistiger Arbeit, primär der Umgang mit Dokumenten, ihre kollaborative Bearbeitung sowie die Kooperationsunterstützung im Blickfeld (Conklin 1987). Die Projektentwickler versuchten sich dabei konzeptionell nicht durch die verfügbare Hardware oder das Betriebssystem einschränken zu lassen, sondern die weitreichenden Visionen der geisti-

gen Väter der Systeme umzusetzen. Ihre Arbeiten sind trotz des Alters aufgrund der zum Teil grundlegenden Erkenntnisse durchaus bis heute berücksichtigenswert (s. Anhang B.2ff).

Eine **zweite Generation** von Hypertext-Systemen bestimmte die Forschung und Entwicklung der 1980er Jahre. Die zu dieser Zeit aufkommenden Einzelplatzrechner (wie PCs und Heimcomputer) und verhältnismäßig leistungsstarken grafischen Workstations erlaubten die Realisierung lokaler spezialisierter Systeme mit mehrschichtiger Architektur und grafischer Benutzungsoberfläche (Halasz 1988). Die Hypertext-Systeme dieser Generation boten zumeist keine Mehrbenutzerunterstützung und konnten nur lokal durch einzelne Personen verwendet werden. Viele dieser Applikationen waren für ein spezielles Einsatzgebiet optimiert; sie dienten beispielsweise als Hilfesystem, Online-Manual oder Museumsführer. Wissenschaftlich wurde diese Periode durch das Aufkommen des Forschungsgebietes *Human Factors* entscheidend mitgeprägt, welches auf die Erforschung und Weiterentwicklung der damaligen Hypertext-Systeme ebenfalls maßgeblichen Einfluss hatte. Die zahlreichen zu dieser Zeit durchgeführten Benutzbarkeitsstudien der vielfältigen Systeme lieferten maßgebliche Resultate, die zum erheblichen Teil auch für heutige Entwicklungen noch interessant und relevant sind (s. Anhang B.9ff).

Die Entwicklung einer **dritten Generation** von Hypertext-Systemen begann ab dem Ende der 1980er Jahre (s. Anhang B.14ff). Sie zeichnet sich durch zwei charakteristische Bestrebungen aus: *Verteilung* und *Offenheit*; beide Konzepte werden im Folgenden charakterisiert.

2.2.2.1 Verteilter Hypertext

Das *grundlegende Ziel* eines verteilten Hypertext-Systems ist es, den Anwendern einen problemlosen Zugriff von *überall* auf *alle* Informationen und Dienste des Hypertext-Systems zu bieten. Gleichzeitig soll wieder die Kooperation und Kommunikation der Benutzer unterstützt werden, wie dies bereits bei den Systemen der ersten Generation im Blickfeld stand. Im Gegensatz zu früher werden aber nun nicht mehr Großrechner mit passiven Terminals eingesetzt, sondern es handelt sich um mehrschichtige Client-Server-Architekturen. Goose et al. (2000) definieren verteilte Hypertext-Systeme wie folgt (Goose, Hall et al. 2000):

1. Das System bietet den *verteilten Zugriff* auf die Dokumente sowie eine *gemeinsame Nutzung* aller enthaltenen Informationen.
2. Informationen können *verteilt organisiert* sein: Logische Strukturen erstrecken sich dabei über physikalisch verteilt angebotene Dokumente.
3. Das System unterstützt konfigurierbare *verteilte* Dienste: Es soll möglich sein, die Funktionalität der Hypertext-Umgebung durch die Integration und Konfiguration zusätzlicher Dienste an neue Anforderungen des Benutzers anzupassen.[10]

[10] Goose et al. dachten hierbei an Möglichkeiten zur Erweiterung von Hypertext-Frameworks wie ihres *Microcosm TNG* (Goose, Hall et al. 2000). Bezogen auf das Web ist die Integration neuer Dienste und Funktionen auf unterschiedliche Art gegeben: Serverseitig können nahezu beliebige neue Dienste angeboten werden, da

Hypertext erhält durch die Verteilung der Informationen und den verteilten Zugriff eine neue Qualität: Da es möglich wird, dass sehr viele Autoren Informationen anbieten, nimmt die Menge der *direkt* zugreifbaren Daten in gewaltigem Maße zu, ein Sachverhalt, der sich sehr gut im Web beobachten lässt (s. auch Kapitel 3.3.3). Für den Zugriff auf diese verteilten Informationen spielen Hyperlinks eine ganz entscheidende Rolle, die über ihre Bedeutung in lokalen Informationssystemen hinausgeht: Sogenannte *externe Links* vernetzen die Informationen unterschiedlicher Anbieter und erlauben den einfachen und schnellen Zugang zu den Angeboten vieler Verfasser (s. Abb. 11).

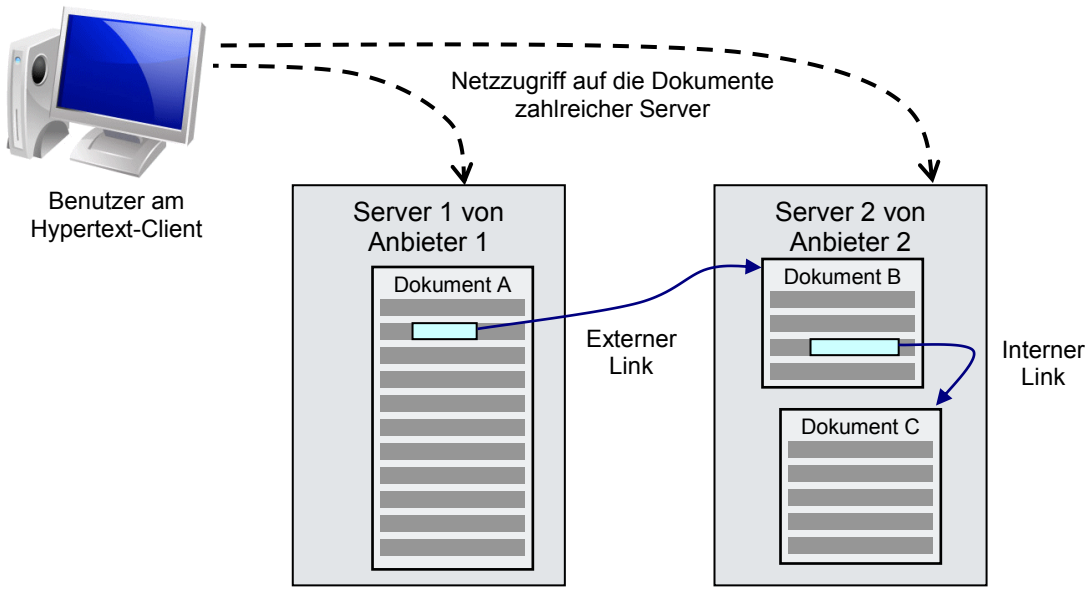


Abb. 11: Schematische Darstellung eines verteilten Hypertext-Informationssystems mit zwei Servern. Der Benutzer navigiert mithilfe der Links von Dokument A eines Anbieters zu den Dokumenten B und C eines zweiten Anbieters.

Neben ihren Potenzialen bergen verteilte Hypertext-Systeme eine ganze Reihe von neuen Herausforderungen in sich, die zwar bereits recht früh erkannt, aber bis heute nur teilweise praktikabel gelöst wurden (Kappe 1991). Entscheidende technische Probleme sind die *Gewährleistung der Datenkonsistenz* (s. auch Abschnitt 3.3.4; Pitkow & Jones 1996) und die *Skalierbarkeit* (Hall 1997). Anwender werden dabei mit neuen Benutzbarkeitsproblemen konfrontiert, wie der oft ungenügenden *Performanz* der Systeme (s. Abschnitt 3.3.1) und dem Phänomen des *Information Overload* (s. Abschnitt 3.3.2). Auf diese und weitere neue Schwierigkeiten bei der Benutzbarkeit verteilter Informationssysteme wird in Abschnitt 3.3 eingegangen.

2.2.2.2 Offener Hypertext

Das zweite Bestreben bei den Hypertext-Systemen der dritten Generation wird unter dem Begriff *offener Hypertext* zusammengefasst. Hierunter verstand man ursprünglich primär die Integrierbarkeit beliebiger Dokumenttypen und Anwendungen in eine Hypertext-Um-

---

Protokolle und Sprachen die interne Funktionalität der Web-Server kapseln. Clientseitig gelingt eine Erweiterung des Webs über die Browser-Schnittstellen für Plug-ins und Extensions (s. Kapitel 8.3ff).

gebung, die es so den Anwendern erlaubt, alle vorhandenen Materialen weiter zu verwenden und mit Links zu versehen (Pearl 1989). Später wurde die Definition für offenen Hypertext deutlich erweitert; nach Davis et al. muss ein offenes Hypertext-System die folgenden Eigenschaften erfüllen (Davis, Hall et al. 1992):

1. Das System setzt kein bestimmtes Dateiformat für die Dokumente voraus, welches andere Applikationen, die ebenfalls auf die Daten zugreifen, beeinträchtigen könnte.
2. Die Hypertext-Funktionalität kann in jedes Werkzeug integriert werden, das auf dem Betriebssystem des Computers läuft. Dateien, die mit anderen Programmen erstellt wurden, sollen sich mit dem Hypertext-System verwenden lassen, ohne dass sie angepasst oder verändert werden müssen.
3. Das System schafft keine künstliche Unterscheidung zwischen Lesern und Autoren.
4. Das System ist einfach erweiterbar und erlaubt es, neue Funktionalität hinzuzufügen.
5. Daten und Prozesse des Hypertext-Systems können verteilt über ein Netzwerk als auch auf unterschiedlichen Hardware-Plattformen laufen.

Diese Charakterisierung von offenem Hypertext scheint somit auch Eigenschaften von verteilten Hypertext-Systemen zu umfassen. Allerdings waren die offenen Hypertext-Systeme der damaligen Zeit, wie *Sun's Link Service* (s. Anhang B.12) oder *Microcosm* (s. Anhang B.13), zwar verteilte Client-Server-Anwendungen, aber aufgrund ihrer Architektur nur im lokalen Netz nutzbar, zumal sie aus technischen Gründen für den globalen Einsatz viel zu schlecht skalierten. Einen derartig umfassenden, globalen Einsatz ihrer offenen Hypertext-Systeme wie heute beim Web hatten Davis et al. vermutlich noch gar nicht im Sinn (s. Abschnitt 2.2.4).

Offener Hypertext ist ein Konzept, das den Umgang mit Applikationen und Informationen ganz signifikant erweitern und verändern kann: Durch die Integration in sämtliche Anwendungen wird Hypertext bei der Arbeit am Computer *allgegenwärtig*. Der Benutzer wird in die Lage versetzt, all seine Dokumente, Termine und Nachrichten miteinander zu vernetzen und so den Zugriff auf sie zu vereinfachen. Des Weiteren kann er ebenfalls die Werke anderer Autoren mit Links versehen und in seine Arbeitsumgebung integrieren (Davis 1995). Da diese Verknüpfungen auch auf selbst erstellte Texte verweisen können, lassen sich die Links ebenfalls zum Erstellen von *Anmerkungen* verwenden – ein offenes Hypertext-System integriert somit automatisch die Funktionalität zur *Annotation beliebiger Informationen* (Meyrowitz & Van Dam 1982a). Dies ist ein ausgesprochen bedeutsamer Zugewinn für die Arbeit am Computer, da Benutzern bis heute häufig keine Annotationsmöglichkeiten für die Dokumente anderer Autoren zur Verfügung stehen – dies wurde aber bereits vor Jahren als ein gravierendes Benutzbarkeitsdefizit für das Lesen von Dokumenten am Bildschirm ermittelt (siehe O'Hara & Sellen 1997; Price, Glovchinsky et al. 1998).

Aus technischer Sicht stellen offene Hypertext-Systeme zusätzliche Anforderungen. Damit sich beliebige existierende Objekte mit Verweisen versehen lassen, ist es notwendig, die Link-Strukturen *separat* von den Dokumenten zu speichern. Zudem müssen die strukturellen Informationen von beliebigen Anwendungen aus zugreifbar sein. Beides setzt einen entsprechenden *Link-Dienst* voraus, der die Strukturinformationen anwendungsübergreifend bereitstellt und den Zugriff auf die Link-Datenbank – die sogenannte *Linkbase* – steuert (s. auch Anhang B.13; Davis, Hall et al. 1992). Diese Trennung von Dokumenten und Links und die Offenheit gegenüber beliebigen Dokumenttypen sind mit einer Reihe konzeptioneller Herausforderungen verbunden (vergl. Pearl 1989; Davis, Hall et al. 1992; DeRose & Durand 1994):

1. Die meisten der vielfältigen existierenden Dateitypen benötigen eine spezifische Applikation zur Darstellung. Eine Integration der Hyperlink-Funktionalität in diese Programme setzt entweder voraus, dass sie entsprechende *Schnittstellen zur Erweiterung* zur Verfügung stellen oder dass ein Zugriff auf den *Quellcode* möglich ist. Oft ist keines von beidem gegeben.
2. Die Position und Größe der Link-Anker innerhalb der Dokumente muss je nach Dokumenttyp auf unterschiedliche Art und Weise definiert werden, um die Eigenschaften des Mediums zu berücksichtigen. Link-Anker in Texten benötigen beispielsweise andere Konzepte zur Beschreibung als Anker in Diagrammen, Audiodateien oder Animationen.[11]
3. Autoren können ihre Dokumente beliebig *bearbeiten* und *löschen*, auch wenn bereits Verweise von oder zu ihnen erstellt wurden. Dies kann dazu führen, dass für entsprechende Hyperlinks kein Zielobjekt mehr existiert oder die Platzierung von Link-Ankern innerhalb der geänderten Dokumente nicht mehr eindeutig möglich ist. Die inhaltliche Überarbeitung eines Dokuments kann zudem Verknüpfungen aus *semantischer* Sicht ungültig machen (s. auch Abschnitt 3.3.4).
4. Es gibt Probleme mit der *Versionierung* von Ressourcen: Führt ein Link zur alten, zur neuen oder zu beiden Versionen eines modifizierten Dokuments?
5. Die *Benutzungsschnittstelle* der Hyperlinks stellt ebenfalls neue Herausforderungen. Für die Hervorhebung der Link-Anker werden Interface-Techniken benötigt, die den Anker eindeutig kennzeichnen, ohne dass die Lesbarkeit der Dokumente eingeschränkt wird oder die Darstellung der Dokumente angepasst werden muss. Zudem ist es häufig notwendig, dass zusätzliche Informationen (wie der Typ des Links) beim Link-Marker erscheinen, um dem Benutzer Informationen über Funktion und Ziel des Links zu vermitteln.

Lösungsansätze für die meisten *technischen* Aspekte dieser Herausforderungen wurden bereits in früheren Hypertext-Systemen (s. Anhang B) und den offenen Hypertext-Standards

[11] Die Sprache HyTime setzt sich mit dieser Problematik auseinander (s. Anhang C.2).

(s. Anhang C) vorgestellt. Die Gestaltung der Benutzungsschnittstelle von Links für offene Hypertext-Systeme (Punkt 5) wurde hingegen in der Hypertext-Forschung zumeist nur peripher behandelt oder sogar ausgegrenzt (Weinreich, Obendorf et al. 2001; Obendorf & Weinreich 2003). Die vorliegende Arbeit beschäftigt sich daher systematisch mit der Anwendungsoberfläche von Links und insbesondere den Methoden zur Darstellung von Link-Markern (s. Abschnitt 5.2) und zusätzlichen Link-Informationen (s. Abschnitt 5.3); auf dieser Basis wurden neue, ergonomischere Konzepte für die Benutzungsschnittstelle von Hyperlinks in offenen, verteilten Hypertext-Systemen entwickelt (s. Abschnitt 5.5ff) und evaluiert (s. Abschnitt 6).

2.2.3 Vom offenen, verteilten Hypertext zum World Wide Web

Einige der bisher entwickelten Hypertext-Systeme lassen sich der dritten Hypertext-Generation zuordnen: Sowohl *Microcosm* (s. Anhang B.13), *Hyper-G* (s. Anhang B.14) als auch das *World Wide Web* (s. Anhang B.15) versuchen, neben Offenheit auch Verteilung zu realisieren; sie setzen allerdings unterschiedliche Schwerpunkte. Kritisch betrachtet erfüllt aber kein jemals entwickeltes System sämtliche Anforderungen an offenen, verteilten Hypertext, da entweder das Dateiformat durch das Hypertext-System vorgegeben ist oder die Systeme nicht ausreichend skalieren, um gruppenübergreifend einsetzbar zu sein.

Hyper-G kam wohl insgesamt dem Ziel eines offenen, verteilten Hypertext-Systems am nächsten, da es ein internetbasiertes Client-Server-System war, das den entfernten Zugang zu allen enthaltenen Informationen bot und (im Sinne von offenem Hypertext) Links von den Dokumenten getrennt speicherte (s. Anhang B.14). Allerdings boten die Anfang der 1990er Jahre verfügbaren *Hyper-G* Clients und Server beim überregionalen Zugriff auf die Dokumente wesentlich langsamere Zugriffszeiten als vergleichbare Web-Angebote (Weinreich 1997). Das World Wide Web – als gegenwärtig bedeutendstes Hypertext-System – erfüllt zwar die Kriterien für verteilten Hypertext, weist aber vor allem im Bereich der Offenheit klare Defizite auf: Webbrowser setzen bis heute (X)HTML als Auszeichnungssprache (*Markup Language*) voraus, und Hyperlinks sind in den Code der Dokumente eingebettet. Die unterschiedlichen Konzepte und Systeme zur Erweiterung des Webs zur Trennung von Link-Strukturen und Dokumenten blieben leider alle prototypisch.[12]

Auf der anderen Seite ist anzuerkennen, dass Tim Berners-Lee mit dem URI (*Uniform Resource Identifyer*) als Adressierungsschema (Berners-Lee, Fielding et al. 2005) und dem Übertragungsprotokoll HTTP („*Hypertext Transfer Protocol*“, siehe: Fielding, Gettys et al. 1999) zwei Mechanismen zur Verfügung gestellt hat, die unabhängig vom Dateityp sind und die die Einbindung fast beliebiger Ressourcen erlauben. Das einfache, verbindungslose Protokoll und die dezentrale Architektur des Webs gewährleisten zudem eine Skalierbarkeit,

[12] Beispiele hierfür sind der *Atlas Link Server* (Pitkow & Jones 1996), der *Distributed Link Service DLS* (s. Anhang B.13) als auch die später entwickelten komponentenbasierten Architekturen und Protokolle u. a. von der *Open Hypermedia Systems Working Group* (OHSWG)(Reich, Wiil et al. 1999; Holm 2002).

die den globalen Einsatz ermöglicht. Alle auf diese Weise zugreifbaren Ressourcen lassen sich auf beliebigen Systemen mit Internetzugang abrufen, sofern die entsprechende Applikation für das Dateiformat auf dem Computer verfügbar ist. Des Weiteren ist eine Integration von Hyperlinks, die auf Ressourcen im Web verweisen, auch in andere Dateiformate als HTML möglich, wie es *Macromedia Flash*, *Adobe Acrobat* und *Microsoft Word* demonstrieren; allerdings mussten die Programme und Datenformate entsprechend erweitert werden, und die Verweise sind ebenfalls (wie bei HTML) in die Dokumente eingebunden. Es ist zu hoffen, dass mit der sukzessiven Einführung von XML-Techniken und XLink (s. Anhang C.3) auch für das Web eine Trennung von Struktur und Inhalt im Sinne offener Hypertext-Systeme (s. Abschnitt 2.2.2.2) möglich wird (Obendorf 2001).

Obwohl das World Wide Web nur eingeschränkt als offenes Hypertext-System angesehen werden kann, weist es viele der Herausforderungen auf, die offener Hypertext an die Benutzungsschnittstelle von Hyperlinks stellt (s. Abschnitt 2.2.2.2): Zwar sind Link-Anker im Text von Webseiten zumeist noch als unterstrichener, farbiger Text zu identifizieren (s. Abschnitte 5.1.3 und 5.2), es fehlen aber für Verknüpfungen in Grafiken oder in Animationen Standards, und die Link-Marker werden von den Autoren nach Gutdünken gestaltet. Als Folge muss der Benutzer oft mit der Maus nach aktiven Bereichen in diesen Medien suchen, um die Links auszumachen. Sofern aus anderen Applikationen (beispielsweise einer Flash-Animation, einem E-Mail-Programm oder einer Office-Anwendung) auf eine Web-Ressource verwiesen wird, kann der Anwender häufig nur aufgrund seiner Erfahrung mit dem Programm erahnen, dass es sich bei einem hervorgehobenen Objekt *wahrscheinlich* um einen Link-Marker handelt. Bei einem offenen Hypertext-System (bei dem *jedes Objekt in jeder Anwendung* ein Hyperlink sein kann) wäre folglich ein systemübergreifender Interface-Standard für Link-Marker eine Voraussetzung für die einfache Identifikation von Links und eine gute Benutzbarkeit des Systems.

Da mit einem Web-Link (bzw. dem URI oder dem mit dem Link verknüpften Programm) nahezu beliebige Arten von Ressourcen adressierbar sind, kann es ebenfalls große Probleme bereiten, die Folgen beim Auswählen eines Links vorherzusehen: Verweist ein Link auf eine persönliche Datei, ein externes Dokument, eine Applikation oder handelt es sich möglicherweise um gar keinen Hyperlink? Entsprechende Konzepte und Standards für einheitliche, verständliche Hinweise zum Link-Ziel und zur Link-Funktion wären eine große Hilfe, um viele der mit Hyperlinks verbundenen Benutzbarkeitsprobleme zu reduzieren (s. Abschnitte 3.4 und 4.1).

2.2.4 Spezifische Potenziale und Herausforderungen des Webs

Verglichen mit den Hypertext-Systemen der 1990er Jahre weist das World Wide Web zahlreiche spezifische Eigenschaften und neue Möglichkeiten auf, und es stellt sich die Frage, inwieweit es noch ein offenes, verteiltes Hypertext-Informationssystem ist oder inzwischen einer eigenen Kategorie von Informationssystemen zugeordnet werden muss.

Im Folgenden werden die wesentlichen neuen Merkmale des World Wide Webs vorgestellt und diskutiert. Zu ihnen gehört, dass man inzwischen im Web einen fließenden Übergang zwischen *Navigation im Hypertext* und *Interaktion mit Applikationen* vorfindet, dass viele Ressourcen des Webs von *dynamischer Natur* sind und dass es sich in vielen Bereichen *vom Informations-* zum *Kommunikationsmedium* gewandelt hat, in dem die *soziale Navigation* zunehmend an Bedeutung gewinnt.

Hypertext-Navigation und Applikations-Interaktion im Web

Obgleich das Web ursprünglich als reines Hypertext-Informationssystem konzipiert wurde, dienen Webbrowser inzwischen regelmäßig zur Interaktion mit Online-Applikationen. Dieser Wandel lässt sich bereits anhand der Häufigkeit einzelner Benutzeraktionen mit dem Webbrowser identifizieren: So hat der Anteil von Aktivitäten, die mit dem Ausfüllen von Web-Formularen zusammenhängen, in den letzten Jahren deutlich zugenommen: von unter 5% bei einer Studie im Jahr 1996 auf über 14% im Jahr 2005 (Tabelle 1; Tauscher 1996; Weinreich, Obendorf et al. 2008). Die Nutzungshäufigkeit der Backtracking-Funktion (auch „Zurück-Button" genannt: Abschnitt 3.1.7) zum Wiederkehren zu einer kurz zuvor besuchten Seite hat gleichzeitig von über 30% auf unter 15% abgenommen (s. Tabelle 1). Hierfür gibt es zwei wesentliche Ursachen: Die Interaktion mit Online-Applikationen entspricht eher einem Workflow, bei dem seltener eine „Rückwärts-Navigation" benötigt wird als bei der Hypertext-Recherche, und viele Web-Applikationen unterstützen das Backtracking nicht im Sinne eines *Undo* der letzten Aktion oder geben sogar eine Fehlermeldung aus.[13]

Hyperlinks haben zwar ihre zentrale Bedeutung behalten und machen weiterhin über 40% der Benutzeraktionen aus (Tabelle 1 und Abb. 12 links), aber der Anteil von Links, die Parameter in dem URI an den Server übergeben (und somit zu einer dynamisch generierten Ressource verweisen), ist angewachsen und betrug bei den Teilnehmern der Studie von 2005 bereits über 40% (s. Abb. 12 rechts; Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Weinreich, Obendorf et al. 2008).

[13] Eine detailliertere Analsye der Ursachen und Folgen dieses Wandels im Web findet man in (Weinreich, Obendorf et al. 2006c; Obendorf, Weinreich et al. 2007).

Tabelle 1: Vergleich unterschiedlicher Langzeitstudien zur Navigation im Web.

	(Catledge & Pitkow 1995)[14]	(Tauscher & Greenberg 1997)	(Weinreich, Obendorf et al. 2006b)
Studienzeitraum	1994	1995-1996	2004-2005
Anzahl der Teilnehmer	107	23	25
Studiendauer (Tage)	21	35-42	52-195, ø=105
Anzahl der Aktionen	31.134	84.841	137.272
Links	45,7%	43,4%	43,5% ➔
Direkter Zugriff[15]	12,6%	13,2%	9,4% ↘
Neues Fenster	0,2%	0,8%	10,5% ↗
Formularaktionen	-	4,4%	15,3% ↗
Backtracking / Zurück	35,7%	31,7%	14,3% ↘
Neu Laden	4,3%	3,3%	1,7%
Vorwärts	1,5%	0,8%	0,6%
Andere Aktionen	-	2,3%	4,8%

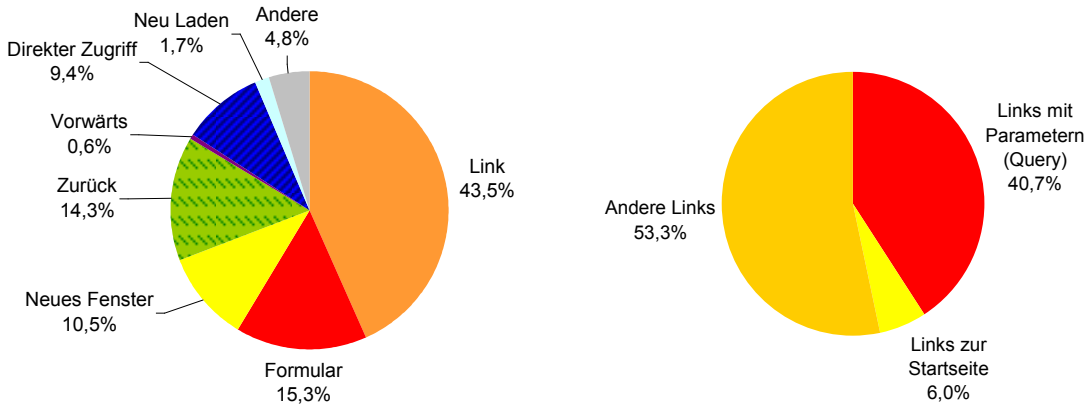


Abb. 12: Übersicht über die Aktionen der Studienteilnehmer der Web-Browsing-Studie von 2005. Links die Anteile aller Navigationsaktionen, rechts die unterschiedlichen Link-Aktionen (Weinreich, Obendorf et al. 2008).

Die Bedeutung der Verbindung von Hypertext-Navigation und Applikations-Interaktion kann im Web aus unterschiedlichen Perspektiven betrachtet werden:

- Aus der Sicht des Anwenders, der das Web zur *Informationsrecherche* nutzt, dienen web-basierte Dienste zum vereinfachten Zugriff auf Informationen: So besteht der erste Schritt eines Recherchevorgangs oft im Einsatz einer globalen Suchmaschine wie *Google*, die aus

[14] Da die früheren Studien teilweise unterschiedliche Kategorisierungen der Benutzeraktionen vornahmen, wurden für diese Übersicht einige der Werte zusammengefasst bzw. neu berechnet (s. Weinreich, Obendorf et al. 2006b).

[15] Der „direkte Zugriff" umfasst Navigationsaktionen wie Bookmarks, History und die direkte Eingabe der URL in der Adressleiste des Browsers (s. Weinreich, Obendorf et al. 2006b).

Hunderten von Millionen von Webseiten die (hoffentlich) relevantesten zur Anfrage ermittelt (s. Abschnitt 3.1.1). Dabei interagiert der Benutzer mit einer Online-Applikation, indem er Suchanfragen eingibt, sie reformuliert und innerhalb der Ergebnislisten blättert. Erst mit der Auswahl eines Suchergebnisses beginnt er mit der eigentlichen Hypertext-Navigation, bei der er seine Recherche weiter verfeinern kann.

Ein weiterer Aspekt der Integration von Anwendungen in die Informationsrecherche ist der zunehmende Anteil programmatischer Links, die mittels JavaScript neue Navigationsmöglichkeiten erzeugen. Solche Links führen beispielsweise dazu, dass der Benutzer *innerhalb* einer Seite navigiert und sich bei der Link-Auswahl lediglich ein Teil der Seite ändert oder dass er zusätzliche Informationen in Tooltips (s. Abschnitt 5.3.7) angezeigt bekommt. Clientseitige Skripte ermöglichen so ursprünglich für das Web nicht vorgesehene Navigationstechniken wie *Stretchtext*, *Note Links* und *Fluid Links* (s. Abschnitt 4.5.1.3, Abschnitt 5.3.8 und Anhang B.7).

- Bei einer zweiten Sichtweise des Webs nutzt der Anwender den Browser, um einen *Arbeitsablauf* zu erledigen, wie eine Flugbuchung, einen Einkauf oder das Lesen seiner E-Mails. Dabei interagiert er vermehrt mit Eingabeformularen; Links dienen nur noch zum Teil zum Zugriff auf weiterführende Informationen und sind stattdessen häufig mit Kommandos verknüpft, die die Web-Applikation steuern. Die Konzepte für hypertextbasierte Datenbank-Anwendungen wurden bereits mit *NoteCards* (s. Anhang B.9) und *HyperCard* (s. Anhang B.11) gelegt, bei denen Hyperlinks mit Skripten verknüpft waren und hierüber auch externe Programme integriert werden konnten.

  Das Web als global verteiltes Client-Server-System bietet darüber hinaus die Möglichkeit, auf die Online-Dienste unzähliger Anbieter zuzugreifen. Mangels verbindlicher Richtlinien oder Standards zur Gestaltung der Benutzungsschnittstelle weisen sie häufig sehr unterschiedliche Interaktionsmodelle auf, und der Web-Nutzer wird so regelmäßig mit neuen Herausforderungen bei der Nutzung dieser Systeme konfrontiert. Zusätzlich erlauben clientseitige Skripte und die asynchrone Kommunikation mit dem Server[16] die Entwicklung interaktiver Webseiten, die im Browser die Funktionalität verschiedenartiger Desktop-Applikationen nachbilden. Solche *Rich Internet Applications* dienen unter anderem als Textverarbeitungsprogramm, Tabellenkalkulation und E-Mail-Client.[17] Tim O'Reilly – Schöpfer des Begriffes *Web 2.0* – spricht in diesem Zusammenhang vom „Web as platform" (O'Reilly 2005) und drückt damit die Hoffnung aus, dass Benutzer für die meisten ihrer Aufgaben zukünftig nur noch den Webbrowser benötigen werden. Diese Rich Internet Applications weisen aber häufig gegenüber lokalen Anwendungen diverse

[16] Die asynchrone Kommunikation zwischen Web-Client und Server – also der Transfer zusätzlicher Daten zum Client, nachdem die Hauptseite bereits vollständig geladen wurde – wird zumeist unter dem Namen Ajax (für „**A**synchronous **J**avaScript **a**nd **X**ML") zusammengefasst (Garrett 2005).

[17] Für diese Bereiche findet man Beispiele bei Google mit *Google Text und Tabellen* (beide unter http://docs.google.com) und *Google Mail* (http://www.gmail.com/).

Einschränkungen – beispielsweise beim Austausch von Daten mit anderen Applikationen – auf, die ihre Gebrauchstauglichkeit wesentlich mindern können.

- In einer dritten Perspektive ist das Web eine flexible Erweiterung für Desktop-Anwendungen. Mit seiner Hilfe werden aktuelle Informationen, Updates und neue Funktionen für lokale Programme verfügbar gemacht. Anstatt beispielsweise das Hilfesystem auf der Festplatte des PCs zu installieren, wird auf Hilfeseiten im Web verwiesen, die bei Bedarf vom Anbieter schnell und problemlos zu aktualisieren sind. So schwindet zunehmend die Trennung lokaler und netzbasierter Anwendungen und Daten.

Das dynamische Web

Eine weitere besondere Eigenschaft des Webs ist der hohe Anteil dynamischer Ressourcen. Dies bezieht sich nicht nur auf die programmatisch generierten Ausgaben von Online-Applikationen (s. o.), sondern auch auf die vielen dynamisch erstellten Informationsseiten. Hierzu gehören z. B. die häufig aktualisierten Übersichtsseiten von Nachrichtendiensten, die einen Überblick der aktuellsten Informationen geben, oder Wikis und Blogs, bei denen zahlreiche Benutzer stetig Daten ergänzen und überarbeiten. Ein anderes Beispiel sind personalisierte Informationsdienste, die die Präsentation der Webseiten automatisch an die Vorlieben und Anforderungen der Benutzer anpassen.[18] Zur Personalisierung werden beispielsweise die früheren Benutzeraktionen, Browser-spezifische Parameter (Land, Spracheinstellung, Bildschirmauflösung) oder persönliche Anwenderprofile zugrunde gelegt.

Das Erstellen von Hyperlinks zu solch dynamischen Ressourcen wird erschwert, da sich Inhalt und Adresse schnell ändern können oder bestimmte, nur temporär gültige Parameter (wie eine Sitzungs-ID) zum Zugriff auf die Informationsquelle notwendig sind. Bei einer vom Autor dieser Arbeit durchgeführten Studie im Jahre 2005 hat der Anteil dynamischer Seiten, die sich innerhalb von einer Stunde ändern, bereits bei über 60% gelegen (s. Abb. 13; Weinreich, Obendorf et al. 2008). Statt auf konkrete Inhalte innerhalb einer Website zu verweisen, kann man bei solchen Angeboten oft nur die Hauptseite der Site als Link-Ziel verwenden, um defekte oder semantisch fehlerhafte Verweise zu vermeiden.

[18] Oft halten sich die Möglichkeiten allerdings noch in Grenzen: Die Personalisierung beschränkt sich auf grundlegende Eigenschaften, wie die Wahl des Schrifttyps und der Farben, der Anordnung von Elementen und die Inhalte der eingeblendeten Werbung.

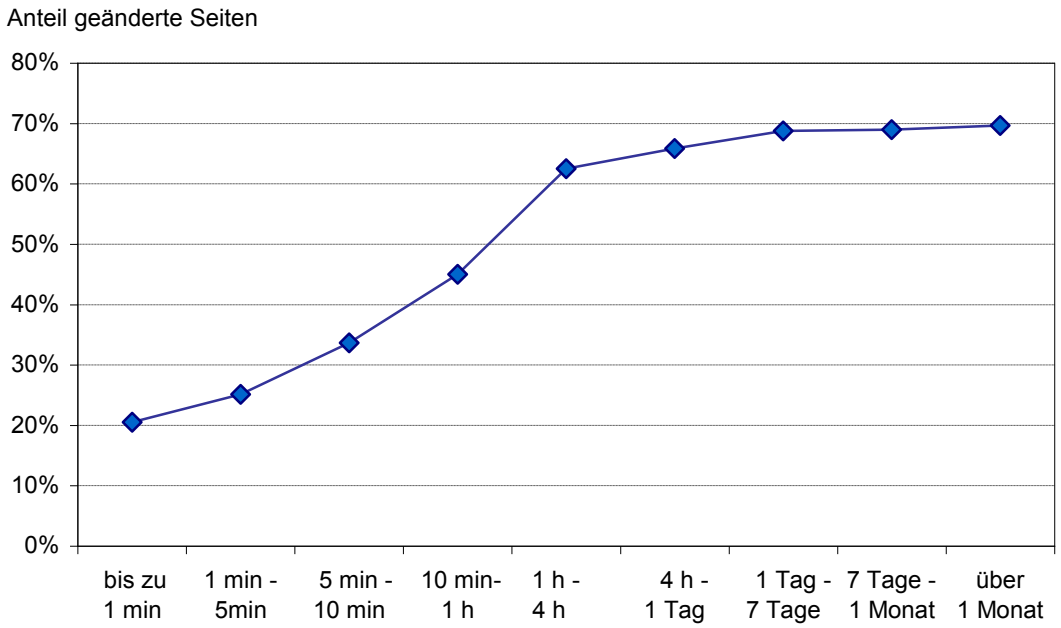


Abb. 13: Anteil geänderter Seiten für unterschiedliche Wiederbesuchszeiten (Weinreich, Obendorf et al. 2008).

Das Web als Publikations- und Kommunikationsmedium für jedermann

Als eine entscheidende Weiterentwicklung des World Wide Webs der letzten Jahre wird häufig die Möglichkeit hervorgehoben, dass inzwischen jeder Benutzer einfach selbst aktiv werden und eigene Beiträge im Web leisten kann. Dies geschieht auf unterschiedliche Weise: Beispielsweise verfügen viele bekannte Sites über Web-Foren, in denen die Benutzer Nachrichten, Angebote oder auch Medieninhalte (Fotos, Videos) kommentieren und bewerten können. Für den Leser erschließt sich so nicht nur eine neue Informationsquelle, er erhält auch die Möglichkeit zur Kommunikation mit anderen Besuchern derselben Site.

Das Erstellen eigener Webseiten ist inzwischen auch für Personen ohne IT-Fachwissen dank webbasierter *Autorensysteme*, *Wikis* und *Blogs* einfach geworden. So entstehen nicht nur gemeinsam erstellte Wissensarchive wie *Wikipedia* und *HowtoForge*,[19] die schnelle, grenzübergreifende Publikation von Informationen, Fotos und Filmen im Web führt auch zu einer Art „fünfter Macht im Staate“,[20] der einige autokratische Regimes bereits mit bekannten Mitteln wie der Zensur entgegenzutreten versuchen.

Die soziale Navigation im Web

Im Web haben sich inzwischen zahlreiche Konzepte der *sozialen Navigation* (s. Abschnitt 3.1.6) etabliert, die dem Anwender neue Möglichkeiten zum Zugriff auf die Daten bieten. Ein

[19] HowtoForge bietet technische Anleitungen für Open Source Software: http://www.howtoforge.com/

[20] Neben den drei Staatsgewalten *Legislative*, *Exekutive* und *Judikative* wird in demokratischen Verfassungsstaaten westlicher Prägung oft die *Presse* als vierte Gewalt bezeichnet. Das Web bietet den Bürgern eine eigene Plattform der öffentlichen Meinungsbildung, die von den vorherigen weitgehend unabhängig ist und zunehmend an Bedeutung gewinnt.

grundlegendes Element der sozialen Navigation im Web ist das *Tagging*: Dabei kategorisieren die Benutzer gemeinsam Ressourcen des Webs (Bilder, Videos, Lesezeichen von Webseiten), indem sie diese mit Schlagwörtern – den sogenannten *Tags* – versehen und so implizit einer Gruppe von Objekten zuordnen. Weitere Elemente der Gruppe werden angezeigt, wenn man eines der Tags auswählt. Die Ähnlichkeit von Ressourcen lässt sich über die Menge gemeinsamer Tags bestimmen. Auf diese Weise entsteht eine von den Benutzern erstellte *Folksonomy* (Guy & Tonkin 2006; Wal 2007), die im Gegensatz zu einer strukturierten Ontologie nicht von den Anbietern des Systems vorgegeben wird und beliebig erweiterbar ist (s. Abschnitt 3.1.6).[21] Allerdings handelt es sich bei solchen Tagging-Mechanismen um Website-spezifische Dienste, die kein Teil der Infrastruktur des Webs sind und somit nicht für beliebige Web-Ressourcen zur Verfügung stehen.

2.2.5 Resümee der Neuerungen des Webs

Ein Großteil der geschilderten Entwicklungen wird bereits seit 2005 unter dem Modewort *Web 2.0* zusammengefasst und in den Medien bis heute als revolutionäre Innovation angepriesen. Genau betrachtet handelt es sich allerdings um Konzepte, die auf Entwicklungen und Ideen der 1990er Jahre und früher zurückzuführen sind: Fast alle Hypertext-Systeme der dritten Generation boten den Benutzern die Möglichkeit, eigene Beiträge einzustellen sowie Links und Kommentare zu vorhandenen Ressourcen hinzuzufügen. Das einfache „Publizieren für alle" war auch ursprünglich schon für das Web vorgesehen,[22] ging aber bei den späteren Browsern (bereits ab Mosaic) verloren (s. Anhang B.15). Zudem war das Web von Anfang an als Kommunikationsmedium zwischen seinen Benutzern gedacht, erst die Kommerzialisierung der späten 1990er hat dies in den Hintergrund gerückt. Folglich kommentierte Tim Berners-Lee 2006 den Hype um das Web 2.0 auch etwas zynisch als „Kauderwelsch, von dem keiner weiß, was es bedeuten soll" *(„I think Web 2.0 is of course a piece of Jargon, nobody even knows what it means"*, siehe: Berners-Lee 2006; Roth 2006).

Allerdings gibt es einen entscheidenden Unterschied zwischen den gegenwärtigen Publikationsmöglichkeiten im Web und denen in früheren Hypertext-Systemen: Im Web sind sie *kein* integraler Bestandteil der Infrastruktur von Browsern und Servern, sondern es handelt sich um spezifische serverseitige Anwendungen, die sich in ihrer Funktionalität auf die jeweilige Site beschränken. Folglich können weder *beliebige* Dokumente mit zusätzlichen

---

[21] Implizit gewinnen so die Konzepte von taxonomischem Hypertext (siehe Abschnitt 2.1.4) für das Web an Bedeutung. Da beim Tagging aber unsystematisch vorgegangen wird, kommt es häufig dazu, dass semantisch gleichbedeutende Klassifikationsmerkmale (Tags) auftreten, die dennoch nicht in Beziehung zueinander stehen. Beispielsweise können die drei Tags „WWW", „Web" und „WorldWideWeb" zwar dasselbe Konzept repräsentieren, aber je nach Bentzervorlieben unabhängig voneinander eingesetzt werden. Ähnliche Probleme entstehen auch durch die Verwendung unterschiedlicher Sprachen in einem System.

[22] Das HTTP-Protokoll bietet entsprechende Kommandos wie PUT und DELETE, die aber von aktuellen Webbrowsern nicht unterstützt werden und erst im Zusammenhang mit Web-Diensten und Konzepten wie *REST (Representational State Transfer)* wieder zum Einsatz kommen (s. Abschnitt 7.4.1; Fielding 2000).

Links oder Kommentaren versehen werden, noch sind *persönliche* Anmerkungen für beliebige Ressourcen möglich, die nur vom Autor selbst einzusehen sind.

Trotz dieser Schwächen ist das Web inzwischen vielseitiger, größer, flexibler und vor allem dynamischer, als es anfangs abzusehen war. Aus Sicht des Autors dieser Arbeit kann man das Web zusammenfassend als eine Art globale *Hyper-Applikation* bezeichnen, da es sich einerseits um ein Hypertext-Informationssystem handelt, das ohne Dienste wie Suchmaschinen kaum nutzbar wäre und andererseits die Bedeutung von Web-Applikationen zur Bearbeitung von Aufgaben stetig zunimmt, diese aber weiterhin Hypertext-Charakteristika aufweisen. Der Übergang zwischen Informationssystem und Online-Applikation ist dabei so fließend, dass eine klare Trennung zwischen beiden Interaktionsformen in der Regel nicht möglich ist. Insgesamt bleibt das Web aufgrund seiner dokumentenzentrierten Benutzungsschnittstelle und der Dominanz der Links *im Kern ein Hypertext-System* (Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007), das wachsende Anforderungen an seine Benutzer stellt (s. auch Kapitel 9.2).

So werden die in den Webbrowsern integrierten Navigationshilfen den neuen Anforderungen des Webs inzwischen häufig nicht mehr gerecht (s. Kapitel 9.3.4). Die Benutzungsschnittstelle heutiger Webbrowser ähnelt noch sehr dem Interface der ersten Browser aus den Anfangstagen des Webs (s. Anhang B.15), und es scheint fast so, dass die Forschung zur Benutzbarkeit von Hypertext in einigen Bereichen zum Erliegen gekommen ist. Es zeigen sich zunehmend hieraus folgende Probleme bei der Benutzung des Webs, beispielsweise bei der Verwendung der Zurück-Funktion (*Backtracking*) in Online-Applikationen oder beim Wiederbesuch von dynamisch generierten Seiten nach längerer Zeit (Obendorf, Weinreich et al. 2007).

Zudem führen die neuen Möglichkeiten des Webs dazu, dass die Vielfalt der Link-Aktionen und Funktionen ständig zunimmt und dass der Benutzer immer häufiger nicht eindeutig erkennen kann, was die Folge der Interaktion mit einem Link sein wird (s. Abschnitt 4.4). Obwohl Hyperlinks das bedeutendste Bedienelement im Web sind, war bisher ihre Schnittstelle und der Umgang mit ihnen kaum Gegenstand systematischer Untersuchungen. Daher wird sich diese Arbeit genauer hiermit beschäftigen und helfen, diese Lücke zu schließen.

Nachdem in diesem Kapitel die Grundlagen von verteilten Hypertext-Informationssystemen dargelegt und einige Probleme in ihrer Gebrauchstauglichkeit gestreift wurden, geht das folgende Kapitel detailliert auf die Benutzbarkeit von verteiltem Hypertext ein. Dabei steht insbesondere die Rolle der Hyperlinks im Blickfeld: Erst wird analysiert, warum sie zum Zugriff auf Informationen in verteilten Informationssystemen unverzichtbar sind und welche Potenziale sie gegenüber anderen Navigationsmöglichkeiten besitzen. Danach wird auf die Benutzbarkeitsprobleme von Hypertext im Allgemeinen und von verteilten Hypertext-Informationssystemen im Besonderen eingegangen und die Rolle der Benutzungsschnittstelle von Links.

3 Zur Benutzbarkeit von verteilten Informationssystemen und Hypertext

Informationssysteme sollen den Umgang mit Informationen für Menschen einfacher und effizienter machen, dennoch bereiten sie häufig auch Probleme bei der Benutzung. Da es verschiedene Einsatzgebiete für Informationssysteme gibt (s. Abschnitt 2.1.3 und 2.1.5), stellen die Anwender unterschiedlichste Anforderungen. Beispielsweise stellen Menschen andere Ansprüche an ein Informationssystem, das in der Lehre eingesetzt wird, als an eines, das zur Unterstützung kreativer Prozesse dient oder zur Literaturrecherche benötigt wird. Individuelle Vorlieben und Erfahrungen sind weitere Faktoren, die die Benutzbarkeit für den Einzelnen beeinflussen, und sogar das Alter der Nutzer kann entscheidende Auswirkungen auf die durchschnittliche Zufriedenheit mit einem Informationssystem haben (Nielsen 1989).

Diese Dissertation legt den Schwerpunkt auf *verteilte Hypertext-Informationssysteme*, die den Zugang zu Informationen vereinfachen sollen, um Menschen bei ihrer (geistigen) Arbeit zu unterstützen. Die Anwender dieser Systeme haben zumeist bestimmte wissensbezogene Zielsetzungen; sie versuchen eine Information zu finden oder eine Aufgabe zu erledigen. Andere Bereiche der Hypertext-Forschung, beispielsweise literarischer Hypertext und hypertext-basierte Spiele, werden in dieser Arbeit ausgegrenzt (s. Kapitel 1.1). Trotz der thematischen Abgrenzung bleiben die Einsatzbereiche der betrachteten Systemkategorie umfangreich (s. Abschnitt 2.1.5).

Die Vielseitigkeit von Hypertext-Informationssystemen wirft die Frage auf, welche ergonomischen Faktoren für ihre Gebrauchstauglichkeit[23] entscheidend sind. Eine ausführliche Analyse der Ergebnisse von über 10 Jahren Forschung im Bereich der Benutzbarkeit von Hypertext-Informationssystemen durch Smith, Newman und Parks führte zu dem Ergebnis, dass es primär um die *Effizienz bei der Lösung von Aufgaben* geht (Smith, Newman et al. 1997). Die Autoren fanden zwei entscheidende ergonomische Faktoren heraus: Die benötigte Information muss *einfach und schnell zu finden* sein sowie in einer den *Aufgaben* des Benutzers *angemessenen Form* angeboten werden. Nach ihren Untersuchungen sei grundsätzlich zwischen dem *explorativen Browsen* und dem *zielgerichteten Suchen* zu unterscheiden. Bei Ersterem ist die Effizienz schlecht zu quantifizieren und muss über indirekte Faktoren – beispielsweise die Menge der erinnerten Items – bestimmt werden. Beim zielgerichteten Suchen können hingegen Daten wie die *Korrektheit gefundener Lösungen*, die *benötigte Zeit* und die *Länge des Navigationspfades*[24] ausgewertet werden (Smith, Newman et al. 1997).

---

[23] Zur Definition grundlegender Begriffe wie Gebrauchstauglichkeit und Benutzbarkeit sei auf das Glossar der Arbeit verwiesen. Auf Verfahren zur Evaluation der Benutzbarkeit von Software wird in Kapitel 6.1.2 eingegangen.

[24] Der Begriff Navigation bezeichnet bei Hypertext-Informationssystemen die Aktionen eines Benutzers, um an eine gewünschte Information zu gelangen (siehe auch Glossar).

Um den einfachen und schnellen Zugang zu den digitalen Inhalten in Informationssystemen zu gewährleisten, wurden unterschiedliche Techniken und Werkzeuge zur Navigation im Informationsbestand entwickelt; Links sind – auch in Hypertext-Systemen – nur eine hiervon (Meyrowitz & Van Dam 1982a). Diese unterschiedlichen Navigationswerkzeuge haben je nach Aufgabe, Anforderung, Vorlieben, und Hintergrundwissen des Anwenders spezifische Stärken und Schwächen, wie in zahlreichen Studien belegt wurde (Nielsen 1990; Höök, Sjölinder et al. 1996; Nation 1997; Tauscher & Greenberg 1997; Ford & Chen 2000; Graff 2005; Kim 2006; Weinreich, Obendorf et al. 2008). Beispielsweise beobachteten Hammond und Allinson, dass ihre Teilnehmer die *Guided Tour* (s. Abschnitt 3.1.5) intensiv zur Erkundung des Systems einsetzten (28 % der Zeit), wogegen sie sie für die Suche nach konkreten Informationen kaum berücksichtigten (8 % der Zeit, siehe: Hammond & Allinson 1989). Marchionini und Shneiderman kamen bei einer Untersuchung zu dem Ergebnis, dass *Index* und *hierarchische Navigation* für die gezielte Suche besonders effizient waren, wogegen *Hyperlinks* für das explorative Erkunden von Texten und die Informationssuche durch unerfahrene Computernutzer Vorteile zeigten (Marchionini & Shneiderman 1988: 76).

Ein wichtiger Aspekt für eine gute Benutzbarkeit eines Hypertext-Informationssystems ist daher die Verfügbarkeit unterschiedlicher Navigationsmöglichkeiten: Dies bezieht sich sowohl auf die bereitgestellten Navigationswerkzeuge wie Suche, hierarchische Ansicht und Index als auch auf die Vielfältigkeit der angebotenen Hyperlinks, die direkt zu zahlreichen weiteren Dokumenten mit verschiedenartigem semantischen Bezug führen sollten.

Der folgende Abschnitt 3.1 gibt einen kurzen Überblick über die wichtigsten Navigationswerkzeuge in Informationssystemen und geht auf die jeweiligen Potenziale und Grenzen ein. Darauf basierend wird gezeigt, wieso Hyperlinks für verteilte Informationssysteme ein unverzichtbares Navigationsmittel darstellen. In Abschnitt 3.2 folgt eine Analyse der grundlegenden Benutzbarkeitsprobleme von Hypertext und ihrer Ursachen. Teil 3.3 geht auf die zusätzlichen Herausforderungen für die Gebrauchstauglichkeit verteilter Hypertext-Informationssysteme ein, die sich durch die Offenheit der Systeme, die großen Datenmengen und die (globale) Verteilung der Informationen ergeben.[25] Abschließend werden die Rolle der Hyperlinks bei den aufgeführten Benutzbarkeitsproblemen sowie mögliche Ansätze zu ihrer Reduzierung diskutiert.

3.1 Benutzbarkeit von Navigationswerkzeugen in verteilten Informationssystemen

Für Informationssysteme wurden zahlreiche Werkzeuge zum Zugriff auf die Informationen entwickelt, die jeweils charakteristische Stärken und Schwächen in Abhängigkeit vom Anwender, seinen Aufgaben und der Art und dem Umfang der verfügbaren Informationen zei-

[25] Der Autor dieser Arbeit hat sich bereits in (Weinreich 1997) mit der Performanz, Navigation und Orientierung im World Wide Web auseinandergesetzt. Einige Abschnitte dieses Kapitels bauen auf den damaligen Arbeiten auf.

gen. Im Folgenden werden die wichtigsten dieser *Navigationswerkzeuge* vorgestellt und ihre jeweiligen Potenziale und Grenzen für den gezielten Zugriff auf Informationen in großen, verteilten Informationssystemen analysiert. Die Übersicht beginnt mit der *Suchfunktion*, gefolgt von der *hierarchischen Navigation*, dem *Index* und der *Guided Tour.* Danach werden Filtermechanismen der *sozialen Navigation* und Navigationswerkzeugen zum *Wiederfinden von Dokumenten* behandelt. Abschließend wird durch eine Vergleichsanalyse dargelegt, warum *Hyperlinks* für die Gebrauchstauglichkeit großer, verteilter Informationssysteme wie dem Web von besonderer Bedeutung sind.

3.1.1 Die Suchfunktion

Eine *Suchfunktion* gibt Benutzern die Möglichkeit, schnell nach bestimmten Begriffen in einem Informationssystem mit großem Datenbestand zu recherchieren und über eine Ergebnisliste direkt auf ein gewünschtes Dokument zuzugreifen. Sie stellt einen der bedeutendsten Vorteile elektronischer Informationssysteme gegenüber gedruckten Dokumenten dar.

Insbesondere bei inhomogenen und umfangreichen Informationssystemen wie dem World Wide Web ist eine Suchfunktion unverzichtbar (ISO9241-151 2008). Für die globale Navigation im Web ist sie das primäre Navigationsmittel, um an *neue* Informationen zu gelangen (Jansen & Spink 2006; Sullivan 2006; Lewandowski & Höchstötter 2007). Der Grund hierfür liegt vor allem in der schier unüberschaubaren Menge von Dokumenten und der fehlenden Gesamtstruktur des Webs. Nur Suchmaschinen sind gegenwärtig in der Lage, einen Großteil des Webs zu erfassen. Sie nutzen dabei die Hyperlinks auf den Webseiten zur Erstellung des Suchindexes: *Web-Crawler* (vergleiche Abschnitt 8.6.2.1ff) folgen den Links rekursiv, um die Ressourcen zu indexieren.[26] Dieses Vorgehen bietet den Vorteil, dass neue Seiten und sogar ganze Websites automatisch mit aufgenommen werden, sofern zu ihnen lediglich ein einziger Hyperlink von einer bereits indexierten Seite existiert.

Es gibt zahlreiche elementare Probleme bei der Benutzung von Suchsystemen. Eines liegt darin, dass Benutzer Anfragen aktiv formulieren müssen und nicht aus vorgegebenen Begriffen auswählen können. So kommt es häufig zu einer mangelnden Übereinstimmung zwischen der Benutzeranfrage und der Repräsentation der Informationen im System (s. Abb. 14), wodurch die Suchergebnisse oft nicht die gewünschten Dokumente enthalten (Robertson 1977).

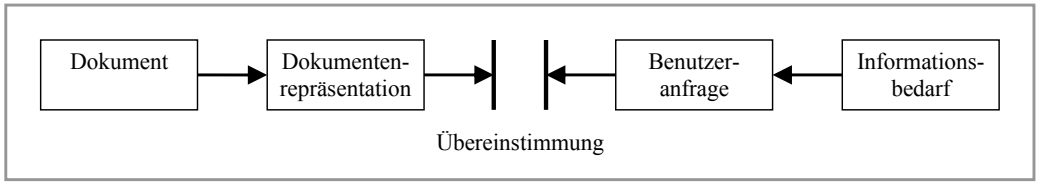


Abb. 14: Das Information-Retrieval-Modell von Robertson (nach Robertson 1977).

[26] Dabei bleibt ein großer Teil des Webs, der als „Deep Web" bezeichnet wird, unerfasst. Dies bezieht sich auf Ressourcen, die nicht über Hyperlinks zugänglich sind, sondern beispielsweise eine Formulareingabe oder ein Login bei einer Web-Applikation erfordern (Bergman 2001).

Beim Web kommt erschwerend hinzu, dass HTML-Dokumente keine *Metadaten* wie Autorenangaben oder Schlüsselworte erfordern und die gängigen Web-Suchmaschinen keine entsprechenden Attribute für die Suche unterstützen (s. Abb. 15). Die Recherche im Web beschränkt sich im Wesentlichen auf die *Volltextsuche*, deren mangelnde Effizienz schon lange vor dem Web nachgewiesen wurde, da sie häufig sehr viele unpassende Dokumente zurückliefert (Blair & Maron 1985; Blair & Maron 1990). Erst durch neue Ranking-Methoden, die ab Ende der 1990er Jahre entwickelt wurden und die Hypertext-Struktur des Webs berücksichtigen, hat man die Ranking-Problematik halbwegs in den Griff bekommen (s. u.).

Das Forschungsgebiet des *Information Retrieval* setzt sich seit Jahrzehnten mit der Problematik der Effizienz von Suchsystemen auseinander. Zur Beurteilung der Güte eines Suchsystems wurden unter anderem als grundlegende Maße *Recall* (als Wahrscheinlichkeit, dass ein relevantes Dokument gefunden wird) und *Precision* (die Wahrscheinlichkeit, dass ein gefundenes Dokument relevant ist) definiert (s. Lancaster & Fayen 1973, Kapitel 6). Da sich diese Werte aus absoluten Zahlen – wie der Anzahl korrekter Suchergebnisse im Verhältnis zu allen vorhandenen Dokumenten – berechnen und solche absoluten Zahlen für das Web nicht verfügbar sind, kann man diese Kriterien auf die Web-Suche nicht mehr ohne Weiteres anwenden.[27] Für die globale Suche im Web ist aufgrund der häufig zahlreichen Treffer die Güte des *Rankings* – also der Reihenfolge der Elemente in der Trefferliste – zu einem zentralen Element der Gebrauchstauglichkeit der Systeme geworden (Lewandowski & Höchstötter 2007). Viele Forschungsgruppen und Anbieter von Suchmaschinen befassen sich intensiv mit der Weiterentwicklung entsprechender Algorithmen. Die zwei bekanntesten grundlegenden Strategien sind in diesem Zusammenhang das *Query Specific Ranking* (Kleinberg 1999) und das *PageRank Citation Ranking* (Page, Brin et al. 1998). Beide Verfahren sowie die vielen darauf aufbauenden Konzepte berücksichtigen die *Link-Struktur* des Webs (insbesondere die eingehenden Links), um die Relevanz der Dokumente zu bestimmen, wodurch die (von Menschen erstellten) Hyperlinks auch für die Funktion globaler Web-Suchsysteme maßgeblich geworden sind.

[27] Für Web-Suchmaschinen werden zur Bestimmung von Precision und Recall stattdessen häufig die ersten Ergebnisse mehrere Suchmaschinen verglichen (vergl. Chu & Rosenthal 1996; Shafi & Rather 2005).

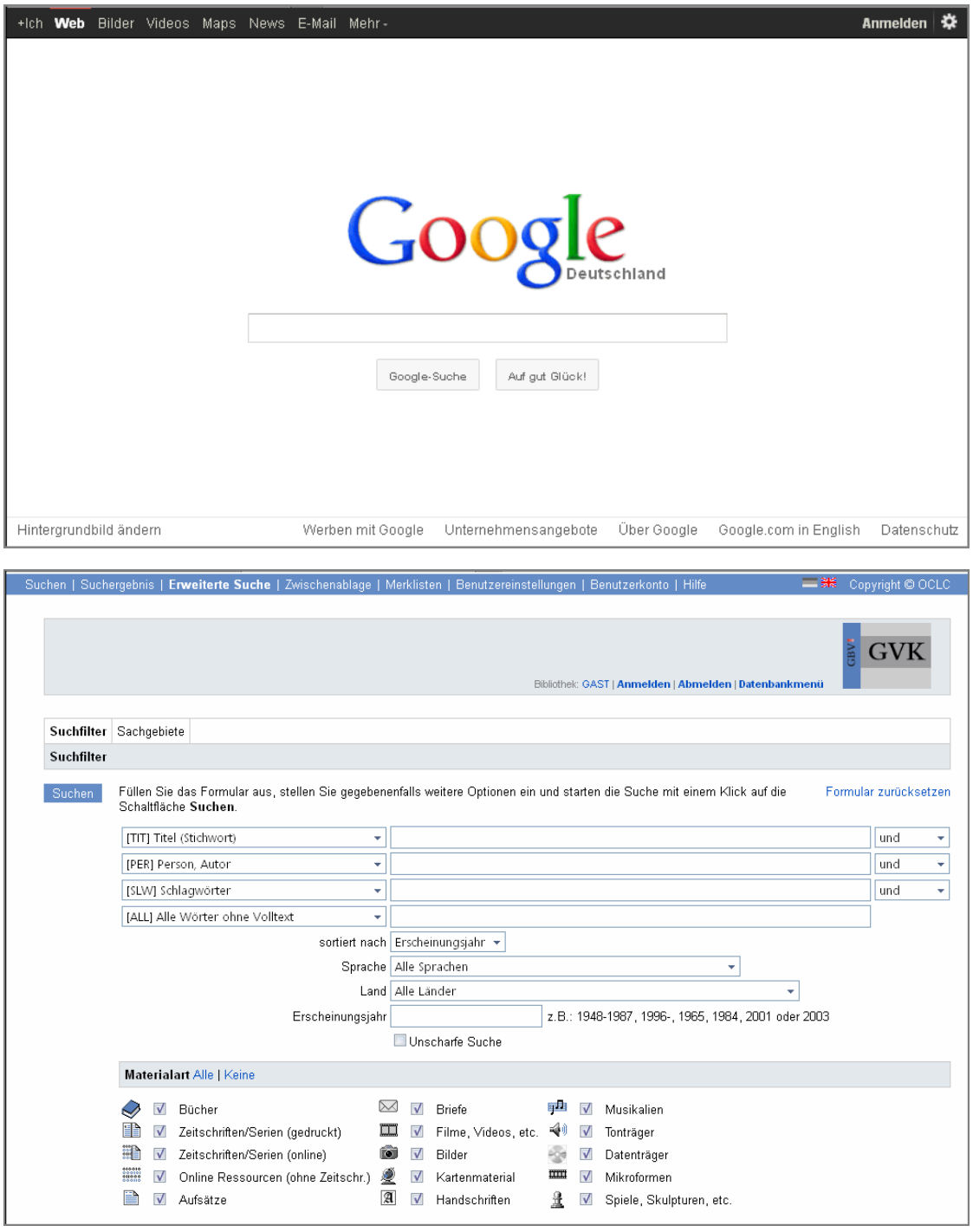


Abb. 15: Oben die typische „minimalistische“ Benutzungsschnittstelle für die Suche im Web, unten die Recherche-Schnittstelle einer Bibliotheksdatenbank, die eine wesentlich präzisere Anfrage erlaubt.

3.1.2 Hierarchische Navigation

Eine gängige Methode zur Strukturierung von Informationen und Objekten sind *Hierarchien*. Ihre Bedeutung zeigt sich bereits bei einem Blick auf gedruckte Bücher: Neben dem linearen Strukturierungsprinzip der Seite bieten sie auch die Hierarchie von Kapiteln und Unterkapiteln sowie Abschnitten und Absätzen. Nur so lassen sich Sachbücher effizient überblicken und thematische Bereiche schnell eingrenzen (DeRose 1989).

Die meisten elektronischen Informationssysteme bieten für den Zugriff auf die Inhalte eine hierarchische Übersicht an (s. Abb. 16 links und Tabelle 5), und sie gilt im Allgemeinen zur Navigation und Orientierung in größeren Informationssystemen als unverzichtbar (Mohageg 1992: 352; Andrews 1996; Nielsen 2000). Auch grafische Desktop-Oberflächen machen sich Hierarchien zunutze, beispielsweise zur Strukturierung des Dateisystems und der Pull-down-Menüs. Obwohl die Protokolle und Dienste des Webs nicht explizit hierarchische Strukturierungsmechanismen unterstützen,[28] verfügt ein überwiegender Anteil der Informationsangebote des Webs implizit über eine hierarchische Struktur (s. auch ISO9241-151 2008), in der sich mittels *struktureller Hyperlinks* (s. Abschnitt 4.5.1.1) navigieren lässt (s. Abb. 16 rechts).

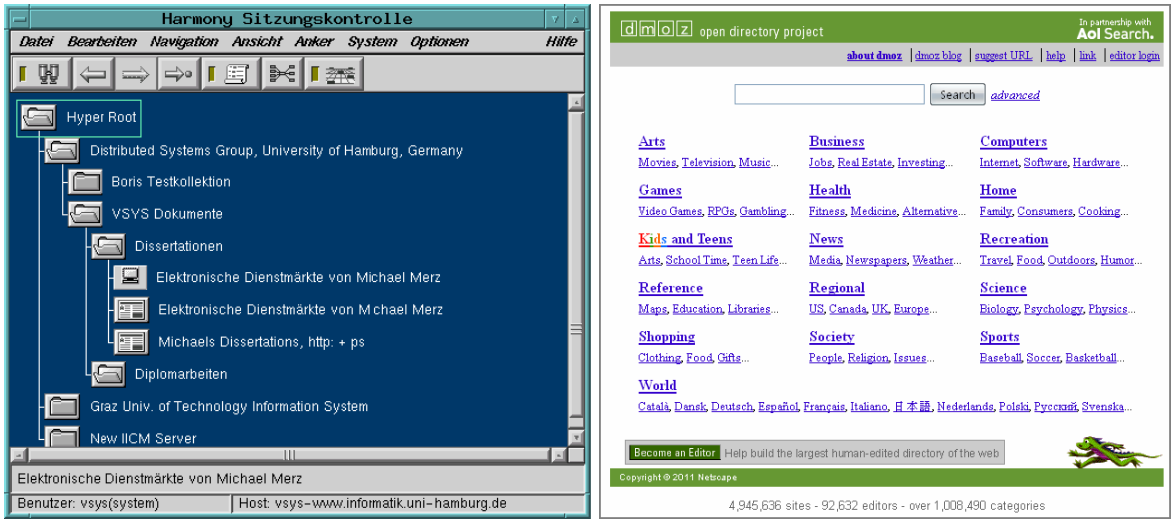


Abb. 16: Links die hierarchische Ansicht des Hyper-G Systems (s. Anhang B.14; Andrews 1996), rechts die Übersichtsseite des hierarchisch strukturierten Open Directory Projects DMOZ.

Die konzeptionelle Stärke von Hierarchien zur semantischen Strukturierung von Daten liegt darin, dass dem Benutzer auf jeder Hierarchieebene eine beschränkte Anzahl von Auswahlmöglichkeiten angeboten werden kann und so die einzelnen Ebenen überschaubar bleiben (s. auch ISO9241-14 1997). Darüber hinaus werden überblickartiges Wissen und mentale Modelle vom menschlichen Intellekt zum Großteil hierarchisch organisiert und verarbeitet (Stevens & Coupe 1978), und das Langzeitgedächtnis gruppiert Informationen in Kategorien, die ihrerseits bevorzugt in hierarchischer Beziehung zueinander stehen (Collins & Quillian 1969).

Trotz dieser Vorteile hat die hierarchische Navigation ihre Grenzen. Bereits bei kleineren lokalen Informationssystemen ergibt sich oft das Problem, dass Benutzer ein anderes *mentales Modell* von der Struktur des Informationsangebotes haben, als es durch die Autoren

[28] Mit dem „`link`"-Element gibt es einen Ansatz, um strukturelle Informationen in HTML einzubinden, der aber nur recht einfache Links zwischen den Dokumenten einer Site gestattet (s. Kapitel 4.4.7). Vielversprechender ist der „Sitemaps"-Standard, der 2006 von mehreren Anbietern von Suchmaschinen (Google, Yahoo! und Microsoft) verabschiedet wurde und mit dem sich die Struktur einer Website als XML-Datei beschreiben lässt. Diese Informationen werden aber von aktuellen Browsern nicht ausgewertet. Siehe: http://sitemaps.org/.

vorgegeben wurde. Dies kann die Navigation wesentlich erschweren und das Finden gewünschter Dokumente fast unmöglich machen (Spool, Scanlon et al. 1998). Die Ursache für diese Diskrepanz ist darin begründet, dass hierarchische Modelle und Strukturen durch das Vorwissen und die Aufgaben eines Menschen bestimmt werden (Tulving & Thompson 1973), und die individuelle Wahrnehmung der Sachverhalte entscheidenden Einfluss auf den Aufbau des hierarchischen mentalen Modells hat (Stevens & Coupe 1978).

Im global verteilten World Wide Web sind unterschiedliche Modelle durch die heterogenen Benutzergruppen und die unterschiedlichen Interessen unvermeidbar. Eine weitere Einschränkung ergibt sich durch die Verteilung der Informationen auf sehr viele Anbieter. Es gibt zwar Bestrebungen, möglichst viele Web-Angebote in redaktionell bearbeitete hierarchische Verzeichnisse wie das *Yahoo! Directory*[29] oder *DMOZ*[30] (s. Abb. 16 rechts) aufzunehmen, diese Kataloge umfassen aber nur einen kleinen Teil des Webs und sind dennoch sehr umfangreich und schwer überschaubar. Zudem eignen sich die Verzeichnisse wegen der Unidirektionalität der Links im Web nur für das *Auffinden* von Angeboten, nicht aber für die Navigation zwischen inhaltlich verwandten Angeboten, wenn man zu einer Seite auf andere Weise gelangt ist. Hier zeigt sich die Stärke der assoziativen Hyperlinks, mittels derer auch zwischen den Dokumenten unterschiedlicher Anbieter verwiesen werden kann und der Benutzer den direkten Zugriff auf weiterführende Informationen erhält.

3.1.3 Der Index

Bei Sachbüchern ist der *Index* eine vertraute Möglichkeit, um schnell auf Informationen zu bestimmten Stichworten zuzugreifen. Ein Index kann auch für Informationssysteme als alphabetisch geordnetes Verzeichnis wichtiger Begriffe, Themen, Namen oder Dokumenttitel angeboten werden. Die Erstellung des Index erfolgt dabei entweder manuell oder automatisch aus markierten Begriffen innerhalb der Dokumente des Informationsangebotes (Horton 1990: 91). Hieraus ergibt sich die Beschränkung, dass im Index nur die Begriffe zu finden sind, die durch die Autoren des Informationssystems vorgegeben wurden.

Inzwischen wird der Index bei Informationssystemen nur noch relativ selten eingesetzt. Dabei wird seine Nützlichkeit wahrscheinlich unterschätzt: Bei einer Studie von Marchionini und Shneiderman war der Index bei der Suche nach Informationen in einem Hypertext-System gerade bei Anfängern beliebt: 14 der 16 Teilnehmer bevorzugten die alphabetische Liste der Seiten gegenüber dem ebenfalls angebotenen hierarchischen Menüsystem (Marchionini & Shneiderman 1988: 74). Allerdings stößt ein Index aufgrund seiner mangelnden Struktur bei einer größeren Anzahl von Einträgen schnell an seine Grenzen, und für das Web ist er auf globaler Ebene – bei Milliarden von Dokumenten – kaum einsetzbar.

---

[29] Yahoo! begann 1994 als Verzeichnis von Websites, inzwischen ist es eine der größten Internet-Firmen, die Online-Dienste aller Arten anbietet. Das Yahoo! Directory findet man heute unter http://dir.yahoo.com/.

[30] DMOZ ist das "Open Directory Project", das von über 90.000 Freiwilligen geführt wird und fast 5 Millionen Einträge enthält (Stand Ende 2011, s. Abb. 16 rechts). Adresse: http://www.dmoz.org/.

3.1.4 Übersichtskarten

Ein ähnliches Problem gibt es bei der Navigation mit *Übersichtskarten.* Sie geben dem Benutzer einen grafischen Überblick zur Strukturierung der Dokumente und können sowohl hierarchische Strukturen repräsentieren als auch das durch die Hyperlinks aufgespannte Netz. Karten erleichtern die Orientierung, indem sie die aktuelle Position des Benutzers im System visualisieren (Edwards & Hardman 1999) und Zusammenhänge zwischen Informationen erfassbar machen (Dalitz & Heyer 1995: 51). Es gibt zahlreiche unterschiedliche Formen von Karten (s. Abb. 17), die sich beispielsweise anhand der Repräsentation der Objekte, dem Layout-Algorithmus, der Anzahl verwendeter Dimensionen und der Interaktionsform unterscheiden.[31]

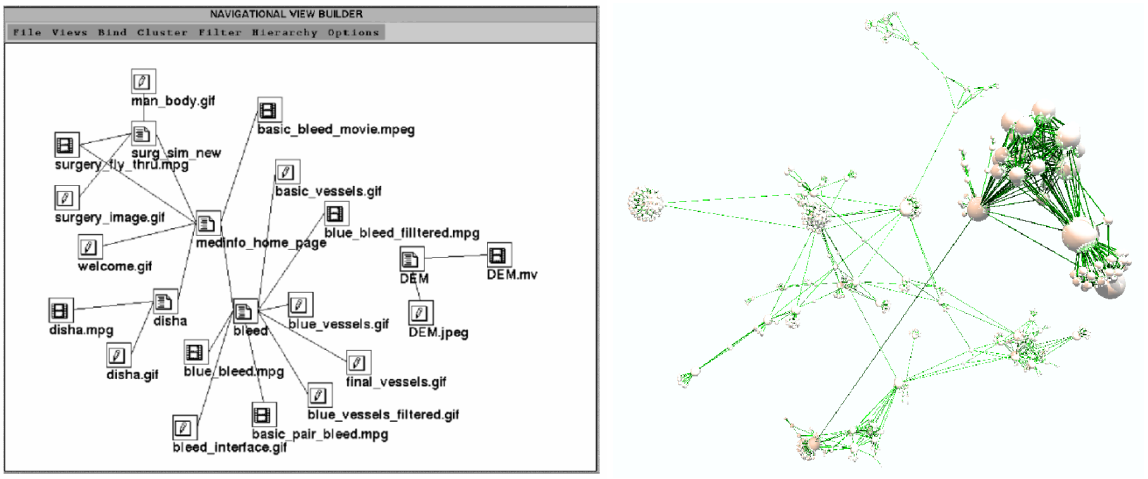


Abb. 17: Zwei Werkzeuge zur Visualisierung von Web-Ressourcen: Links eine 2D-Übersicht bestimmter Dokumenttypen im „Navigational View Builder" (Mukherjea & Foley 1995), rechts die 3D-Ansicht einer Website mit dem System „Narcissus" (Hendley, Drew et al. 1995).

Übersichtskarten ist der gravierende Nachteil zu eigen, dass sie aus Platzgründen immer nur eine abstrahierte Repräsentation der Knoten und Links darstellen können, deren Zuordnung zu den Dokumenten und den Links in den Dokumenten oft Schwierigkeite bereitet. Zudem haben Studien gezeigt, dass Karten für größere Objektmengen schnell an ihre Grenzen stoßen, da sie unüberschaubar werden (Utting & Yankelovich 1989).

3.1.5 Die „Guided Tour"

Eine *„Guided Tour"* (auch *Rundgang* genannt) verknüpft ausgewählte Dokumente eines Informationssystems sequenziell, um dem Leser einen Einblick in ein bestimmtes Thema zu geben (Trigg 1988). Die Idee, inhaltlich zusammenhängende Dokumente zu solchen „Pfaden" zu verknüpfen, geht auf das ursprüngliche Hypertext-Konzept von Vannevar Bush zurück (s. Anhang A.1; Bush 1945a). Im Vergleich zu Bushs Pfaden steht bei der Guided Tour aber die geleitete Navigation im Vordergrund, sodass sich der Benutzer nicht bei jedem Dokument neu entscheiden soll oder muss, welchen Pfad bzw. Link er folgen will. Das minimiert einerseits das Risiko der Desorientierung und erlaubt es dem Leser, sich stärker auf die

[31] Übersichten zu dem Thema geben (Card, Mackinlay et al. 1999; Mayer 2008, S. 47ff).

Inhalte zu konzentrieren; andererseits schränkt es die Möglichkeiten der Navigation ein, sodass die Guided Tour für viele Recherche-Aufgaben keine Alternative zu anderen Navigationstechniken darstellt (Zellweger 1989).

3.1.6 Filtermechanismen, Empfehlungssysteme und soziale Navigation

In den letzten Jahren haben *Filtermechanismen*, die auf Prinzipien der *sozialen Navigation* beruhen, bedeutend an Relevanz gewonnen. Beim ***automatischen kollaborativen Filtern*** wird auf Basis der Bewertungen anderer Benutzer eine individuelle Vorauswahl der Dokumente getroffen (Resnick, Iacovou et al. 1994; Hussein & Ziegler 2011). Fortgeschrittene *Empfehlungssysteme* nutzen zusätzlich *Meta-Informationen* über die Dokumente und die Ähnlichkeiten zwischen *Benutzerprofilen*, um die Relevanz der Bewertungen für den Einzelnen zu berechnen und so die passendsten Objekte zu bestimmen (Wörndl, Bader et al. 2011). Solche Verfahren weisen gerade in Systemen mit umfangreicher Dokumentenbasis ein großes Potenzial auf, die Orientierung und Navigation zu erleichtern (Baier, Weinreich et al. 2004; Wollenweber 2004).

Im Web haben sich bisher keine global einsetzbaren Konzepte für das Hinzufügen und Verarbeiten von entsprechenden Meta-Informationen zu den Ressourcen durchsetzen können, obgleich sich die *Semantic Web Initiative* seit 2000 mit diesen Themen befasst (Herman, Hawke et al. 2011). Im Juni 2011 haben sich die drei großen Anbieter der Suchmaschinen *Google*, *Microsoft Bing* und *Yahoo!* nach mehreren Jahren auf einen Standard mit dem Namen *Microdata* geeinigt, der für (X)HTML zusätzliche semantische Attribute definiert, um Websites für die Suche besser maschinell auswerten zu können.[32] Bisher wird dieser Standard jedoch kaum eingesetzt. Ein Grund für die Probleme bei der Einführung solcher Standards kann sein, dass Menschen häufig Verständnisschwierigkeiten mit vorgegebenen Schemata semantischer Attribute haben oder sich durch sie zu sehr eingeschränkt fühlen (Kopak 1999).

Stattdessen haben sich seit ca. 2006 wesentlich einfachere Methoden zur *freien Kategorisierung* von Inhalten einzelner Websites durch die Anwender etabliert: Das ***kollaborative Tagging*** beschreibt einen Prozess, bei dem Benutzer beliebige Schlüsselwörter, genannt *Tags*, für gemeinsame Inhalte wie Fotos, Bookmarks oder Literaturlisten vergeben, um eine sogenannte *Folksonomy* zu erstellen[33] (Müller-Prove 2007). Die Tags kennzeichnen und verknüpfen inhaltlich ähnliche Ressourcen und erlauben die Navigation zwischen ihnen. Obgleich Probleme bei der Verwaltung solch unstrukturierter Kategorisierungen über längere Zeit bekannt sind (Guy & Tonkin 2006), ist das Tagging ausgesprochen populär, und Studien haben gezeigt, dass es die Zufriedenheit der Benutzer mit Online-Systemen erhöhen kann (Gedikli, Ge et al. 2011). Dienste, die ihren Erfolg auch den Möglichkeiten des Tagging verdanken,

[32] Die gemeinsamen Vorschläge für die neuen semantischen Attribute und Werte in HTML5 findet man auf der Website http://schema.org/.

[33] Der Begriff Folksonomy wurde von Thomas Vander Wal geprägt: http://www.vanderwal.net/folksonomy.html.

sind die Bookmark-Verzeichnisse *delicious* und *Digg*, das Fotoportal *Flickr* und das Verzeichnis bibliografischer Daten *BibSonomy*[34] (s. auch Abschnitt 2.2.3ff und 9.2ff).

Noch simpler und erfolgreicher sind die verteilten Empfehlungssysteme einiger großer Online-Dienste. Das soziale Netzwerk *Facebook* bietet beispielsweise seit April 2010 einen „Like"-Button, der sich in beliebige Webseiten einbinden lässt, und mit dem man seine „Freunde" direkt auf interessante Informationen hinweisen kann (Newman 2010). Ein Link zu dem „gemochten" Dokument wird dann Freunden auf ihrer Startseite (auch genannt „*Neuigkeiten*" bzw. Englisch „*News Feed*") als eine Art Empfehlung dargestellt (Fletcher 2010). Der Suchdienst *Google* versucht seit März 2011 aufzuschließen und bietet eine Funktion namens „*+1*" (sprich: „*Plus One*") an,[35] mit der registrierte Benutzer Suchergebnisse und Webseiten anderen Google-Nutzern[36] weiterempfehlen können.

3.1.7 Navigationswerkzeuge zum Wiederfinden von Objekten

Neben den aufgeführten Konzepten zum Finden *neuer* Informationen in Informationssystemen gibt es einige Navigationswerkzeuge, die das *Wiederfinden* von bereits gesichteten Dokumenten erleichtern sollen.[37]

Von zentraler Bedeutung ist das ***Backtracking*** (auch *Zurück-* oder *Back-Button* genannt), das dem Benutzer das Wiederkehren zu einem (kurz) zuvor betrachteten Objekt erlaubt. Frühere Studien haben gezeigt, dass dieses Werkzeug für die Navigation in Hypertext-Systemen nahezu unverzichtbar ist (Akscyn, McCracken et al. 1988; Shneiderman & Kearsley 1989) und auch bei Webbrowsern eines der wichtigsten Navigationsmittel darstellt (s. Tabelle 1 und Abb. 12; Catledge & Pitkow 1995; Tauscher & Greenberg 1997; Obendorf, Weinreich et al. 2007). Allerdings sind mit dem Backtracking auch zahlreiche Probleme verbunden, die mit der Struktur und Repräsentation der Liste zuvor besuchter Objekte (*History-Stack*) zusammenhängen (Greenberg & Cockburn 1999), sich bei der Verwendung mehrerer Browser-Tabs oder Fenster ergeben und im Zusammenhang mit Online-Applikationen auftreten (Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Dubroy & Balakrishnan 2010; Huang & White 2010).

Für das Zurückkehren zu Seiten nach *längerer Zeit* ist das Backtracking zumeist ungeeignet. Zwei alternative Werkzeuge sind *History* und *Bookmarks*. Die ***History*** eines Informationssystems (auch *Verlauf* und *Chronik* genannt) bietet einen Überblick der in den letzten Tagen und Wochen besuchten Ressourcen und gestattet dem Benutzer die direkte Auswahl früher

---

[34] Die aufgeführten Dienste sind unter den Adressen http://del.icio.us/, http://digg.com/, http://flickr.com/ und http://bibsonomy.org/ zu finden.

[35] Google stellt den Dienst „+1" unter folgender Adresse vor: http://www.google.com/+1/button/

[36] Diese Benutzer müssen ebenfalls über einen Konto bei Google verfügen. Die Beziehung stellt Google über die Adresslisten der Benutzer und die Kontakte beim sozialen Netzwerk Google+ her.

[37] Einen detaillierten Überblick zu den Konzepten und Werkzeugen zum Wiederfinden von Objekten im Web findet man in (Mayer 2008, Kapitel 2).

eingesehener Objekte. Im Vergleich zum Backtracking zeigt die History meist mehr Informationen über die Dokumente an, um das Wiedererkennen zu erleichtern. Hierzu können neben dem Seiten-Titel, der Zeitpunkt des letzten Zugriffs und die Verweildauer gehören (Utting & Yankelovich 1989; Nielsen 1993a: 28). Einige Systeme bieten auch strukturelle Informationen, wie die Position des Knotens in der hierarchischen Struktur eines Angebots (Thüring, Hannemann et al. 1995: 65) oder eine verkleinerte Darstellung (ein *Thumbnail*) des Objektes (Cockburn, Greenberg et al. 1999; Kaasten, Greenberg et al. 2001; Zühlsdorff 2002; Won, Jin et al. 2009). Frühere Studien haben gezeigt, dass es zudem sinnvoll sein kann, Benutzern die Möglichkeit zur Annotation der Einträge in der History zu geben (Utting & Yankelovich 1989). Zwar verfügen alle aktuellen grafischen Webbrowser über eine History-Funktion (s. Abb. 18 links unten), Studien haben aber gezeigt, dass sie fast gar nicht genutzt wird und gravierende Schwächen aufweist (Tauscher 1996; Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Weinreich, Obendorf et al. 2008; Won, Jin et al. 2009).

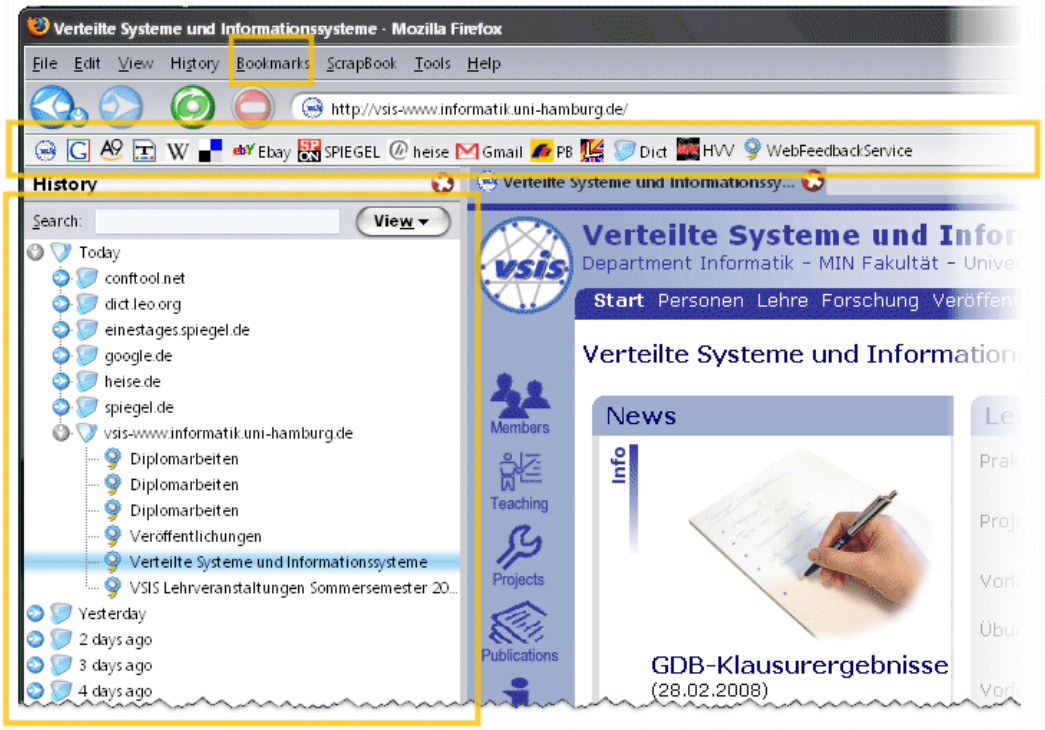


Abb. 18: Oben das Bookmark-Menü und der Bookmark-Toolbar, links unten die History im Sidebar von Firefox 3.

Mithilfe von ***Bookmarks*** (auch *Lesezeichen* oder *Favoriten* genannt) kann der Benutzer interessante Dokumente vermerken, um sie später gezielt wieder aufzurufen (s. Abb. 18). Im Gegensatz zur History werden als Bookmark nur die Knoten gespeichert, die der Benutzer zum Zeitpunkt der Aufnahme als relevant eingestuft hat. Eine empirische Untersuchung zur Benutzung von Bookmarks in Webbrowsern hat bereits vor Jahren diverse Defizite für Anwender aufgezeigt (Abrams, Baecker et al. 1998). Viele Teilnehmer der Untersuchung hatten

zwar zahlreiche Bookmarks,[38] die sie aber zum Großteil nie aufgerufen hatten oder die inzwischen ungültig waren.[39] Zudem gaben mehrere Teilnehmer an, dass sie Archivierung, Kategorisierung und Verwaltung der Bookmarks als zeitaufwendig und störend empfänden. Neuere Studien bestätigen dies und weisen darauf hin, dass Web-Nutzer Bookmarks hauptsächlich für regelmäßig besuchte Seiten einsetzen (Weinreich, Obendorf et al. 2006c) und aufgrund der Benutzbarkeitsprobleme mit History und Bookmarks zunehmend andere Methoden verwenden, um Dokumente aus dem Web für längere Zeit im Zugriff zu behalten (Jones, Bruce et al. 2001; Jones, Dumais et al. 2002). Eine häufig eingesetzte Strategie ist das Reproduzieren alter Suchanfragen und früherer Pfade (Weinreich, Obendorf et al. 2006c; Obendorf, Weinreich et al. 2007; Teevan, Adar et al. 2007), wobei ihnen dieses Vorgehen ebenfalls regelmäßig Schwierigkeiten bereitet (Teevan 2004; Teevan 2007).

3.1.8 Die Bedeutung von Hyperlinks in verteilten Informationssystemen

Die in diesem Kapitel vorgestellten wesentlichen Navigationskonzepte für Informationssysteme sind jeweils für bestimmte Anwendungsbereiche gut geeignet, dennoch stellen sie aufgrund der dargestellten Schwächen für große verteilte Informationssysteme keine Alternative zu Hyperlinks dar. Insbesondere die Möglichkeit, Site-übergreifende Bezüge herzustellen und die gute Skalierbarkeit zeichnen Hyperlinks aus – das Web belegt dies mit hunderten Millionen von Sites und über einer Billion Dokumenten (vergleiche Abschnitte 3.3.4 und 7.3.2).

Sogar die für die globale Navigation im World Wide Web unentbehrliche *Suche* wäre ohne Hyperlinks und die durch sie aufgebauten Querverweise zwischen den Dokumenten unterschiedlicher Anbieter kaum realisierbar: Links werden nicht nur zur automatischen Indexierung neuer Seiten und Sites benötigt, sie sind auch für die Berechnung der *Relevanz der Suchergebnisse* ein Schlüsselelement. Darüber hinaus ergänzen sich in Hypertext-Systemen Suche und Links zu *interaktiven Suchvorgängen*: Der Benutzer versucht im ersten Schritt mit der Suchfunktion ein lediglich annäherungsweise passendes Dokument zu finden, von dem aus er in weiteren Schritten durch das Folgen assoziativer Links das Thema weiter eingrenzt, bis er exakt die gewünschte Information lokalisiert hat. Zudem stößt der Anwender dabei auf weitere geeignete Stichwörter für neue Suchanfragen, wodurch sich der intellektuelle Aufwand beim Erdenken alternativer Suchanfragen erheblich reduziert. Dieses Vorgehen ist daher insbesondere für die Suche nach ungenau definierten Zielen prädestiniert (Horton 1990: 62).

---

[38] Dies deckt sich mit dem 10. GVU User Survey, bei dem ca. 35% der Benutzer angaben, dass sie über 100 Bookmarks hätten (Kehoe, Pitkow et al. 1999).

[39] Ein weiteres Konsistenz-Problem des Webs (s. Abschnitt Konsistenz): Web-Bookmarks können leicht ungültig werden, wenn ein Objekt gelöscht oder verschoben wird. Da die gängigen Webbrowser keine automatischen Tests der Bookmarks durchführen, wird dies dem Benutzer nicht einmal gewahr.

All dies macht Links im World Wide Web zu dem am häufigsten verwendeten Navigationsmittel. Bei unterschiedlichen Langzeitstudien zur Benutzung des Webs hatte die Interaktion mit Hyperlinks jeweils einen Anteil von über 40 % der Navigationsaktionen der Teilnehmer (s. Tabelle 1 und Abb. 12; Catledge & Pitkow 1995; Tauscher & Greenberg 1997; Weinreich, Obendorf et al. 2006b). Links sind im Web sowohl zum Finden von neuen Informationen als auch zum Wiederfinden von vor längerer Zeit bereits gesichtetem Material unverzichtbar[40] (Weinreich, Obendorf et al. 2006c), und dennoch sind mit ihnen zahlreiche grundlegende Benutzbarkeitsprobleme verbunden, auf die in den nächsten Abschnitten eingegangen wird.

3.2 Grundlegende Benutzbarkeitsprobleme von Hypertext

Hypertext bietet einerseits den Vorteil, durch die Aufteilung der Informationen in viele kleine Objekte und die semantischen Querverweise zwischen den Objekten, Lesern einen schnellen und einfachen Zugriff auf weiterführende Informationen zu einem Thema zu geben (Nielsen 1993a: 154ff); s. auch Abschnitt 2.1.4). Andererseits führen die vielen kleinen Informationseinheiten und die in der Regel unüberschaubare Vernetzung zwischen ihnen zu zwei grundlegenden Benutzbarkeitsproblemen: Die *Orientierung und Navigation* in dem Netz der Dokumente wird zu einer Herausforderung, und die vielen Navigationsmöglichkeiten in den Dokumenten führen zu einer zusätzlichen Belastung des Lesers, die als *Cognitive Overhead* bezeichnet wird.

3.2.1 Orientierung und Navigation im Hyperspace

Die netzartige Struktur, die durch viele Hyperlinks aufgespannt wird, kann zwar den Zugriff auf weitere Informationen vereinfachen, sie kann aber auch leicht dazu führen, dass Anwender die *Orientierung* verlieren und sie nicht mehr wissen, *wo sie sind* und *woher sie kamen* (Vainio-Larsson 1989). Das Problem der *Desorientierung* wurde von (Conklin 1987) als das „*Lost-in-Hyperspace-Phänomen*" charakterisiert. Es wurde bereits bei frühen Hypertext-Systemen beobachtet und ließ sich in Studien auch bei Hypertext-Systemen mit einer überschaubaren Anzahl von Dokumenten nachweisen (Nielsen 1993a: 133).

Eine entscheidende Ursache für die Orientierungsprobleme der Anwender liegt in der Natur der Hyperlinks: Indem sie von jedem Dokument auf beliebige weitere Dokumente verweisen können, besitzen sie einen ähnlichen Charakter wie das in der Softwaretechnik seit Langem kritisierte „Goto"-Statement (Dijkstra 1968). Gleichsam wie ein Programmierer bei einem Quellcode mit vielen Goto-Sprüngen den Überblick verliert, hat ein Leser bei der Navigation entlang von Hyperlinks kaum eine Chance, sich einen längeren Navigationsweg zu merken (Woodhead 1991: 117).

[40] Weitere Analysen zum Navigationsverhalten von Benutzern im Web sind in diversen Publikationen des Autors dieser Arbeit zu finden (siehe Weinreich, Obendorf et al. 2006b; Weinreich, Obendorf et al. 2006c; Weinreich, Obendorf et al. 2008).

Ein weiterer Grund für die Desorientierung ist, dass der Benutzer vom Hypertext normalerweise nur einzelne Knoten – und somit einen Bruchteil der im *Hyperspace* vorhandenen Informationen – sieht. Dies ist bei gedruckten Dokumenten anders: Zeitschriften und Bücher geben dem Leser anhand physischer Eigenschaften einen Eindruck über den Umfang des Werkes und zur eigenen Position. Solche Hinweise bleiben bei digitalen Dokumenten verborgen. Ein Ansatz zur Reduzierung dieses Problems sind Karten des Hyperspace (s. Abschnitt 3.1.4), wie sie frühere Systeme wie *NoteCards* (s. Anhang B.9) und *Intermedia* (s. Anhang B.10) boten; sie erwiesen sich allerdings nur für eine lokale Übersicht als nützlich und versagten bei großen Objektzahlen (Conklin 1987; Kappe 1991).

Die Desorientierung des Benutzers im Hypertext *kann* positive, gewollte Effekte haben: Es hat sich gezeigt, dass Leser selbst bei einer gewissen Orientierungslosigkeit in Hypertext-Systemen häufig noch auf Informationen stoßen, die sie zwar eigentlich nicht erwartet oder gesucht hatten, die aber dennoch hilfreich und interessant für sie sind. Dieses Phänomen wird als *Serendipity-Effekt* bezeichnet und kann beispielsweise das explorative Lernen unterstützen (Kuhlen 1991) und zu neuen Erkenntnissen führen (Merton & Barber 2006).

Die negative Folge des *Lost-in-Hyperspace-Phänomens* ist hingegen, dass Benutzer Schwierigkeiten bei der zielgerichteten *Navigation* haben und benötigte Informationen nicht finden. Ein Schlüssel zur Reduzierung dieser Problematik liegt in den Angaben, die zur Orientierung und Navigation zur Verfügung stehen. Dabei benötigt der Leser zum einen ausreichende Metadaten zum aktuell angezeigten Dokument, anhand derer er identifizieren kann, *wo* er gerade ist und in welchem Kontext die Informationen stehen. Im World Wide Web müssen solche *Orientierungshinweise* durch die Autoren der Seiten angeboten werden, da weder Webserver noch Browser automatisch entsprechende Informationen bereitstellen. Zum anderen sollte der Anwender möglichst exakte *Navigationshinweise* bei jedem Link erhalten, sodass er erkennen kann, *wohin* sie führen. Dies bezieht sich auf die Aussagekraft der Link-Anker: *Funktion, Ziel* und *Aktion* jedes Links müssen eindeutig erkennbar sein. Auch dies ist im Web (fast) allein den Autoren überlassen (s. Abschnitte 4.4ff, 5.1.3 und 5.1.4).

Die Herausforderungen der Orientierung und Navigation in Hypertext-Informationssystemen sind bereits häufiger beschrieben und untersucht worden. Dennoch sind aufgrund der Komplexität der mit ihnen verbundenen Probleme bis heute viele Fragen ungelöst, und Benutzer werden weiterhin regelmäßig – auch im Web – mit dem *Lost-in-Hyperspace-Phänomen* konfrontiert.

3.2.2 Cognitive Overhead

Hyperlinks beeinflussen die Art, in der ein Text gelesen wird: Jeder eingebettete Link bietet dem Leser die Wahl, das Studium des aktuellen Dokuments zu unterbrechen, dem Verweis zu folgen und in einem anderen Dokument weiter zu lesen. Der Benutzer muss sich somit bei jedem Link entscheiden, ob er ihm folgen oder sich weiterhin auf das aktuelle Dokument

konzentrieren will (McKnight, Dillon et al. 1991: 6). Die hieraus folgende zusätzliche geistige Belastung des Lesers bezeichnet man als *Cognitive Overhead*.

Erstmals wurde der Cognitive Overhead bei der Verwendung von Hypertext-Systemen von (Conklin 1987: 40) als „*the additional effort and concentration necessary to maintain several tasks or trails at one time*" charakterisiert. Conklin bezog sich dabei auch auf die doppelte Herausforderung an die *Autoren* von Hypertext, die nicht nur einen Text erstellen, sondern gleichzeitig auch definieren müssen, wo sie Links einfügen wollen, wohin diese verweisen und welchen Namen und Typ sie ihnen geben.

Heute wird der Cognitive Overhead zumeist im Zusammenhang mit dem Lesen von Hypertext beschrieben (Keep, McLaughlin et al. 1993; Bieber, Vitali et al. 1997a; Witt & Tyerman 2002). Diese kognitive Überlastung[41] liegt darin begründet, dass bereits das „normale" Lesen und Erfassen von anspruchsvollem Text eine kognitive Herausforderung darstellt. Die Verarbeitungskapazität von Informationen durch den Menschen ist aber beschränkt, und so kann eine zusätzliche Beanspruchung – wie sie sich durch die eingebetteten Links ergibt – über das Potenzial des Lesers hinausgehen, ihn von der eigentlichen Aufgabe ablenken und seine Effizienz mindern.

Es gibt mehrere Ansätze zur Reduktion des Cognitive Overhead. Zwei wesentliche liegen darin, den Benutzer beim Lesen des Textes möglichst wenig abzulenken und ihm die Entscheidung zu vereinfachen, ob ein Link für ihn relevant ist oder nicht (Conklin 1987; Bernstein 1996). Folgende konkrete Maßnahmen wurden zur Reduzierung des Cognitive Overhead vorgeschlagen:

- Link-Anker sollten möglichst unauffällig hervorgehoben werden oder sind sogar nur bei Bedarf sichtbar zu machen. Auf diese Weise wird der Benutzer beim „normalen" Lesen eines Dokuments nicht durch die Verweise abgelenkt (Bernstein 1996; Weinreich, Obendorf et al. 2001; Obendorf & Weinreich 2003).
- Die Antwortzeit eines Systems ist zu minimieren, sodass der Benutzer bei irrtümlichen Navigationsschritten möglichst wenig Zeit verliert, schnell zurücknavigieren und seine vorherigen Gedanken weiterführen kann (Conklin 1987; Akscyn, McCracken et al. 1988).
- Wenn ein grafischer Browser eine Übersicht der lokalen Struktur anzeigt, kann der Benutzer erkennen, wohin die angebotenen Links führen (Conklin 1987; Utting & Yankelovich 1989; Edwards & Hardman 1999).
- Es sollte unmittelbar eine kurze Erklärung zu jedem Link in einem „*Popup-Fenster*" erscheinen (Conklin 1987). Der Leser kann so bei Bedarf mehr Details zum Link und Link-Ziel erfahren, die Auswahl der Links wird vereinfacht und unnötige Navigationsschritte

[41] Der Ausdruck Cognitive Overhead lässt sich nicht wörtlich übersetzen, da es sich um eine Metapher handelt. „*Overhead*" ist eigentlich ein betriebswirtschaftlicher Begriff, der allgemeine Betriebskosten bezeichnet („*general business expenses [as rent or heating]*", siehe: Merriam-Webster 1995).

werden vermieden (Kopetzky & Mülhäuser 1999; Weinreich & Lamersdorf 2000; Witt & Tyerman 2002).

Für das Web ist gegenwärtig keine dieser Lösungen verfügbar bzw. zum Teil sind sie sogar nur eingeschränkt realisierbar: Da es ein global verteiltes Informationssystem ist, können kurze Antwortzeiten nicht garantiert werden (s. Abschnitt Performanz 3.3.1), und zumeist liegen keine Zusatzinformationen über die Bedeutung der Links oder der Zielobjekte vor (s. typisierte Links im Web). Die benötigten Daten für eine grafische Übersicht oder die vorgeschlagenen Popup-Fenster fehlen bisher. Daher werden in dieser Arbeit Konzepte zur Übertragung und Visualisierung zusätzlicher Link-Informationen entwickelt und evaluiert (s. Abschnitte 5.5 und 6.4), die es einfacher machen sollen, die Relevanz der Links einzuschätzen, um so die Herausforderungen des Cognitive Overhead zu reduzieren.

3.3 Besondere Herausforderungen verteilter Hypertext-Informationssysteme

Abgesehen von den in den vorherigen Abschnitten diskutierten grundlegenden Benutzbarkeitsproblemen von Hypertext, gibt es bei verteilten Hypertext-Informationssystemen wie dem World Wide Web eine ganze Reihe weiterer Herausforderungen für Anwender. Diese ergeben sich aus der Verteilung der Daten auf viele Server, den potenziell großen Datenmengen und der nicht kontrollierbaren und inhomogenen Autorenschaft. Einige dieser Probleme sind zwar erst durch das Web offenbar geworden, es handelt sich aber um grundlegende Probleme großer verteilter Hypertext-Systeme, die nicht (allein) in der Architektur und den Techniken des Webs begründet sind; der Anschaulichkeit halber werden sie aber am Beispiel des Webs dargelegt.

3.3.1 Performanz

Die Performanz ist ein seit Langem bekannter Benutzbarkeitsfaktor für Hypertext-Informationssysteme. Er wurde bereits Ende der 1970er Jahre bei einem der ersten Hypertext-Systeme – dem ZOG-System der Carnegie Mellon University (s. Anhang B.4) – systematisch untersucht. ZOG war zwar kein verteiltes System, wies aber aufgrund der verwendeten Mainframe-Technik oft lange Antwortzeiten auf. In einem Test wurden drei verschiedene Systemversionen verglichen, die Reaktionszeiten von 1/10, 2 und 5 Sekunden boten. Dabei wurden klare Unterschiede in der Gebrauchstauglichkeit festgestellt, weshalb eine Antwortzeit von *maximal* zwei Sekunden empfohlen wurde. Bei längerer Wartezeit zweifelten die Tester, ob solche Hypertext-Systeme überhaupt noch ein sinnvolles Hilfsmittel darstellen könnten. Für erfahrene Benutzer, die sich schnell in dem Hypertext-System umsehen möchten, wurde sogar eine Antwortzeit von einer zehntel Sekunde als angemessen erachtet, wobei dennoch die Systemreaktion wahrnehmbar bleiben muss (Robertson, McCracken et al. 1979: 31). Etwas später formulierte dieselbe Forschergruppe, basierend auf ihren Arbeiten, „Akscyn's Law":

> Hypertext systems should take about 1/4 second to move from one place to another. If the delay is longer, people may be distracted; if the delay is much longer, people will stop using the system.

> If the delay is much shorter, people may not realize that the display has changed (Akscyn, McCracken et al. 1988).

Unterschiedliche Studien haben diese Ergebnisse für Computersysteme im Allgemeinen bestätigt: Die Antwortzeit eines Systems sollte maximal zwei bis vier Sekunden betragen, da eine geringere Performanz den Anwender ablenken kann, seinen Arbeits- und Gedankenfluss stört und zu messbar schlechteren Leistungen des Benutzers führt (z. B. Miller 1968; Shneiderman 1984; Horton 1990: 74; Shneiderman 1992: 297).

Es ist offensichtlich, dass lange Wartezeiten auf Systemausgaben von der effektiven Arbeitszeit des Benutzers abgehen und den reibungslosen Arbeitsablauf stören. Hinzu kommt, dass das Kurzzeitgedächtnis bereits nach 15-30 Sekunden anfängt, Informationen zu vergessen (Shneiderman 1992: 280). Da diese Informationen dann nicht mehr abrufbar sind und das Kurzzeitgedächtnis benötigt wird, um *Aufgaben und Probleme zu lösen* (ibid.), behindern lange Wartezeiten das Weiterführen von Gedanken und Ideen. Eine verminderte Leistungsfähigkeit und häufigere Fehler sind die Folge. Bei Hypertext kann dies konkret dazu führen, dass sich Anwender unnötigerweise im System verirren und sie vorherige Informationen und Navigationspunkte nicht mehr im Gedächtnis haben.

Herausforderungen bezüglich der verfügbaren Bandbreiten und Übertragungstechniken

Im Gegensatz zu den meisten früheren verteilten Hypertext-Systemen wie *Intermedia* (s. Anhang B.10) und *Microcosm* (s. Anhang B.13) skaliert das Web gut und ist auch auf globaler Ebene einsetzbar (Hall 1997). Dennoch ist es exemplarisch für die Geschwindigkeitsprobleme, die sich durch die globale Verteilung von Daten bei einem Hypertext-Informationssystem ergeben. So wurde die Performanz des Webs für lange Zeit als das *gravierendste* Benutzbarkeitsproblem angesehen. Zu diesem Ergebnis kam bereits der *GVU User Survey* von 1998 mit über 11700 Teilnehmern: 81 % der Befragten gaben die Transferzeiten als Benutzungsproblem Nummer eins im Web an (Kehoe, Pitkow et al. 1999).

Die Probleme in den 1990er Jahren waren vor allem auf die mangelnde Bandbreite des Internets zurückzuführen. Die Situation hat sich seitdem deutlich gebessert, zumal der Bandbreitenzuwachs anfänglich bei über 100 % pro Jahr lag und inzwischen noch bei 35 % - 60 % pro Jahr liegt (Jander 2002; Mauldin 2011). Neuere Umfragen kommen bezüglich der Performanz als Benutzbarkeitsproblem daher nicht mehr zu so derart kritischen Ergebnissen wie der GVU User Survey, dennoch bleibt die Zugriffsgeschwindigkeit für viele Anwender ein gravierendes Defizit. Eine Anfang 2011 durchgeführte Umfrage unter deutschen Internetnutzern mit über 4700 Teilnehmern ergab, dass Benutzer erst ab einer Bandbreite von ca. 3 MBit/s mehrheitlich mit der Zugriffsgeschwindigkeit zufrieden waren (Kossel & Mansmann 2011). Eine solche Bandbreite ist aber keine Selbstverständlichkeit. Zwar bieten die inzwischen recht verbreiteten Kabel-, ADSL- und VDSL-Modems zumeist ausreichende[42]

[42] Eine Erhebung zur Größe von Webseiten einschließlich der eingebundenen Grafiken ergab 2009 eine durchschnittliche Größe der Homepages der 500 am häufigsten besuchten Websites von 507kBytes (Charzinski 2010).

Transferraten von 6 Mbit/s bis 100 MBit/s, solche Zugänge sind aber nicht überall verfügbar (beispielsweise in ländlicheren Gebieten, siehe: Technologie & Rheinland 2011), für viele Personen mit inakzeptablen Zusatzkosten verbunden, und ihre Verwendung ist erheblich von den demografischen Daten der Benutzer abhängig (Demunter 2005; Kohut, Wike et al. 2007; Eurostat 2010; Seybert & Lööf 2010). Schätzungen einer Studie von Ende 2010 besagen, dass in Deutschland nur ca. 66 % bis 75% der Haushalte die Möglichkeit haben, mit einem Breitbandzugang auf das Internet zuzugreifen[43] (Cisco 2010; Eurostat 2010). In Entwicklungsländern liegt die Breitbandrate oft deutlich unter 10 % (Acharya 2010). Bei einem Internetzugang über Telefonleitungen (analog oder ISDN) beträgt die maximale Bandbreite nur 56 bis 64 kBit/s (pro Leitung), und Wartezeiten im Minutenbereich sind oft unvermeidbar (s. u.).

Erfreulicherweise werden Gegenden mit schlechter Breitbandabdeckung in Deutschland und anderen Industrieländern seit Ende 2010 zunehmend durch die neue Mobilfunktechnik LTE[44] versorgt (Bundesnetzagentur 2010), die eine Bandbreite von bis zu 50 MBit/s bietet, wobei sich die Benutzer allerdings die Bandbreite ihrer Zelle teilen müssen.

Ein weiterer bezüglich der Zugriffsgeschwindigkeit wichtiger Aspekt ist die zunehmende Bedeutung mobiler Geräte für den Zugriff auf das Web (UMTS-Report 2007; GSA 2008; Acharya 2010). Obgleich UMTS mit HSPA eine Download-Übertragungsrate von bis zu 14,4 MBit/s bietet, ist in der Praxis die Performanz mobiler Geräte meist nicht mit denen von PCs mit Festnetz-Breitbandanschluss vergleichbar.[45] Die relativ leistungsschwachen CPUs der Mobilgeräte führen zu zusätzlichen Verzögerungen und Benutzbarkeitsproblemen bei der Darstellung von Webseiten (Nielsen 2009a; Meyerovich & Bodík 2010).

Gleichzeitig hat sich die durchschnittliche Größe und Komplexität von Webseiten im letzten Jahrzehnt vervielfacht (vergl. Sullivan 1999; King 2003, Kapitel 3; King 2006; Meyerovich & Bodík 2010; King 2011). Ein wichtiger Grund für lange Wartezeiten ist insbesondere die Zunahme von Dokumenten mit zahlreichen und umfangreichen Medienelementen wie Grafiken, Flash-Animationen und Filmen. Letztere findet man inzwischen sogar für in Web-

---

Somit ist eine minimale Netto-Bandbreite von mindestens 2MBit/s nötig, um eine Übertragung dieser Dokumente in weniger als 2s zu ermöglichen.

[43] Die Anzahl der Haushalte, die Ende 2010 mit mindestens 1 MBit/s auf das Internet zugreifen könnten, ist mit 98,3% deutlich höher (Technologie & Rheinland 2011), aber nicht jeder will und kann sich die zusätzlichen Kosten leisten.

[44] LTE steht für „Long Term Evolution“ und nutzt unter anderem Frequenzbereiche, die früher für die analoge Fernsehausstrahlung reserviert waren und durch die Digitalisierung verfügbar geworden sind.

[45] HSPA („High Speed Packet Access“, ein auf UMTS basierendes Übertragungsverfahren) erreicht in der Praxis die optimale Downstream-Geschwindigkeit von 14 Mbit/s nur selten: Die Verbindungsqualität ist oft nicht optimal, da die verwendete Funktechnik im Mikrowellenbereich arbeitet und empfindlich auf physische Hindernisse reagiert. Dies reduziert automatisch die nutzbare Bandbreite. Die Abdeckung mit UMTS ist zudem keinesfalls flächendeckend, und die verfügbare Bandbreite steht immer nur allen Nutzern einer Zelle *gemeinsam* zur Verfügung (Sauter 2011). Die Nachfolgegeneration HSPA+ erreicht im Labor 42MBit/s. Es wird seit Anfang 2011 sukzessive ausgebaut, ist aber bisher nur in wenigen Gebieten tatsächlich verfügbar (siehe: http://www.telekom.com/dtag/cms/content/dt/de/989662).

seiten eingebettete Werbung (Nielsen 2010). Solche Anbieter lassen Benutzer ohne Breitbandzugang unberücksichtigt.

Defizite der Protokolle und Dienste des Webs

Genauere Analysen zeigen, dass es sogar bei schnellen PCs mit Breitbandanbindung häufig zu längeren Wartezeiten kommt (s. u.). Gründe hierfür sind nicht nur überlastete Webserver oder temporäre Engpässe in Teilbereichen des Internets, sondern auch die physikalischen Gegebenheiten des Internets und Schwächen der verwendeten Protokolle TCP/IP, DNS und HTTP:

- Beim ersten Zugriff auf eine Website muss von dem in dem URI enthaltenen *Domain-Namen* die IP-Adresse des Webservers ermittelt werden. Wenn der lokale DNS-Server die IP-Adresse nicht im Cache parat hat, muss erst der entfernte Root-Server für die „Top Level Domain" (TLD) und dann der dort angegebene „autorisierte Name Server" angefragt werden (Tanenbaum 2003). Durch die Anfragen an drei verschiedene Systeme ergibt sich so bereits im Vorfeld der eigentlichen Datenübertragung eine nicht unerhebliche Latenzzeit (s. u. und Abb. 19).
- Nach dem Bestimmen der IP-Adresse kommt es zu weiteren Verzögerungen: Im Web wird für die Übertragung von Daten HTTP (ein *verbindungsloses* Protokoll) eingesetzt, das auf das *verbindungsorientierte* TCP aufsetzt. Für jede Ressource, die der Benutzer von einer Website abruft, muss daher eine neue TCP-Verbindung aufgebaut werden (mit Ausnahmen bei HTTP 1.1, s. u.). Der TCP-Verbindungsaufbau ist aufgrund des eingesetzten *Drei-Wege-Handshakes* relativ zeitraubend, insbesondere wenn Client und Server physikalisch weit voneinander entfernt stehen.
- Hinzu kommt, dass TCP zwar das effiziente *Sliding-Window-Verfahren* für die Übertragung der Datenpakete verwendet, aber erst mittels des *Slow-Start-Algorithmus* die Fenstergröße festgelegt wird. Das ist für langlebige Verbindungen sinnvoll, aber für nur wenige Sekunden dauernde HTTP-Verbindungen ineffizient (Tanenbaum 2003).[46]
- Nach der Übertragung müssen die Daten vom Webbrowser interpretiert und dargestellt werden, wodurch es zu weiteren Verzögerungen für den Anwender kommt.

Diese Probleme wurden bereits vor Längerem erkannt. Die heute gebräuchliche Version HTTP 1.1 (Fielding, Gettys et al. 1999) versucht, ihnen Rechnung zu tragen, indem über eine TCP-Verbindung mehrere Objekte – z. B. eingebettete Bilder – sukzessiv übertragen werden können. Nach dem Transfer eines Objektes bleibt die TCP-Verbindung für einige Sekunden

[46] Es gibt mehrere Vorschläge, wie man die Probleme mit TCP reduzieren könnte. Einer kam Anfang 2010 von *Google*: Das Protokoll mit dem Namen SPDY soll als zusätzliche Schicht unterhalb von HTTP die Verbindungen zwischen Client und Server effizienter machen. Es ist aber zu bezweifeln, dass sich diese Erweiterung mittelfristig durchsetzen wird, da hierfür sowohl alle Web-Server als auch Browser angepasst werden müssten. Mehr Informationen findet man unter: http://www.chromium.org/spdy.

bestehen, falls der Benutzer weitere Ressourcen auf dem Server nutzen möchte. Zudem öffnen alle gängigen Browser mehrere TCP-Verbindungen gleichzeitig, sodass der Transfer der ersten eingebundenen Grafiken bereits aufgenommen wird, während das Hauptdokument noch in der Übertragung ist (Padmanabhan & Mogul 1995; Touch, Heidemann et al. 1998). Bei HTTP 1.1 kann weiterhin der Dateitransfer durch eine automatische Kompression der Daten optimiert werden; allerdings ist dies nur für textliche Formate wie HTML und CSS von Vorteil, nicht aber für die gängigen Grafik- und Multimedia-Dateiformate, da diese bereits effiziente Kompressionsverfahren nutzen (Fielding, Gettys et al. 1999).

Eine Studie zur Performanz des Webs beim täglichen Gebrauch

Bei einer vom Autor dieser Arbeit gemeinsam mit drei anderen Doktoranden gestalteten und durchgeführten Studie Anfang 2005 wurden die Verzögerungen beim täglichen Browsen im Web genauer untersucht. Dabei wurde mithilfe des Web-Prototyping-Frameworks *Scone* (s. Kapitel 8) und eines angepassten Firefox-Browsers die Wartezeit direkt im Browser gemessen (Weinreich, Obendorf et al. 2006a; Weinreich, Obendorf et al. 2006b; Weinreich, Obendorf et al. 2008). Während der viermonatigen Studie griffen 25 Teilnehmer insgesamt auf über 137.000 Webseiten zu. Sie verwendeten dabei unterschiedliche Internetzugänge, von einer ISDN-Einwahlleitung bis zur 100-MBit-Standleitung.

Es zeigte sich, dass die Verzögerung zwischen einer Benutzeraktion und dem Beginn der Darstellung der neuen Seite im Browser im Median bei etwa einer Sekunde lag und in über 90 % der Fälle mehr als eine halbe Sekunde verging, bis der Browser nach einer Navigationsaktion des Anwenders die ersten Datenpakete der neuen Seite erhielt und verarbeitete (s. Abb. 19, oberer Graph).

Beim Zugriff auf Ressourcen von einer bis dahin unbekannten Website war durch die zusätzlich notwendige Auflösung des Domain-Namens[47] die Latenzzeit merklich höher: Hierbei betrug sie im Median 1,7 Sekunden, in ca. 75 % der Fälle musste der Teilnehmer über eine Sekunde warten, und 10 % der Benutzeraktionen wurden nicht einmal innerhalb von 7 Sekunden beantwortet (s. Abb. 19 unterer Graph; vergl. Weinreich, Obendorf et al. 2006b).

[47] Hierbei ist anzumerken, dass die meisten Browser-Hersteller seit 2008 das sogenannte „DNS-Prefetching“ in ihre Browser integrieren. Beim DNS-Prefetching werden die Domain-Namen aller externen Links der aktuell angezeigten Seite bereits nach dem Laden der Seite aufgelöst, wodurch die hier geschilderte Problematik bei diesen Browsern in der Regel nicht mehr auftritt.

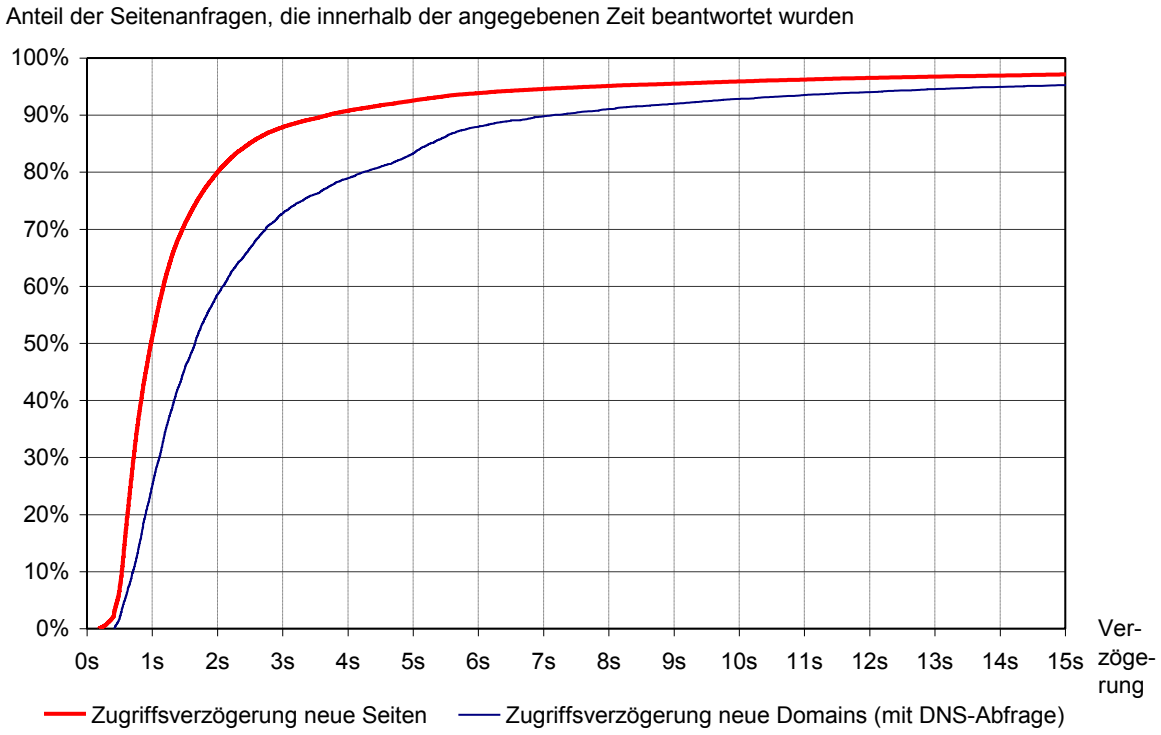


Abb. 19: Verzögerung zwischen der Navigationsaktion des Benutzers und der Verarbeitung der ersten Daten im Webbrowser (Weinreich, Obendorf et al. 2006b).

Die jeweilige Bandbreite des Internetzugangs des Teilnehmers hatte zwar eine messbare Auswirkung auf diese Verzögerung, sie war aber verhältnismäßig gering. Deutlicher waren die Effekte der Bandbreite bezüglich der *Übertragungszeit* der Dokumente zum Webbrowser, also der zusätzlichen Zeit, die zwischen dem Eintreffen des ersten Datenpaketes und dem vollständigen Laden aller eingebetteten Objekte verging. Erwartungsgemäß war für die Teilnehmer mit ISDN-Zugang die Übertagungszeit häufig unzumutbar: Beim ersten Zugriff[48] auf Dokumente im Web dauerte der Transfer in fast 60 % der Fälle über zwei Sekunden. Aber selbst für die Teilnehmer mit Breitbandzugang waren die Wartezeiten oft zu groß: Hier mussten die Benutzer bei 12 % (Standleitung mit 100 MBit/s) bzw. 34 % (ADSL mit min. 3 MBit/s) der Seiten mehr als 2 Sekunden auf die vollständige Übermittlung der Seiten warten (s. Abb. 20).

Da sich beide Zeitspannen (Latenz- und Übertragungszeit) für den Anwender addieren, sind die in *Akscyn's Law* geforderten 250 ms (s. Anfang dieses Abschnitts) im Web in aller Regel nicht einhaltbar, und sogar die im Allgemeinen für Computersysteme geforderte Obergrenze von 2 bis 4 Sekunden wird häufig überschritten.

[48] Bei den Auswertungen wurde jeweils nur der erste Zugriff des Benutzers auf eine Webseite berücksichtig, da Objekte zumeist (schnell) aus dem Browser-Cache geladen werden, wenn der Anwender innerhalb kurzer Zeit wiederholt auf sie zugreift. Auf diese Weise wurde die Verteilung der Übertragungszeiten ermittelt und nicht, mit welcher Geschwindigkeit Seiten aus dem Cache wiedergeholt werden.

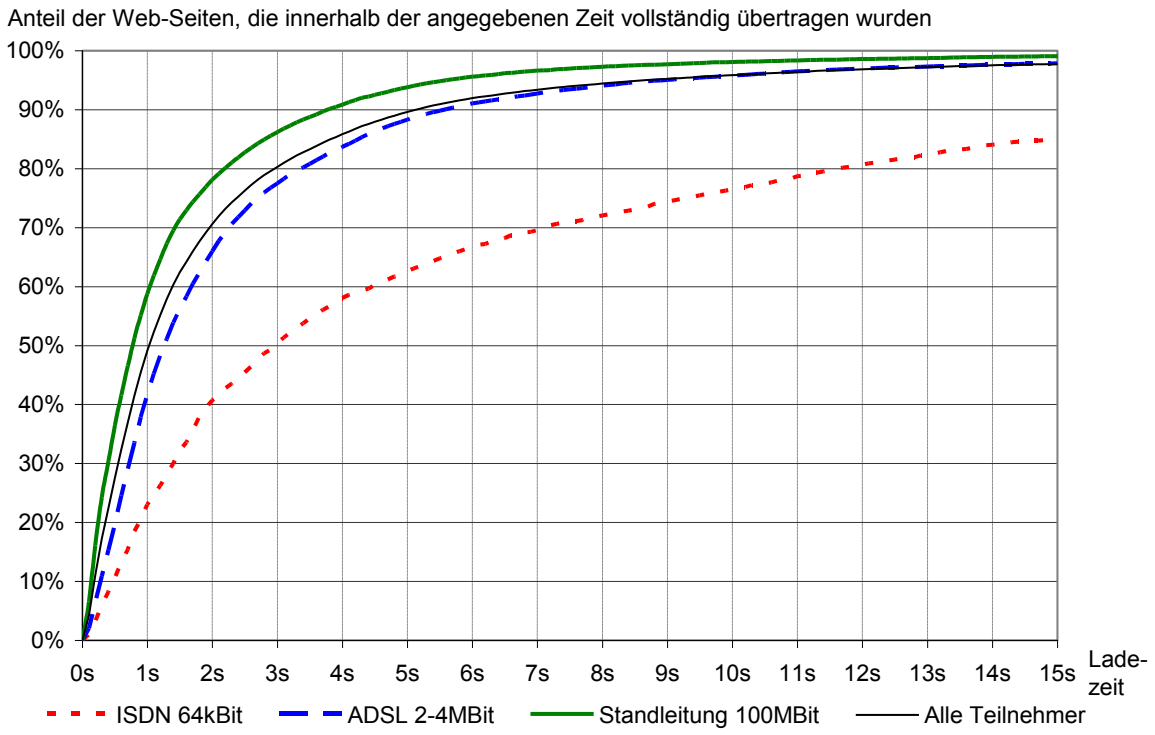


Abb. 20: Verteilung der Übertragungszeiten von Webseiten bei unterschiedlichen Bandbreiten.

Die Studie belegt gleichzeitig die stark variierenden Zugriffszeiten auf Web-Ressourcen. Obwohl dem Web-Nutzer häufig alternative Angebote mit unterschiedlich schnellem Zugriff zur Verfügung stehen (beispielsweise in der Ergebnisliste einer globalen Suchmaschine), kann er vor einer Navigationsaktion nicht erkennen, wie hoch die voraussichtliche Wartezeit ist.

Es gibt viele Richtlinien und Gestaltungshinweise, wie Autoren den Transfer ihrer Webseiten verbessern können (King 2003). Dazu gehört die Optimierung der Kompression von Grafiken (Weinreich 1997: 24ff; Siegel 1999: 33ff), die spezielle Aufteilung von Mediendateien (Zhi 2001) oder die Verwendung von CSS als gestalterische Alternative zu grafischen Elementen (Nielsen, Gettys et al. 1997).

Eine weitere Optimierungsmöglichkeit liegt im *Caching* (Zwischenspeichern) der Daten nahe dem Anwender. Dies kann regional mittels eines Proxy-Servers wie SQUID (Hamilton, Rousskov et al. 1998) oder überregional durch „*Content Distribution Networks*“ (*CDN*)[49] geschehen (s. Abschnitt 8.4.2.1). Solche Techniken reduzieren die Zugriffszeit, wenn sich ein Objekt bereits im Cache befindet, allerdings kann sich die Antwortzeit für nicht zwischengespeicherte Daten etwas verlängern, da der Proxy die Anfrage erst verarbeiten und weiterleiten muss (Glassman 1994).

[49] Ein Beispiel hierfür ist Akamai (www.akamai.com), das an Knotenpunkten des Internets Caching-Server zur Verfügung stellt, die umfangreiche Dateien großer kommerzieller Websites (z. B. Microsoft, Symatec, Apple) zwischenspeichern und so transparent die Server- und Netzwerklast reduzieren, sowie die Performanz für den Benutzer steigern.

Die in diesem Überblick aufgeführten Probleme führen dazu, dass die Performanz verteilter Hypertext-Informationssysteme wie dem Web – obwohl umfangreich untersucht – weiterhin ein kritischer Benutzbarkeitsfaktor bleibt.

3.3.2 Information Overload

Informationen galten früher als rare, wertvolle Ressource, und sie waren für die meisten Personen nur in beschränktem Maße zugänglich. In der heutigen Informationsgesellschaft ist oft das Gegenteil der Fall: Durch Medien wie Zeitschriften, Bücher, Fernsehen, Radio und das Web ist die Menge der verfügbaren Informationen so groß geworden, dass für den Einzelnen die Herausforderung besteht, *welche* der vielen Quellen für ihn von Bedeutung sind. Gleichzeitig versuchen viele Akteure, die *Aufmerksamkeit („Attention")* des Menschen[50] – als eine seiner wertvollsten Ressourcen – zu erlangen (Levy 1997). Wird die Anzahl der angebotenen Informationen zu groß, kann dies den Umgang mit ihnen erheblich erschweren und die Nützlichkeit schmälern: Man spricht von *Information Overload* (Lang & Luketick 1996; Farhoomand & Drury 2002). In einer repräsentativen Studie von Anfang 2011 beklagten knapp ein Drittel (31 %) der befragten Deutschen, dass sie sich von Informationen häufig überflutet fühlen. Weitere 30 % gaben an, zumindest manchmal so zu empfinden. Dabei gaben ältere Menschen häufiger an, unter dem Problem des Information Overload zu leiden als jüngere (BITCOM 2011).

Information Overload führt dazu, dass Benutzer vermehrt Informationen übersehen, vergessen oder durcheinanderbringen. Zeitverlust und eine verringerte Effizienz bei der Arbeit sind die Folge sowie Frustration, Müdigkeit und Stress (Farhoomand & Drury 2002).

Dieses Phänomen kann bereits im Umgang mit relativ wenigen Daten auftreten, wenn der Anwender sie nicht ausreichend überblicken kann oder wenn sie zu schnell erscheinen. Ein Beispiel hierfür bei Desktop-Anwendungen sind Pull-down-Menüs mit zu vielen (ähnlichen) Einträgen. Dies beeinträchtigt die Auswahl, da Menschen nur eine begrenzte Menge an Informationen parallel im Überblick behalten und verarbeiten können (Hiltz & Turoff 1985).

Für verteilten Hypertext bzw. das World Wide Web wird Information Overload zumeist im Zusammenhang mit der *Gesamtmenge* der zugänglichen Informationen verwendet. Sie führt dazu, dass Anwender gravierende Probleme mit der Übersicht haben und gesuchte Informationen nicht finden (Berghel 1997). Zwei Formen von globalem Information Overload lassen sich im Web unterscheiden (December 1994): *Information Pollution* und *Information Saturation*. Erstere – die *Verschmutzung* der Informationen – bezieht sich auf redundante, fehlerhafte oder veraltete Dokumente, die den Blick auf hochwertige Informationen „ver-

[50] Im Marketing wird dies häufig in dem klassischen Stufenmodell mit dem Akronym AIDA formuliert: Als erstes ist *Aufmerksamkeit (**A**ttention)* der Person zu erlangen, um sein *I**n**teresse (**I**nterest)* auf sich zu ziehen. Werbung soll danach zusätzlich *Verlangen (**D**esire)* hervorrufen und den Menschen zu einer *Aktion (**A**ction)* verleiten (Smith & Swinyard 1982).

sperren“. Die *Sättigung* bezieht sich dagegen auf die Fülle wertvoller Informationen, die aufgrund ihrer Menge viel Zeit beanspruchen, bis die relevantesten extrahiert wurden.

Eine wesentliche Ursache für diese Problematik liegt in der grundsätzlich anderen Natur des Webs gegenüber gedruckten Texten: Bücher und Magazine sind *selektiv* und *exklusiv* – jede Seite, jeder Artikel und jedes Buch hat nur eine begrenzte Anzahl an Wörtern, und der Zugriff auf weitere Informationen ist mit größerem Aufwand verbunden. Dadurch beschränken sich automatisch der Inhalt des gedruckten Textes und zumeist auch die Zielgruppe eines Werkes. Es wird zwar häufig auf andere Quellen verwiesen, aber das Verfolgen solcher Verweise ist – verglichen mit dem Folgen eines Hyperlinks – ein zeit- und arbeitsintensiver Vorgang. Hypertext im Web ist dagegen *nicht-exklusiv* und nahezu *umfassend*: Die direkt verfügbaren Informationen haben aus individueller Sicht einen praktisch unbegrenzten Umfang, da mehr Ressourcen unmittelbar zugreifbar sind, als eine Person jemals lesen könnte (Burbules 1997).

David Shenk geht noch weiter: Da das Internet mit seiner Eigendynamik und Größe für den Einzelnen nicht kontrollierbar zu sein scheint, sieht er es als eine Art *Naturgewalt* (Shenk 1997). Die Menschen müssen folglich die neue Kompetenz erlangen, mit dem *Data Smog* des Webs umzugehen, damit die vielen Informationen keinen negativen Einfluss auf sie, ihre Arbeit und ihr Sozialleben haben. Tatsächlich wurde bei einer Umfrage unter *ehemaligen* Nutzern des Internets von über der Hälfte der Teilnehmer die zu große Informationsmenge und die oft zweifelhaften Inhalte als entscheidende Gründe angegeben, es nicht mehr zu nutzen (Gerhards & Mende 2003).

Eine praktikable Lösung für den Information Overload steht bisher aus. Dabei wurde das potenzielle Problem des Information Overload als Folge globaler verteilter Hypertext-Systeme bereits 1991 – vor dem Durchbruch des Webs – im Zusammenhang mit der Konzeption von Hyper-G beschrieben. Zur Reduzierung wurde damals vorgeschlagen, dass Dokumente nach einer bestimmten Zeit (üblicherweise 6 Monate) automatisch gelöscht werden sollten, wobei das Verfallsdatum dem Autor überlassen bliebe (Kappe 1991). Obwohl Hyper-G – anders als das Web – automatisch die Konsistenz von Links und Suchindex sicherstellte, ist fraglich, wie sinnvoll diese Strategie wäre. Für einen Großteil der Dokumente scheint ein solches Vorgehen kaum praktikabel, da ihre Informationen eben *nicht* nutzlos werden, wenn sie altern.[51] Ein illustrativer Vergleich wäre ein Museum, das aus Gründen der Übersichtlichkeit die ältesten Objekte zuerst entfernt.

Gegenwärtig sind Suchmaschinen wie *Google* und *Microsoft Bing* sowie redaktionell erstellte Verzeichnisse wie *DMOZ* und *Yahoo!* die wichtigsten Hilfen, dem globalen Information Overload im Web etwas entgegenzusetzen. Insbesondere Suchmaschinen stellen ein unver-

[51] Ted Nelson (s. Anhang A.3) verfolgt in dem Projekt Xanadu (s. Anhang B.5) daher die gegenteilige Strategie: Es soll nichts gelöscht werden und das System vermerkt sämtliche Änderungen, damit immer der ursprüngliche Urheber von Ideen erkennbar bleibt – für den Information Overload ist das offensichtlich keine Lösung.

zichtbares Navigationsmittel dar, um an Dokumente zu neuen Themen zu gelangen (s. Abschnitt 3.1.1; Jansen & Spink 2006; Weinreich, Obendorf et al. 2008). Die globale Web-Suche funktioniert erstaunlich gut, wenn man recht allgemeine Informationen finden möchte, wie die Website einer bekannten Firma oder einer größeren Stadt.[52] Da die Suchmaschinen beim Erfassen der Daten nicht automatisch bewerten können, welche Qualität die Dokumente haben, liefern sie oft auch nicht die gewünschten hochwertigen Ergebnisse (Berghel 1997). Insbesondere bei der Suche nach spezielleren Dokumenten oder Diensten ist es oft eine Herausforderung, die „richtigen" Suchbegriffe zu wählen, um zu den gewünschten Ergebnissen zu gelangen.

Die Verwendung von Metadaten wird heute zumeist als vielversprechende Lösung gesehen, um der Informationsflut Herr zu werden. Metadaten können die Bedeutung der Dokumente genauer spezifizieren und Such- und Filtertechniken zu höherer Effizienz verhelfen (Lang & Luketick 1996; Farhoomand & Drury 2002). Die *Semantic Web Initiative* versucht, derartige Konzepte für das Web zu realisieren (Herman, Hawke et al. 2011). Automatische Mechanismen auf Basis von Software-Agenten sollen dabei für den Benutzer die von ihm benötigten Informationen suchen, filtern und aufbereiten (Hendler, Berners-Lee et al. 2002). Allerdings bezweifeln viele Experten die Realisierbarkeit dieser Vorhaben in absehbarer Zeit, da sie ähnliche Komplexitätsprobleme wie die einst so viel versprechende künstliche Intelligenz aufweisen (Marshall & Shipman III 2003) und Autoren Probleme bei der Definition entsprechender Metadaten haben (Cleary & Bareiss 1996).

Eine andere Möglichkeit zum Eingrenzen der Informationsmenge ist das *Collaborative Filtering* (Resnick, Iacovou et al. 1994). Solche Verfahren verwenden die Bewertungen vieler Benutzer, um aus einer Fülle von Meinungen individuelle Empfehlungen zu generieren. Dabei werden beispielsweise *Ähnlichkeiten der Benutzerprofile* zur Gewichtung der Nutzermeinungen ausgewertet. Statt komplexe Mechanismen der künstlichen Intelligenz vorauszusetzen, profitiert man von der Intelligenz anderer Benutzer. Entsprechend zu den vielen manuell erstellten Hyperlinks im Hypertext, die zu weiteren hilfreichen Informationen führen, bringen die manuell erstellten Bewertungen beim Collaborative Filtering die wichtigsten Informationen für den Einzelnen zur Geltung (s. Abschnitt 3.1.6).

3.3.3 Qualität und Vertrauen

Bei öffentlich zugänglichen, global verteilten Informationssystemen kann man auf die Dokumente, Links und Anmerkungen von unzählig vielen, häufig anonymen oder unbekannten Verfassern zugreifen. Bei diesen Ressourcen stellt sich für den Leser die Frage, ob er dem Autor *vertrauen* kann und von welcher *Qualität* die Beiträge sind. Die Bedeutung dieser Pro-

[52] Bei einer nordamerikanischen Umfrage im Jahre 2004 gaben zwei Drittel der Teilnehmer an, dass sie in der Regel spätestens bei der zweiten Suchanfrage die gesuchte Seite finden würden. Nur 4% führten an, dass sie Suchen häufiger erfolglos aufgeben würden (Hotchkiss, Garrison et al. 2004).

blematik für die Nützlichkeit und Benutzbarkeit eines Hypertext-Systems zeigt sich gut am World Wide Web.

Um als Autor im Web aktiv zu werden, benötigt man heute dank *Autorensystemen*, *Blogs*, *Wikis* und *sozialer Netzwerke* kaum noch technische Kenntnisse (s. Abschnitt 2.2.4). Es gibt zahllose kostenlose und preiswerte Angebote, sodass auch Privatpersonen beliebige Informationen unter eigener Verantwortlichkeit schnell und einfach veröffentlichen können. Das Fehlen von Kontrollinstanzen für Publikationen im Web, die minimalen Kosten sowie die Probleme der Haftbarkeit durch die Internationalität des Webs haben dazu geführt, dass es heute besorgniserregend viele Websites mit fragwürdiger oder gar krimineller Intention gibt, beispielsweise im pornografischen oder extremistischen Bereich (eco 2011). Andere Anbieter versuchen, Benutzer bewusst in die Irre zu leiten und ihnen unter Vorspiegelung falscher Tatsachen Geld abzuknöpfen oder sich anderweitige Vorteile zu verschaffen. Beispiele hierfür sind Websites, die über Sicherheitslücken in den Browsern *Trojaner*[53] oder *Adware*[54] installieren oder eine falsche Anbieteridentität vorspiegeln (beispielsweise die eines Geldinstituts), um vertrauliche Benutzerdaten zu erschleichen. Im Zusammenhang mit gefälschten E-Mails wird letzteres Vorgehen als *Phishing* bezeichnet (Kay 2004) und hat seit ca. 2004 in beunruhigendem Maße zugenommen. Hier kann von *Qualität* einer Webseite keine Rede mehr sein – der Benutzer muss primär herausfinden, inwieweit die angebotenen Informationen und Dienste authentisch sind oder sie ihm sogar potenziell schaden.

Bei der Vielgestaltigkeit dieser Problematik stellt sich die Frage, wie sich *Qualität* in einem solch großen und heterogenen Informationssystem wie dem Web überhaupt definieren lässt. Diverse Forscher haben systematisch versucht, Aspekte für die Qualität von Informationen im Web zu identifizieren. Zusammenfassend lassen sich vier primäre Faktoren abgrenzen (Katerattanakul & Siau 1999; Amento, Terveen et al. 2000; Gertz, Özsu et al. 2004):

- Die *intrinsische Qualität* beschreibt die inhaltliche Korrektheit und Exaktheit der Daten. Sie setzt voraus, dass die Informationen mit Sorgfalt zusammengestellt werden, alle relevanten Daten umfassen und aktuell sind (Gertz, Özsu et al. 2004).
- Die *kontextuelle Qualität* bezieht sich auf die Nützlichkeit und Aufgabenangemessenheit der Informationen im Zusammenhang mit den Aufgaben des Benutzers. Entscheidend hierfür ist die Relevanz der Informationen, ihr Umfang und die zeitnahe Zugriffsmöglichkeit (Katerattanakul & Siau 1999). Zudem ist es wichtig, dass Verweise auf weitere zweckmäßige Informationen verfügbar sind.

[53] Trojaner werden beispielsweise eingesetzt, um vertrauliche Daten vom PC des Anwenders zu erspähen oder Spam-E-Mails zu versenden (siehe auch Abschnitt 3.3.3: Qualität und Vertrauen im Web).

[54] Adware blendet (ungewollte) Fenster mit Werbung auf dem Computer des Benutzers ein oder verstellt seine Browser-Homepage und Bookmarks und behindert ihn so bei der Arbeit.

- Die *Qualität der Darstellung* befasst sich mit der Aufbereitung der Informationen. Dies umfasst die stilistische Qualität des Textes, seine inhaltliche Konsistenz[55] sowie die Präsentation und Gestaltung des Dokuments (Gertz, Özsu et al. 2004).
- Die *Qualität der Zugänglichkeit* ist das vierte entscheidende Kriterium. Es bezieht sich primär auf technische Faktoren, wie die syntaktische Korrektheit der Seiten und das Dateiformat. Wenn für ein Dokument eine spezielle Software benötigt wird oder es sich nicht korrekt in jedem Browser darstellen lässt, so kann es für einige Benutzer unbrauchbar sein. Ein weiterer Aspekt der Zugänglichkeit ist die Korrektheit der Links (vergl. Abschnitt 3.3.4; Katerattanakul & Siau 1999) und der schnelle Zugriff auf die Ressourcen (vergl. Abschnitt 3.3.1) – dauert der Zugriff zu lange, kann dies den Wert einer Information mindern oder sie sogar nutzlos machen (Johnson 1997a).

Werden erforderliche Qualitätskriterien nicht eingehalten, so reduziert dies die Nützlichkeit der angebotenen Daten und Dienste. Das Problem verschärft sich, wenn der Leser nicht zuverlässig feststellen kann, welche Gütemaßstäbe erfüllt werden. Genau diese Situation herrscht im Web vor, da Kontrollinstanzen fehlen, wie es sie beispielsweise bei Printmedien in Form von Verlagen und Lektoren gibt, die eine Vorauswahl treffen und eine gewisse Güte der Publikationen sicherstellen. Web-Benutzern scheint dieses Problem bewusst zu sein, denn sie trauen Informationen aus dem Internet häufig nicht: Bereits bei einer 2002 durchgeführten Umfrage der UCLA hielten über 47 % der Teilnehmer *maximal die Hälfte* der im Web angebotenen Informationen für verlässlich und präzise (Cole & al. 2003). Die im Rahmen dieser Arbeit Anfang 2003 durchgeführte Umfrage bestätigt diese Problematik (s. Abschnitt 6.3): Auf die offene Frage, welches sie als das gravierendste Benutzbarkeitsproblem des Webs ansehen, kamen qualitätsbezogene Faktoren auf den fünften und sechsten Platz: 16 der 138 Teilnehmer kritisierten die „zweifelhafte Authentizität“ und „mangelnde Authentifizierbarkeit“ der Dokumente, 14 Teilnehmer bemängelten die „vielen schlechten Dokumente des Webs“. Aufgrund der zunehmenden Kriminalität im Internet (BSI 2011) ist die Situation in den letzten 10 Jahren potenziell noch kritischer geworden.

Die mangelnde Verifizierbarkeit ist auch ursächlich dafür, dass die Qualität und Vertrauenswürdigkeit von Informationen und Diensten im Web nicht nur ein Problem der *Nützlichkeit*, sondern auch ein Problem der *Benutzbarkeit* ist. Da keine Mechanismen zur Verfügung stehen, um die Authentizität von Dokumenten sicherzustellen und Kriterien der Qualität und Vertrauenswürdigkeit zu kontrollieren, müssen sich Benutzer oft (zeit-)aufwendig absichern: Autoren sind zu überprüfen, zusätzliche stützende Quellen zu recherchieren und heuristische Kriterien einzubeziehen, wie der Gesamteindruck und der Domain-Name einer Website. Letztendlich müssen sich Benutzer häufig auf ihre Intuition verlassen.

[55] Die Konsistenz eines Textes ist ein Maßstab dafür, inwieweit die Informationen passend zusammengestellt wurden und der Autor folgerichtig argumentiert (Gertz, Özsu et al. 2004).

Bereits vor 15 Jahren wünschte sich Tim Berners-Lee einen „Oh Yeah?"-Knopf im Webbrowser, mit dem er die *Echtheit* und *Authentizität* eines Dokuments überprüfen könne (Berners-Lee 1996). Er schlug vor, dafür digitale Unterschriften zu verwenden. Dies wurde aber leider nur teilweise realisiert, denn es besteht heute lediglich die Möglichkeit, die Authentizität des *Betreibers einer Site* sicherzustellen, wenn dieser die asymmetrischen Verschlüsselungsprotokolle SSL (Secure Socket Layer) oder TLS[56] (Transport Layer Security) einsetzt (Dierks & Allen 1999): Hierfür wird ein Serverzertifikat benötigt, das von einer entsprechenden Zertifizierungsstelle (Certificate Authority) ausgestellt werden *sollte*[57] und anhand dessen der Benutzer den Anbieter einer Website ermitteln kann. Allerdings ist die Anzeige des Zertifikates in den gängigen Browsern immer noch wenig intuitiv gelöst und es lässt sich *nicht* bestimmen, welche Qualitätskriterien die angebotenen Informationen und Dienste erfüllen. Lediglich für die seit 2007 angebotenen „Extended Validation Certificates" (s. Abb. 21) bieten neuere Browser eine einfach aufzurufende Anzeige des Zertifikatsinhabers (vergl. Jackson, Simon et al. 2007). Diese Zertifikate sind aber wesentlich teurer als die bisherigen und nur für Kapitalgesellschaften erhältlich.

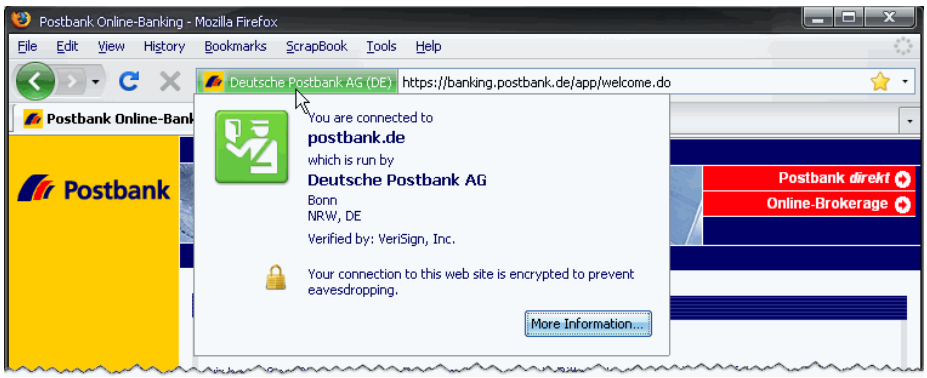


Abb. 21: Ein „Extended Validation Certificate" belegt die Identität des Betreibers einer Website.

Um dem zunehmenden Betrug im Web Herr zu werden, verwenden inzwischen alle aktuellen Browser *Phishing-Filter*, die mithilfe von Negativlisten (*Blacklists*) vor Sites mit gefälschter Identität warnen. Anbieter wie *MacAfee*,[58] *NetCraft*[59] und die Firma „*Web of Trust*" (s. Kapitel 9.2.3) bieten darüber hinaus Browser-Erweiterungen an, die für viele Sites Daten zur Vertrauenswürdigkeit (Reputation) zugänglich machen (s. Abb. 22 und Abb. 163). Darüber hinaus hat das W3C im Jahr 2010 Empfehlungen veröffentlicht, wie Browser Anwender über die Identität von Websites informieren sollen: Hierzu werden „Extended Validation Certificates" und TLS eingesetzt, und Benutzer sollen mittels „Identity Signals" auf verständliche Weise über den Anbieter einer Website informiert werden (Roessler & Saldhana 2010).

---

[56] Der TLS-Standard geht auf das SSL-Protokoll von Netscape Communications Corporation zurück, Version 1.0 von TLS entspricht SSL 3.1. Das Protokoll HTTPS verwendet TLS, um einen sicheren Tunnel zwischen Client und Server aufzubauen.

[57] Es ist auch möglich, ein solches Zertifikat selbst zu erstellen, es erfolgt dann aber eine Warnmeldung im Webbrowser, dass der Aussteller des Zertifikates nicht vertrauenswürdig ist.

[58] Der McAfee *SiteAdvisor* wird unter der Adresse http://www.siteadvisor.com/ angeboten.

[59] Die *NetCraft Toolbar* ist unter http://toolbar.netcraft.com/ erhältlich.

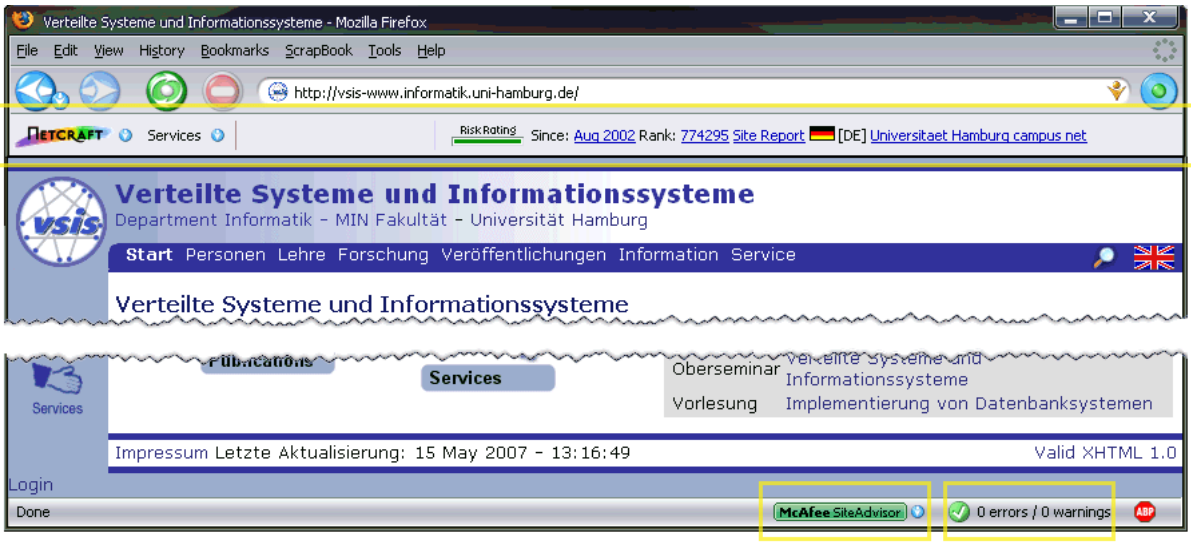


Abb. 22: Der Firefox-Browser mit den Extensions NetCraft Toolbar (oben), McAfee SiteAdvisor und HTML-Validator (beide rechts unten im Statusbereich).

Trotz dieser Bestrebungen zur Sicherstellung der Vertrauenswürdigkeit von Web-Ressourcen bleiben bis heute viele Qualitätskriterien kaum automatisch überprüfbar. Lediglich das Qualitätskriterium der *Zugänglichkeit* der Informationen ist maschinell validierbar, da es sich primär auf syntaktische und technische Eigenschaften bezieht. So lässt sich beispielsweise prüfen, ob der HTML-Code korrekt ist (z. B. durch die Firefox-Erweiterung *HTML-Validator*, s. Abb. 22), die *Web Content Accessibility Guidelines*[60] eingehalten werden (Chisholm, Vanderheiden et al. 1999) und ob alle Links fehlerfrei sind.

Für die drei anderen aufgeführten Qualitätskriterien – intrinsische, kontextuelle und darstellerische Qualität – ist eine programmatische Überprüfung problematisch. Ansatzweise gelingt dies mit Methoden, die statistische Eigenschaften einer Site auswerten und die Link-Struktur des Webs analysieren. In einer Studie erwiesen sich unter anderem Googles *PageRank*- und Kleinbergs *Authority*-Algorithmus (Page, Brin et al. 1998; Kleinberg 1999) – zwei Verfahren, die auf einer Auswertung der *Link-Struktur* beruhen – als relativ zuverlässige Indikatoren für die Qualität von Webseiten. Zudem korrelieren auch einfachere Faktoren, wie die Anzahl der Seiten in einer Website oder die Größe der Seiten, häufig mit der Qualität der angebotenen Informationen (Amento, Terveen et al. 2000; Ivory, Sinha et al. 2001; Ivory & Megraw 2005). Allerdings sind solche Eigenschaften bei Kenntnis der Kriterien entsprechend einfach manipulierbar.

Alternativ bieten sich (ähnlich wie zur Reduzierung des *Information Overload*, s. Abschnitt 3.3.2) kollaborative Verfahren an (s. Abschnitt 3.1.6): Die Bewertungen und Kommentare anderer Anwender können Hinweise auf die Güte von Webseiten geben (Baier, Weinreich et al. 2004). Die *kontextuelle Qualität* eines Dokuments im Web lässt sich dagegen selbst mit

[60] Der Dienst Bobby kann Seiten nach den WAI-Guidelines des W3C und den „Section 508 Standards“ – einem bindenden Bundesstandard der USA – überprüfen: http://bobby.watchfire.com/.

solchen Methoden kaum ermitteln, da sie zu sehr von situativen Faktoren wie der aktuellen Aufgabe des Benutzers abhängig sind.

Aus Benutzersicht wäre es sinnvoll, derartige Funktionen nahtlos in die Schnittstelle des Webs zu integrieren. Auf diese Weise könnte die Güte des aktuellen Dokuments schneller und einfacher eingeschätzt werden. Zusätzliche Vorabinformationen bei den Links über die Qualität und Vertrauenswürdigkeit des Verweiszieles würden darüber hinaus helfen, unnötige Navigationsschritte zu minderwertigen oder potenziell gefährlichen Seiten zu vermeiden.

3.3.4 Konsistenz

Die Konsistenz von Daten in verteilten Softwaresystemen ist auf den ersten Blick ein technisches Problem und keines der Benutzbarkeit. Allerdings kann mangelnde Konsistenz die Benutzbarkeit gravierend beeinträchtigen, wenn der Anwender mit den Folgen konfrontiert wird. Im Web ist diese anachronistische Situation bis heute gegeben, da es keine Mechanismen gibt, die automatisch die Konsistenz sicherstellen; der Benutzer wird stattdessen häufig mit wenig hilfreichen Fehlermeldungen konfrontiert. Dabei wurde diese Problematik schon vor der Etablierung des Webs als entscheidende technische Herausforderung für große verteilte Hypertext-Informationssysteme identifiziert (s. Kappe 1991). Die zwei im Folgenden diskutierten Kernaspekte der Konsistenz im Web sind die *Datenbankaktualität von serverübergreifenden Suchsystemen* und die *Korrektheit externer (also zwischen zwei Sites verweisender) Hyperlinks*.

Die globalen Suchmaschinen des World Wide Webs stehen vor dem Problem, dass sie auf einem lokalen Index und damit immer auf einem veralteten Abbild des Webs operieren. Dies ergibt sich aus der Methode zur Erstellung des Suchindex: Crawler durchkämmen das Web anhand der Hyperlinks, um die vorhandenen Seiten sukzessive zu erfassen und in den Index aufzunehmen. Aufgrund der großen Menge an Ressourcen,[61] der Verteilung der Daten auf mehrere hundert Millionen Sites (Netcraft 2011) und der endlichen Bandbreite der Netzwerkverbindungen, ist das Erstellen eines globalen Suchindexes zwangsweise ein zeitintensiver Vorgang (Kappe 1991). Als Folge sind die Dokumente im Index globaler Suchmaschinen häufig einige Tage bis Wochen alt (Notess 2003). Alternde Daten führen zur *Inkonsistenz zwischen Suchindex und Webseiten*. Eine Studie von 2000 ermittelte je nach Suchmaschine zwischen 2 % und 14 % veraltete Suchausgaben (Notess 2000). Eine Reduzierung des Indexierungsintervalles und „intelligentere“ Algorithmen zur Berechnung potenziell veralteter Daten haben in den Folgejahren zu durchschnittlich aktuelleren – und damit konsistenteren – Suchdatenbanken geführt (Lewandowski, Wahlig et al. 2006). Dennoch bleibt das grundsätzliche Problem bestehen, zumal die Größe des Webs stetig wächst.

[61] Googles Web-Crawler verarbeiten nach eigenen Angaben regelmäßig über eine Billion Webseiten (Stand vom August 2008): http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html.

Das Konsistenzproblem von Suchmaschinen ließe sich auf unterschiedliche Weise weiter reduzieren:

- Um das Indexieren zu beschleunigen, kann beim Erfassen des Webs verteilt vorgegangen werden. Die Open-Source-Projekte *YaCy* und *Sciencenet* versuchen, das Web verteilt mithilfe der Computer von Tausenden von Web-Nutzern zu indexieren und so Bandbreitenbeschränkungen zu umgehen (s. 3.1.1).
- Das Suchsystem *Harvest* verwendet ebenfalls eine verteilte Strategie: Die Seiten werden lokal beim Webserver indexiert und dem Suchsystem in bereits vorbereiteter und komprimierter Form angeboten. Dabei ist es auch möglich, nur die Änderungen seit der letzten lokalen Indexierung zu übertragen (Bowman, Danzig et al. 1995).
- Zusätzlich können alle indexierten Dokumente vom Suchsystem gespiegelt angeboten werden, damit der Benutzer eine Ausweichmöglichkeit erhält, falls das Original nicht mehr zugreifbar ist.[62]

Das ***Konsistenzproblem der Links*** im Web hat ähnliche Ursachen wie das der Suchsysteme. Fehlerhafte Links werden primär dadurch verursacht, dass Ressourcen im Web beliebig geändert oder gelöscht werden können, ohne dass der Autor eines darauf verweisenden Links darüber eine Nachricht erhält. Zusätzlich können sich *semantisch* fehlerhafte Links ergeben, wenn ein Zielobjekt geändert wird und dadurch die Intention des Links nicht mehr zutrifft (Verbyla 1999). Studien haben gezeigt, dass defekte Links und die damit verbundenen Fehlermeldungen nicht nur die Benutzbarkeit verschlechtern und die Navigation erschweren können, sondern viele Web-Angebote – beispielsweise Online-Kurse – sogar unbrauchbar machen (Markwell & Brooks 2003).

Mehrere Untersuchungen haben sich mit dieser Problematik beschäftigt. Eine dreijährige Langzeitstudie von 1996 bis 1999 untersuchte die Dynamik des Webs (Koehler 1999): Es zeigte sich, dass bereits nach einem Jahr nahezu 30 % der Webseiten nicht mehr zugreifbar waren. Innerhalb dieser Zeit erfuhren sogar über 97 % aller Seiten irgendeine Art der Änderung, beispielsweise eine Aktualisierung des Textes, Verbesserungen am Design oder Korrekturen an den eingebetteten Links. Zu einem ähnlichen Ergebnis kamen 2002 Biochemiker der University of Nebraska (Markwell & Brooks 2003). Sie versuchten, die „Halbwertszeit" von Links zu bestimmen. Ihre Untersuchung stützte sich auf insgesamt 515 externe Links dreier Biochemie-Online-Kurse, die hauptsächlich auf im Web frei verfügbare Informationen verwiesen. Nach 24 Monaten (im August 2002) waren 142 der Links fehlerhaft, das entspricht einem „Schwund" von 27,5 Prozent. Die Zahl der funktionierenden Links nahm während des Untersuchungszeitraums kontinuierlich ab, wobei die Ergebnisse der monatlichen Stichproben mit physikalischen Prozessen wie radioaktivem Zerfall vergleichbar waren. Eine Extrapolation der Daten ergab eine „Halbwertszeit" von ca. 55 Mona-

[62] Google bietet solche gespeicherten Seiten an, allerdings ohne die eingebetteten Objekte (z.B. Grafiken). Der automatische Zugriff auf solche Kopien als „Fallback" ist aber nicht in heutige Browser integriert.

ten – nach dieser Zeit halbierte sich statistisch die Anzahl intakter Links. Für Hyperlinks zu „.com"-Adressen war die Zeit sogar noch wesentlich kürzer: Bereits nach 24 Monaten waren im Mittel 48 % der Seiten nicht mehr verfügbar (Markwell & Brooks 2003).

Der Anteil fehlerhafter Links im Web scheint zu fallen, wenn auch nur langsam: Eine Studie von 1999 ergab, dass 5,7 % der Links im Web schadhaft sind, (Sullivan 1999), Anfang 2008 (9 Jahre später) ergaben Stichproben immer noch einen Anteil von über 3,2 % und Ende 2011 lag der Wert bei ca. 2,5% (LinkQuality.com 2011). Es ist allerdings zu bedenken, dass diese Statistiken keine semantisch fehlerhaften Links (s. o.) berücksichtigen und viele Websites inzwischen statt eines Fehlercodes eine Übersichts- oder Suchseite präsentieren, wenn eine Ressource nicht verfügbar ist. So werden zwar Fehlermeldungen vermieden, aber der Benutzer gelangt über den Link dennoch nicht zur erwarteten Information.

Aufgrund der Signifikanz des Problems wurden bereits mehrere Strategien zur Gewährleistung der Konsistenz von Links großen verteilten Informationssystemen vorgestellt:

- Eine Möglichkeit besteht darin, das Löschen von Dokumenten im gesamten System[63] zu verbieten; folglich können Links nicht mehr veralten. Allerdings müsste dann das System auch die *Versionierung* der Dokumente unterstützen, damit man Daten aktualisieren kann, ohne dass es zu *semantisch* fehlerhaften Links kommt. Dieses Konzept sollte bei Xanadu (s. Anhang B.5) eingesetzt werden, es wurde aber nie implementiert (Nelson 1974). Einige Experten sehen es auch für das Web als empfehlenswert an, Seiten nie zu löschen (Nielsen 1998b). Dennoch ist diese Strategie im Web kaum durchsetzbar, da es bedeuten würde, allen Anbietern einen derartigen Umgang mit ihren Web-Ressourcen vorzuschreiben.
- Eine weitere Lösung sind *dynamische Links*, deren Ziel erst bei Anwahl des Link-Ankers berechnet wird. Dieser Ansatz eignet sich vornehmlich für *generische* Links, deren Ziel sich aus dem Text des Link-Ankers oder bestimmten Link-Attributen ableitet. Diese Technik wurde beispielsweise in Microcosm/DLS realisiert (s. Anhang B.13). Dynamische Links können zudem die Link-Erstellung vereinfachen und Speicherplatz sparen (Kappe 1991). Allerdings setzen sie *Link-Server* voraus (vergleiche Abschnitt 2.2.2), die das Verweisziel berechnen. Bei der aktuellen Infrastruktur des Webs würde sich das Konsistenzproblem somit lediglich auf diese Systeme verlagern, da sie ähnlich den Web-Suchmaschinen auf einer eigenen Datenbank operieren müssten.[64]
- Alternativ bietet sich eine *automatische Kontrolle* der Links an. Eine unmittelbare Feststellung fehlerhafter Links setzt jedoch voraus, dass jede Änderung an einem Dokument den Anbietern aller eingehenden Links automatisch mitgeteilt wird. Es gibt eine Reihe von Lösungsvorschlägen für die technische Realisierung dieser Strategie, beispielsweise

[63] Im Falle des Webs entspräche dies dem *gesamten* Web.

[64] Es ließen sich auch Suchmaschinen wie Google zur Berechnung von Link-Zielen einsetzen. Für das Web würde laut einer Untersuchung bereits die Angabe von 5 „gut gewählten" Stichwörtern reichen, um Link-Ziele eindeutig zu spezifizieren (Phelps & Wilensky 2000).

den *Atlas Link Service* für das Web (Pitkow & Jones 1996), das *W3Objects-Referenzierungssystem* (Ingham, Caughey et al. 1996) oder die Link-Datenbanken von *Hyper-G*, die einen „Flooding-Algorithmus" einsetzen (Kappe 1995). Bei solchen Verfahren werden Änderungen an den Dokumenten zwischen den Servern übermittelt, sodass fehlerhafte Links automatisch repariert oder gelöscht werden können (Davis 1999). Das automatische Entfernen von Links birgt indessen neue Benutzbarkeitsprobleme in sich, wenn damit wichtige Navigationsmöglichkeiten verloren gehen, die der Anwender für bestimmte Elemente eines Dokuments erwartet.

Da keine dieser Techniken bis heute im Web einsetzbar ist, stellt diese Arbeit in Kapitel 5ff neue Konzepte zur Reduktion der Benutzbarkeitsdefizite durch fehlerhafte Links vor, die die Nachteile obiger Verfahren vermeiden und für eine Integration in die aktuelle Infrastruktur des Webs optimiert sind. Jeder Webserver verfügt dabei über eine zusätzliche Komponente, die automatisch die Links in allen lokalen Ressourcen der Site überprüft. Defekte Links werden aber nicht gelöscht, sondern in einer Datenbank mit Link-Metadaten vermerkt.[65] Bei Abruf eines Dokuments wird von der Komponente der aktuelle Status aller Links der Seite an den Browser übermittelt. Eine Browser-Erweiterung warnt den Benutzer vor fehlerhaften Links (siehe Abschnitt 4.5.4.1) und er kann dann selbst entscheiden, wie er mit dem Problem umgehen will (Weinreich & Lamersdorf 2000). Zudem könnte so der Browser für defekte Links bereits vor der Navigationsaktion des Anwenders systematisch die Verfügbarkeit alternativer Link-Ziele ermitteln und beispielsweise auf eine Kopie der Seite aus dem *Google-Cache* oder dem *„Internet Archive"*[66] verweisen (s. Kapitel 5ff).

3.4 Die Rolle von Links für die Benutzbarkeit verteilter Hypertext-Systeme

In den vorhergehenden Abschnitten wurden die wesentlichen Probleme im Umgang mit verteilten Hypertext-Informationssystemen analysiert. Da Hyperlinks für die Interaktion mit den Systemen eine zentrale Rolle spielen, hat ihre Benutzungsschnittstelle auch eine wesentliche Bedeutung für die Reduzierung dieser Probleme:

- Eine einfache *Orientierung* und *Navigation* (s. Abschnitt 3.2.1) setzen nicht nur voraus, dass dem Leser für jedes Dokument angemessene Informationen über Anbieter und Inhalt zur Verfügung stehen und er immer Gewissheit hat, *wo* er gerade ist; er muss auch für alle Links eindeutig erkennen können, *wohin* sie führen, also was ihn als Link-Ziel erwartet. Nur so lässt sich vermeiden, dass er irrtümlich missverständliche Links ausgewählt und die Orientierung weiter erschwert wird.
- Beim *Cognitive Overhead* (s. Abschnitt 3.2.2) geht die Bedeutung der Hyperlinks noch weiter: Da sie die Verursacher des Phänomens sind, stellen sie auch das Schlüsselelement

[65] Mithilfe dieser Datenbank kann auch der Anbieter jederzeit den Status aller Links seiner Site einsehen. Dies vereinfacht die Sicherstellung der Link-Konsistenz.

[66] Das „Internet Archive" ist eine digitale Bibliothek, in der regelmäßig unzählige Seiten des Webs archiviert werden. Mithilfe der „Wayback Machine" ist der Zugriff auf alte Web-Dokumente bis hin zum Jahre 1996 möglich.

zu seiner Reduzierung dar. Hierfür sind teilweise diametrale Ziele zu vereinen: Einerseits sollen die Link-Marker beim Lesen möglichst wenig beeinträchtigen und ablenken, andererseits müssen sie eindeutig auszumachen sein. Zudem muss der Leser aufgrund der verfügbaren Informationen effizient und ohne besondere geistige Anstrengung entscheiden können, welche Links für ihn berücksichtigenswert sind und welche nicht. Folgt ein Benutzer einem Link und erfüllt das Zieldokument die Erwartungen nicht, so verliert er Zeit, wird von seinem Gedankengang abgebracht und seine kognitive Belastung wird weiter erhöht.

- Die Bandbreiten des Internets und der Zugänge werden zwar sukzessive erhöht, aber *Antwortzeiten*, die ergonomischen Anforderungen genügen, können für den globalen Zugriff auf Web-Ressourcen bis heute nicht sichergestellt werden (s. Abschnitt 3.3.1). Die daraus folgenden Defizite bei der Benutzung ließen sich begrenzen, wenn Anwender auf längere Verzögerungen hingewiesen werden, *bevor* sie einen Link auswählen. Dies gibt ihnen die Möglichkeit, selbst zu entscheiden, ob sie eine gewisse Wartezeit in Kauf nehmen wollen oder gegebenenfalls eine Alternative mit schnellerem Zugriff bevorzugen.
- Zwei Aspekte des *Information Overload* (s. Abschnitt 3.3.2) beziehen sich auf die Benutzungsschnittstelle von Hyperlinks: Befinden sich auf einer Webseite sehr viele Links und sind diese nicht eindeutig verständlich, so kann es zu einer *lokalen* Informationsüberflutung des Anwenders führen. Dieses Problem ist durch verständliche Hinweise zur Bedeutung jedes Links reduzierbar (Spool, Scanlon et al. 1998; Nielsen 1999a). Gleichzeitig wird der Umgang mit dem *globalen* Information Overload vereinfacht, wenn der Leser problemlos alle Links zu irrelevanten Dokumenten ausgrenzen kann und sich so die Anzahl der zu berücksichtigenden Ressourcen verringert.
- Menschen sollte es grundsätzlich rasch und zuverlässig möglich sein, die *Qualität und Vertrauenswürdigkeit* von Dokumenten, Angeboten und Diensten zu überprüfen (s. Abschnitt 3.3.3). Dies ist bei vielen der im Internet verfügbaren Ressourcen nicht gegeben, und bereits der Zugriff auf sie (z. B. das *Anzeigen* einer Webseite im Browser) kann ein Sicherheitsrisiko darstellen.[67] Der Anwender sollte somit nicht nur das aktuell angezeigte – und somit bereits zu seinem Computer übertragene – Dokument kontrollieren können, sondern auch entsprechende Hinweise über die Objekte zur Verfügung haben, auf die er als Nächstes zugreifen kann. Wenn er bereits vor einer Navigationsaktion vor zweifelhaften Ressourcen gewarnt wird, kann er die Übertragung bedenklicher oder schädlicher Dokumente auf seinen Computer vermeiden.
- Die Konsistenz von Links stellt eine der größten Herausforderungen für global verteilte, offene Hypertext-Systeme dar (s. Abschnitt 3.3.4). Die negativen Auswirkungen inkonsistenter Links auf die Benutzbarkeit lassen sich vermindern, wenn fehlerhafte Links

[67] Es gibt regelmäßig neue Würmer, Trojaner und Spyware-Programme, die sich durch Fehler und Sicherheitslücken in den Browsern bereits auf dem Computer des Benutzers installieren, sobald er die Seite aufruft.

bereits vor der Auswahl identifizierbar sind und dem Benutzer Alternativen angeboten werden, wie eine gespiegelte Version des Zielobjektes.

Zusammenfassend zeigt dies die große Bedeutung der Benutzungsschnittstelle von Links für die Gebrauchstauglichkeit verteilter Hypertext-Informationssysteme. Diese Zusammenhänge zwischen Hyperlinks und den Kernproblemen verteilter Hypertext-Informationssysteme gelten für das World Wide Web im besonderen Maße, da hier Links das bedeutendste Navigationsmittel darstellen (s. Abschnitt 3.1.8) und viele andere Navigationsmöglichkeiten wie die Ergebnislisten von Suchmaschinen oder die hierarchische Navigation ebenfalls auf HTML-Links basieren. Die folgenden Kapitel beschäftigen sich daher mit der Schnittstelle von Hyperlinks und den Möglichkeiten zur Reduzierung der mit ihnen verbundenen Benutzbarkeitsprobleme.

4 Typisierung als Grundlage benutzbarer Hyperlinks

In diesem Kapitel werden Konzepte zur Verbesserung der Benutzbarkeit von Links in verteilten Hypertext-Informationssystemen untersucht und weiterentwickelt. Der erste Abschnitt 4.1 beschreibt typisierte Links und Link-Previews als wesentliche Ansätze zur Reduktion der Probleme bei der Link-Navigation und zur Verminderung des Cognitive Overhead. Abschnitt 4.2 befasst sich mit den theoretischen Grundlagen typisierter Links und stellt drei unterschiedliche Methoden zur Typisierung von Links vor. Eine Analyse der Grenzen der Link-Typisierung für Hypertext-Leser und -Autoren folgt in Kapitel 4.3.

In Abschnitt 4.4 werden die gegenwärtig verfügbaren Techniken zur Verknüpfung von Objekten im Web analysiert. Dabei wird auf die Ausdrucksformen zur *expliziten* Link-Typisierung in HTML eingegangen, in welchem Maße diese Daten gegenwärtig eingesetzt werden und welche Potenziale und Grenzen sie bieten.

Die Defizite bei der Typisierung von Links in HTML sind Motivation für eine systematische Klassifikation der *impliziten* Eigenschaften von Links und Objekten im Web (Abschnitt 4.5). Diese Klassifikation berücksichtigt, wie sich dieInformationen bereitstellen und nutzen lassen, und dient als Grundlage für die in den folgenden Kapiteln 5, 6 und 7 vorgestellten und evaluierten Konzepte zur Darstellung erweiterter Link-Informationen.

4.1 Konzepte zur Verbesserung der Benutzbarkeit von Links

Das vorherige Kapitel 3 hat gezeigt, dass Hyperlinks in Informationssystemen nicht nur den Zugang zu digital gespeicherten Daten vereinfachen können, sondern dass sie auch im Zusammenhang mit mehreren Benutzbarkeitsproblemen stehen. Um diese Probleme zu reduzieren, müssen Anwender schnell und sicher erkennen können, welche Bedeutung und Eigenschaften Links haben und wohin sie führen (Smith, Newman et al. 1997).

Ein typisches Verhalten bei der Benutzung von Hypertext ist die sogenannte *Hub-and-Spoke-Navigation*. Sie bezeichnet ein Aktionsmuster, bei dem der Benutzer von einer zentralen Navigationsseite aus – „Hub“ (Radnabe) genannt – nacheinander mehreren Links folgt und sich die Zieldokumente – „Spokes“ (Speichen) – ansieht (s. Abb. 23), um dazwischen immer wieder zur Ausgangsseite zurückzukehren (Catledge & Pitkow 1995).

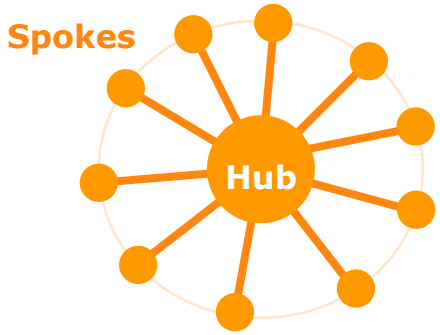


Abb. 23: Illustration der Hub-and-Spoke-Navigationsmetapher.

Die Navigationsschritte können beabsichtigt sein, beispielsweise wenn ein Benutzer von der Übersichtsseite eines Nachrichtendienstes aus mehrere Artikel lesen will. Häufig sind sie aber Ausdruck von irrtümlich gewählten Links, wobei der „Back Button“ (s. Abschnitt 3.1.7) im Sinne eines „Undo“ verwendet wird und den Leser zur letzten nützlichen Seite zurückführt.

Die Problematik der Hub-and-Spoke-Navigation war bereits Gegenstand zahlreicher Forschungsprojekte (Catledge & Pitkow 1995; Tauscher 1996; Cockburn, Greenberg et al. 2003). Es gibt zwei wesentliche Konzepte, den Anteil missverständlicher Links zu reduzieren: *Link-Previews* (Catlin, Bush et al. 1989) und *typisierte Links* (Trigg 1983). Beide Ansätze dienen dazu, *distale* Informationen *proximal* darzustellen, also dem Leser bereits *beim Link-Anker* mitzuteilen, was ihn (entfernt) „am anderen Ende“ des Links erwartet.

4.1.1 Link-Previews

Link-Previews präsentieren ausgewählte Informationen zum Ziel des Links (Catlin, Bush et al. 1989). Sie können je nach Anforderung unterschiedlich detailliert sein: Dies reicht vom *Titel* des Zielobjekts über eine *Beschreibung des Inhalts* bis hin zu längeren charakteristischen Textausschnitten. Es können Typinformationen zum Link-Ziel angeboten werden, beispielsweise zum Dateityp oder zur Ausführlichkeit des Dokuments. Bei grafischen Objekten wird teilweise auch eine verkleinerte Vorschau in Form von Thumbnails eingesetzt. Der Leser erfährt auf diese Weise mehr über den Link, ohne ihm folgen und möglicherweise zur Ausgangsseite zurückkehren zu müssen (vergl. Cockburn & Jones 1997).

Eines der ersten Systeme, das *Link-Previews* realisierte, war Intermedia (s. Anhang B.10): Der sogenannte *Web View* sollte den *Cognitive Overhead* reduzieren und die *Orientierung* vereinfachen, indem er die Navigation für den Benutzer „vorhersagbar“ machte. Es wurde eine Übersichtskarte (Abb. 24) verwendet, die Titel und Typ aller Zielobjekte der aktuellen Ressource darstellte:

> „Intermedia's new Web View enhances system predictability by providing at least two clues of where a link is headed: the type of document and its name” (Utting & Yankelovich 1989: 83).

Bei Benutzbarkeitstests mit Intermedia wurde von den Teilnehmern der *Web View* als wichtige Navigationshilfe lobend hervorgehoben (Utting & Yankelovich 1989).

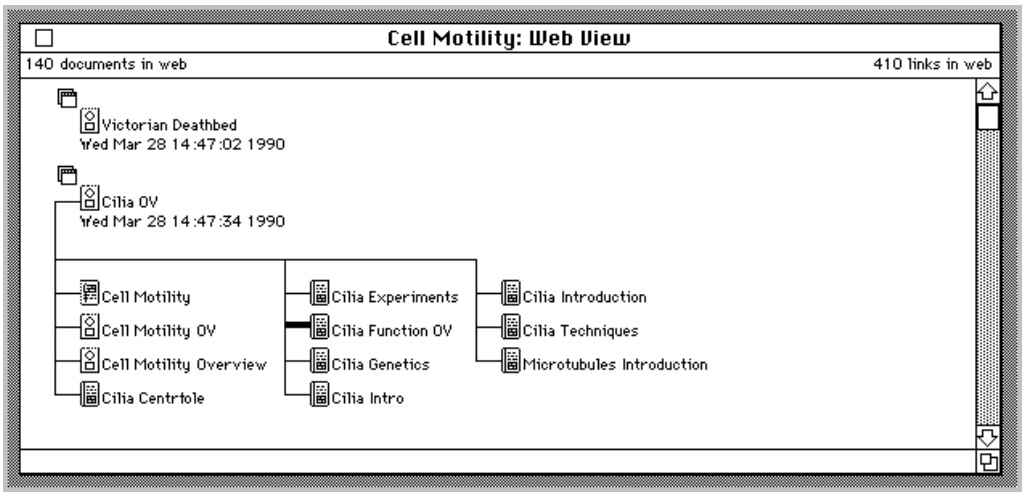


Abb. 24: Intermedia *Web View*: Oberhalb des aktuellen Dokuments („Cilia OV") befindet sich die Personal History des Benutzers, unterhalb die Link-Vorschau. Der schwarze Balken vor „Cilia Function OV" hebt den angewählten Link[68] hervor (nach Utting & Yankelovich 1989).

HyperTies (s. Anhang B.8) bot als Link-Preview den Titel des Zielobjekts des gerade aktiven Links in der untersten Zeile der aktuellen Karte an (s. Abb. 180 und Abb. 181), und im World Wide Web kann der Benutzer (meistens) den URI eines Link-Zieles im Statusbereich des Browsers sehen, wenn er mit dem Mauszeiger über einen Link-Anker fährt (s. Abschnitt 5.3.5 und Abb. 55 sowie Abschnitt 9.2). Abgesehen hiervon und wenigen Studien zu Link-Previews im Web (Noirhomme-Fraiture & Serpe 1998) blieb dieses Konzept in der Hypertext-Forschung und Praxis relativ wenig berücksichtigt.

4.1.2 Typisierte Links

Das zweite Konzept zur Reduzierung der Benutzbarkeitsprobleme mit Hyperlinks ist die *Typisierung*. Ein *Link-Typ* drückt die *semantische Relation* zwischen dem Start und dem Ziel eines Links aus (Trigg 1983). Link-Typen bieten gegenüber Link-Previews den Vorteil, dass sie den Kontext der aktuellen Seite berücksichtigen und so präziser die Verknüpfung beschreiben können. Dadurch lassen sich Probleme bei der Verständlichkeit der Links vermeiden, die bei Link-Previews auftreten können.

Typisierte Links wurden von vielen Experten als sehr bedeutsam für benutzbaren Hypertext angesehen (Nanard & Nanard 1993; Baron, Tague-Sutcliffe et al. 1996; Bieber, Vitali et al. 1997a; Nielsen 1998a; Kopak 1999). Die Entwickler von *Sepia* gingen aufgrund ihrer Erfahrungen sogar so weit, dass sie typisierte Links als erstes *Design-Prinzip* für benutzbares Hypermedia-Design anführten (Thüring, Hannemann et al. 1995).

Ein wesentlicher Vorteil typisierter Links ist die Verbesserung der *sprachlichen Kohärenz* eines Hypertextes. Kohärenz ist ein zentraler Begriff der Textlinguistik und beschreibt den *inhalt-*

[68] Intermedia unterschied zwischen dem **Anwählen** und dem **Folgen** eines Links. Mittels eines einfachen Mausklicks wurde ein Link *markiert* bzw. *angewählt*. Er wurde dann hervorgehoben dargestellt, und es erschienen unter Umständen zusätzliche Informationen zum Link. Erst mit einem Doppelklick folgte man dem Link und gelangte zum Zieldokument.

lichen Zusammenhalt eines Dokuments. Dabei kann zwischen lokaler und globaler Kohärenz unterschieden werden. Die lokale Kohärenz bezieht sich auf Relationen zwischen einzelnen Bedeutungseinheiten („Propositionen"), wogegen sich die globale Kohärenz auf größere Teile des Textes bzw. den Text als Ganzes bezieht (Dijk & Kintsch 1983: 189). Dies lässt sich auch auf Hypertext übertragen: Lokale Kohärenz liegt vor, wenn zwischen den Inhalten jeweils zweier Knoten eine für den Leser verständliche semantische Beziehung besteht. Globale Kohärenz setzt voraus, dass der Leser in den Inhalten aller Knoten ein gemeinsames Thema erkennen kann (Haake, Hannemann et al. 1991: 122). Somit bleibt globale Kohärenz in einem großen verteilten Hypertext wie dem Web unerreichbar, da aufgrund der Heterogenität der Autoren und Anwendungsgebiete weder ein einheitliches Thema noch ein konsistenter Stil eingehalten werden können (s. Hammwöhner 1993: 25). Lokale Kohärenz sollte hingegen immer angestrebt werden, da ihr Fehlen einen Hypertext fragmentiert erscheinen lässt und zu Verständnis- und Orientierungsschwierigkeiten führt (Thüring, Haake et al. 1991: 161-163; Hannemann, Thüring et al. 1992: 88; Oinas-Kukkonen 1999).

Die Potenziale der Link-Typisierung für die Benutzbarkeit von Hypertext-Systemen wurde in mehreren Studien belegt (Spyridakis 1989; Kopak 2000; Mobrand & Spyridakis 2002). Beispielsweise schnitten in einer Untersuchung von Lisa Baron Versuchsteilnehmer, denen frei typisierte Links[69] (s. Abschnitt 4.2.2) zur Verfügung standen, bei einer Suchaufgabe deutlich besser ab als die Vergleichsgruppe ohne Typisierung (Baron, Tague-Sutcliffe et al. 1996). In einem Versuch von Vora, Helander und Shalin erzielten Teilnehmer, die mithilfe einer grafischen Übersicht (s. Abschnitt 3.1.4) in einem Hypertext navigierten, ebenfalls signifikant bessere Ergebnisse, wenn die Links (dargestellt als Kanten) mit einem Titel versehen waren (Vora, Helander et al. 1994: 326).

Ein weiterer Vorteil typisierter Links sind die zusätzlichen Möglichkeiten zur *automatischen Verarbeitung und Aufbereitung* des Hypertextes. Beispielsweise können streng typisierte Links zur Generierung von *Übersichtskarten* (s. Abschnitte 3.1.4 und 5.3.4), zur *Linearisierung*[70] des Hypertextes und zur Erstellung von *Suchindexen* herangezogen werden. Suchsysteme nutzen die Strukturinformationen, um Dokumente mit bestimmten Eigenschaften zu ermitteln (s. Abschnitt 3.1.1), und *Filtermechanismen* (s. Abschnitt 3.1.6) grenzen mithilfe der Link-Typen die angebotenen Verknüpfungen und Dokumente thematisch ein und reduzieren so den *Information Overload* (s. Abschnitt 3.3.2).

Es haben sich allerdings auch Grenzen der Link-Typisierung gezeigt. Beispielsweise sind typisierte Links für den Leser nur dann hilfreich, wenn sie im Aufgabenkontext relevante Informationen für das Verständnis der Links und die Navigation im Hypertext-System beitragen. In Abschnitt 4.3 wird genauer auf diese Probleme eingegangen.

---

[69] Bei diesen sogenannten „Labelled Links" wurde ein Link-Titel hinter dem Text des Link-Ankers angezeigt.

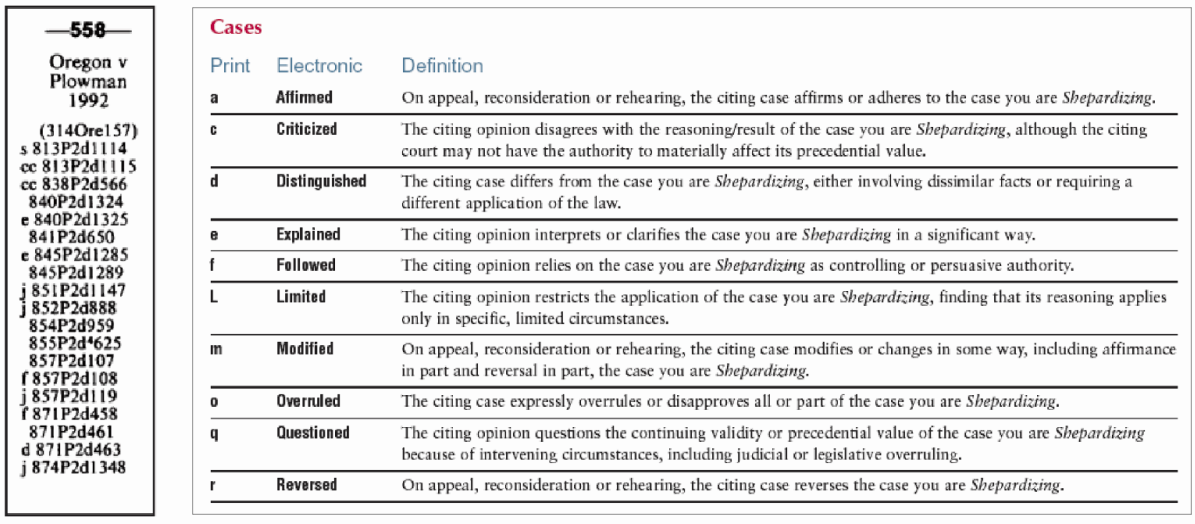
[70] Zur Linearisierung eines Hypertextes müssen die Link-Typen Informationen zu seiner Struktur enthalten.

4.2 Grundlagen und Eigenschaften typisierter Links

Eingeführt wurde der Ausdruck *typisierter Link (typed Link)* 1983 von Randal Trigg (Trigg 1983). Er gilt auch als Pionier der systematischen Analyse über die Möglichkeiten zur Typisierung von Links und hat dabei eine Typhierarchie für Hypertext-Systeme (s. Abschnitt 4.2.3) zur Unterstützung des wissenschaftlichen Diskurses entwickelt (Kopak 1999). Sein prototypisches Hypertext-System *TextNet* (siehe Anhang B.6) basierte auf diesen Arbeiten, und auch spätere Projekte, an denen er beteiligt war, unterstützten typisierte Links.

Das Konzept, dem Benutzer bereits beim Verweis mehr über das Ziel mitzuteilen, ist schon so alt wie das Referenzierungssystem der Bibel (s. Abschnitt 2.1.4): Da die Referenz den Ort des Zieles beschreibt – beispielsweise *„1. Mose 3,1“* für *„Erstes Buch Mose, Kapitel 3, Vers 1: Der Sündenfall und dessen Folgen“*[71] –, kann der theologisch Gelehrte implizit erkennen, welchen ungefähren Inhalt das Verweisziel hat. Dies stellt aber im engeren Sinne noch *keine* Link-Typisierung dar, weil nicht die *Relation* zwischen Quelle und Ziel angegeben wird.

Das Potenzial typisierter Verweise zeigte sich bei einem Konzept, das bereits vor über 100 Jahren in den Vereinigten Staaten unter dem Begriff *Shepardizing* bekannt geworden ist. Diese Idee geht auf einen Handelsvertreter für juristische Literatur namens *Frank Shepard* zurück, der zur Zeit des *Amerikanischen Bürgerkrieges* (1861-1865) nach einer Methode suchte, die es Rechtsanwälten einfacher machen könnte, zu ihrem aktuellen Fall Referenzfälle zu finden (DuCharme 2003). Er entwickelte für seine Kunden kleine *Klebezettel*, die mittels eines Ein-Buchstaben-Schlüssels und eines einfach codierten Literaturverweises zwischen verwandten Fällen charakterisierte Beziehungen herstellten. Der Schlüssel gab einen Hinweis auf die Art der Beziehung der Beschlüsse zueinander (a: bestätigend, c: kritisierend, r: gegensätzlich, siehe Abb. 25); der Code identifiziert die Publikation, in der der Fall erschien. Abb. 25 zeigt links ein Beispiel für bedeutende Referenzen zum *Fall 558: Oregon vs. Plowman*. Der rechte Teil der Abbildung gibt eine Übersicht zu den verfügbaren Verweistypen.



—558—
Oregon v
Plowman
1992
(314Ore157)
s 813P2d1114
cc 813P2d1115
cc 838P2d566
840P2d1324
e 840P2d1325
841P2d650
e 845P2d1285
845P2d1289
j 851P2d1147
j 852P2d888
854P2d959
855P2d625
857P2d107
f 857P2d108
j 857P2d119
f 871P2d458
871P2d461
d 871P2d463
j 874P2d1348

Cases

Print	Electronic	Definition
a	Affirmed	On appeal, reconsideration or rehearing, the citing case affirms or adheres to the case you are *Shepardizing*.
c	Criticized	The citing opinion disagrees with the reasoning/result of the case you are *Shepardizing*, although the citing court may not have the authority to materially affect its precedential value.
d	Distinguished	The citing case differs from the case you are *Shepardizing*, either involving dissimilar facts or requiring a different application of the law.
e	Explained	The citing opinion interprets or clarifies the case you are *Shepardizing* in a significant way.
f	Followed	The citing opinion relies on the case you are *Shepardizing* as controlling or persuasive authority.
L	Limited	The citing opinion restricts the application of the case you are *Shepardizing*, finding that its reasoning applies only in specific, limited circumstances.
m	Modified	On appeal, reconsideration or rehearing, the citing case modifies or changes in some way, including affirmance in part and reversal in part, the case you are *Shepardizing*.
o	Overruled	The citing case expressly overrules or disapproves all or part of the case you are *Shepardizing*.
q	Questioned	The citing opinion questions the continuing validity or precedential value of the case you are *Shepardizing* because of intervening circumstances, including judicial or legislative overruling.
r	Reversed	On appeal, reconsideration or rehearing, the citing case reverses the case you are *Shepardizing*.

Abb. 25: Ein Beispiel für das „Shepardizing“ und eine Übersicht der Verweistypen (LexisNexis 2003).

[71] Eine Umsetzung der Bibel als Online-Hypertext ist hier einzusehen: http://www.bibel-online.net/

Dieses Verfahren wurde von den Anwälten schnell als hilfreich erkannt, um andere relevante Gerichtsbeschlüsse bezüglich ihres aktuellen Falles aufzutun. Aufgrund des großen Erfolges brachte Shepard später Bücher mit solchen Verweisen heraus, in denen Anwälte zu jedem Verfahren schnell die entsprechenden Referenzfälle heraussuchen konnten. Auf diese Weise wurde bereits im vorletzten Jahrhundert eine *analoge typisierte Datensammlung von Links* geschaffen und profitabel vermarktet. Das Beispiel zeigt gleichzeitig, welches Potenzial das Anbieten professionell erstellter typisierter Links hat.

4.2.1 Konzepte zur expliziten Typisierung von Links

Obwohl die Typisierung von Hyperlinks von vielen Experten als ausgesprochen bedeutsam eingeschätzt wird, zeichnen sich nur relativ wenige Hypertext-Informationssysteme explizit durch typisierte Verweise aus (Hill, Hall et al. 1995; Nielsen 1995a; Shum 1996). Beispiele für Systeme mit explizit typisierten Links sind *TextNet*, *Sepia* und *gIBIS* (s. Anhang B und Abschnitt 5.3.4). So stellte Nielsen 1993 in seinem Buch „Hypertext and Hypermedia“ fest:

> „Most current hypertext systems have plain links, which are just connections between two nodes (and possibly anchors). The advantage of that approach is of course the simplicity of both authoring and reading. There is nothing to do with links except to follow them, and that one action can be achieved by the click of the mouse” (Nielsen 1993a).

Genauer betrachtet haben die meisten Hypertext-Systeme aber nicht nur „einfache Links“, wie Nielsen es ausdrückt: Ein Großteil der bisher entwickelten Systeme verfügt über Möglichkeiten, Links und/oder Link-Anker mit ergänzenden Informationen zu versehen, beispielsweise durch einen *Link-Titel*, *Namen* oder *Kommentar*. Da sich diese Daten ebenfalls auf den Verweis beziehen und die Bedeutung der Relation näher beschreiben, lassen sich auch solche Methoden konzeptionell der Typisierung von Links zurechnen (Halasz, Moran et al. 1987).

Basierend auf dieser Sichtweise können die in Hypertext-Systemen (s. Anhang B) und Standards (s. Anhang C) verwendeten Methoden zur Link-Typisierung drei wesentlichen Klassen zugeordnet werden. Sie werden im Folgenden in dieser Arbeit als *freie*, *strenge* und *semi-strenge Typisierung*[72] von Links klassifiziert.

4.2.2 Frei typisierte Links

Bei der *freien Link-Typisierung* können alle Hyperlinks mit einem beliebigen *Titel* oder *Namen* versehen werden. Man spricht daher auch von einem *Link-Titel* oder *Link-Label*. Der Typ besteht aus einem Wort oder einer kurzen Phrase, die direkt für den Anwender des Systems bestimmt ist.

[72] Der Autor dieser Arbeit hat sich bei der Terminologie an anderen Bereichen der Informatik orientiert, insbesondere den Konzepten zur Typisierung in Programmiersprachen (Louden 1994, S. 191 ff).

Diese Idee wurde bereits vom „Urvater" des Hypertextes – Vannevar Bush – eingeführt (s. Anhang A.1). In seinem visionären System *Memex* sollten sämtliche Verknüpfungen einen Namen erhalten:

> "When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard. Before him are the two items to be joined, projected onto adjacent viewing positions… The user taps a single key, and the items are permanently joined. In each code space appears the code word. Out of view, on each item [is designated] … the index number of the other item. Thereafter, at any time, when one of these items is in view, the other can be instantly recalled merely by tapping a button below the corresponding code space" (Bush 1945a: 107).

Bush nannte die Anker seiner Pfade *Code Spaces*, die Namen der Pfade hießen *Code Words*. Sie sollten die Intention des Benutzers beim Erstellen des Pfades festhalten und dem Leser später in Erinnerung rufen, was ihn beim Folgen des Pfades erwartet (s. Anhang B.1).

Das wohl erste Hypertext-System mit frei typisierten Links war *FRESS* (s. Anhang B.3). Die sogenannten *Jumps* waren bidirektional und hatten immer genau einen Start- und einen Zielanker, welche jeweils über einen eigenen Titel verfügten. Die Titel der Anker waren für die Benutzbarkeit des Systems entscheidend, da sie die einzige Information darstellten, die der Anwender über das Ziel (bzw. den Start bei rückwärts gerichteter Navigation) des Links erhielt. Anker wurden in der Form „`%J Titel des Link-Zieles%%`" (bzw. „`%P Titel des Link-Starts%%`" für rückwärts gerichtete Links) innerhalb des Dokumenttextes dargestellt (s. Abschnitt B.3 und Abb. 172). Gleichzeitig führt dieses Beispiel vor Augen, dass für bidirektionale Hyperlinks ein einzelner Link-Typ zumeist nicht ausreicht, sondern für beide Richtungen ein (möglicherweise gegensätzlicher) Typ erforderlich ist.

Ein weiteres renommiertes System, das es erlaubte, Links mit einem beliebigen Typ zu versehen, war *NoteCards.* Obwohl es ursprünglich mit dem Ziel der Wissensrepräsentation und -strukturierung entwickelt wurde und damit einen begrenzten Anwendungsfokus hatte, verzichteten Trigg, Halasz und Moran bei diesem Projekt auf ein festes Typsystem (Halasz, Moran et al. 1987). Die Angabe eines Link-Titels war aber obligatorisch. Er wurde als Link-Anker in spitzen Klammern dargestellt, gefolgt von dem Titel des Zielobjekts (s. Abb. 26, Anhang B.9 und Abb. 182).

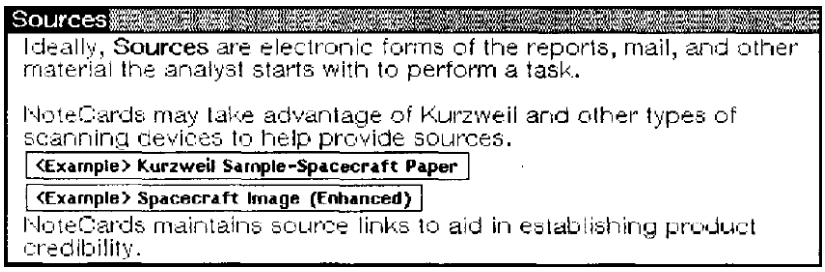


Abb. 26: Frei typisierte Links im *NoteCards*-System (aus Trigg, Suchman et al. 1986).

Mit den Systemen *Guide* (s. Anhang B.7) und *HyperTies* (s. Anhang B.8) wurde für Hyperlinks die Technik der „*Embedded Menues*" eingeführt. Hierbei wird kein Link-Titel definiert, sondern als Link-Anker ein Teil des Fließtextes hervorgehoben (s. Abb. 27). Dies sollte es ermöglichen, Links in normale Dokumente einzubetten, ohne die Lesbarkeit des Textes zu

reduzieren (Koved & Shneiderman 1986). Auf dieser Technik aufbauende Systeme (unter anderem auch das *World Wide Web*) zeigen daher häufig beim Link-Anker keinen expliziten Link-Typ oder Link-Titel mehr an.

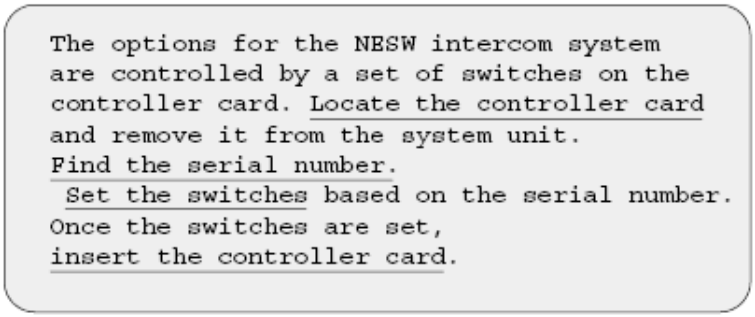


Abb. 27: *„Embedded Menues“* sind (im Beispiel durch Unterstreichung) hervorgehobene Phrasen eines Dokuments, die direkt ausgewählt werden können und als Link-Anker dienen (aus Koved & Shneiderman 1986).

Frei typisierte Links gestatten es dem Autor, sämtliche Referenzen mit individuellen zusätzlichen Informationen zu versehen und so dem Leser ergänzende Hinweise zur Funktion und zum Ziel der Verknüpfung anzubieten. Andererseits birgt diese Art der Typisierung das Risiko in sich, dass die Link-Titel innerhalb des Hypertextes *inkonsistent* eingesetzt werden und folglich ihre Verständlichkeit leidet. Dies gilt insbesondere, wenn zahlreiche Autoren an der Erstellung des Hypertext-Dokuments beteiligt sind (Rada 1990).

4.2.3 Streng typisierte Links

Systeme mit streng typisierten Links verfügen über ein *determiniertes Typschema* mit einer *vordefinierten* Menge von Link-Typen. Bei der Erstellung eines Links muss der Autor hieraus einen Typ auswählen. Solche Typschemata sind zumeist für einen bestimmten Themenbereich konzipiert (Bieber, Vitali et al. 1997b).

Bereits *ZOG* – eines der ersten Hypertext-Informationssysteme – unterschied zwischen zwei grundlegenden Link-Typen (s. Abschnitt B.4): Die sogenannten *Tree Buttons* dienten zur strukturellen Navigation in der Dokumenthierarchie und verwiesen auf untergeordnete Dokumente, wogegen *Annotation Buttons* assoziative Querverweise zwischen beliebigen Dokumenten herstellten. Sie waren einfach unterscheidbar, da bei Annotation Buttons ein „@“ vor dem Link-Text erschien (s. Anhang B.4, Abb. 173 und Abb. 174; Akscyn, McCracken et al. 1988). Im engeren Sinne würde man hierbei allerdings noch nicht von „typisierten Links“ sprechen, da der Link-Typ die Bedeutung des Links nur unwesentlich eingrenzt und ein semantisches Typschema fehlt.

Das erste Hypertext-System mit einer systematisch entwickelten Hierarchie von Link-Typen war *TextNet* von Randal Trigg. Die Typhierarchie unterschied zwischen zwei Hauptkategorien: *Normal Links* für assoziative Verweise und *Commentary Links* für Annotationen und Bewertungen (s. Abb. 28). Hinzu kamen insgesamt 80 Untertypen. Bein Erstellen eines Links musste der Hypertext-Autor einen Typ auswählen, um die semantische Beziehung zwischen den Objekten möglichst genau zu definieren (Trigg & Weiser 1986).

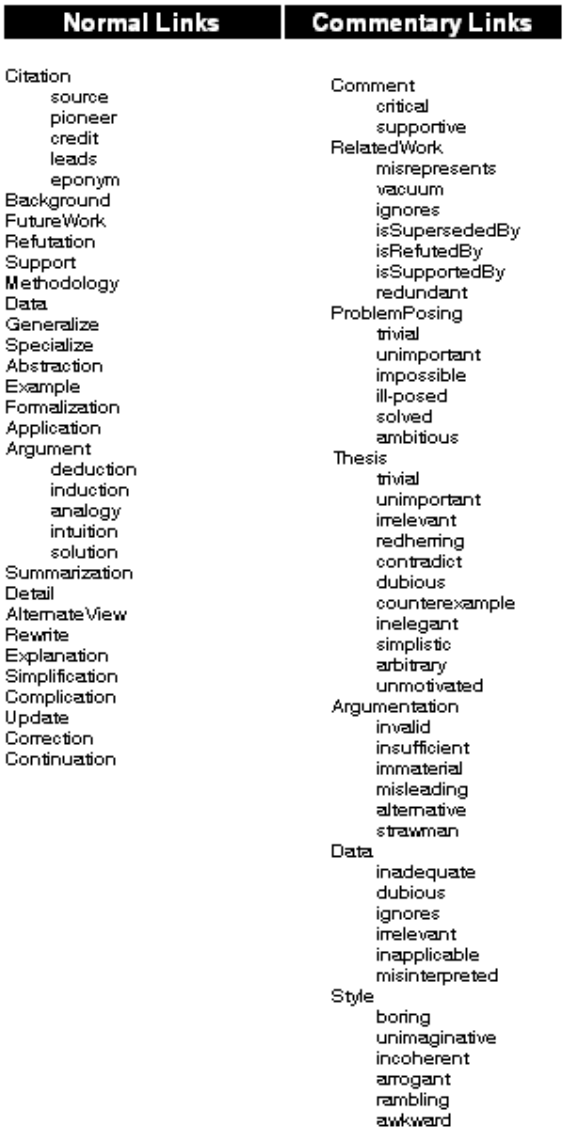


Abb. 28: Typ-Hierarchie des TextNet-Systems (aus Trigg & Weiser 1986).

Die Link-Typen wurden dem Leser im TextNet-Prototyp allerdings *nicht* angezeigt, sondern sie dienten primär zur automatischen Verarbeitung der Knoten, beispielsweise zur linearisierten Darstellung (s. Anhang B.6).

Das *Much* Hypertext-System verfügte über ein Typsystem mit 13 Link-Typen, die in die vier Hauptkategorien *Document* (für einfache und strukturelle Verweise zu anderen Knoten), *Annotation* (für Anmerkungen), *Reference* (für assoziative Verweise) und *Thesaurus* (für verwandte Begriffe, Beispiele und Ähnliches) eingeteilt waren (s. Abb. 29). Zusätzlich gab es die Möglichkeit, Links mit einem Namen zu versehen (Wang & Rada 1995). Die vier Haupttypen wurden durch *unterschiedliche Klammern* (rund, eckig, spitz und geschweift) gekennzeichnet, die den Link-Anker umschlossen. Die Typisierung diente ebenfalls zur automatischen Generierung verschiedenartiger Ansichten aus der Dokumentstruktur.

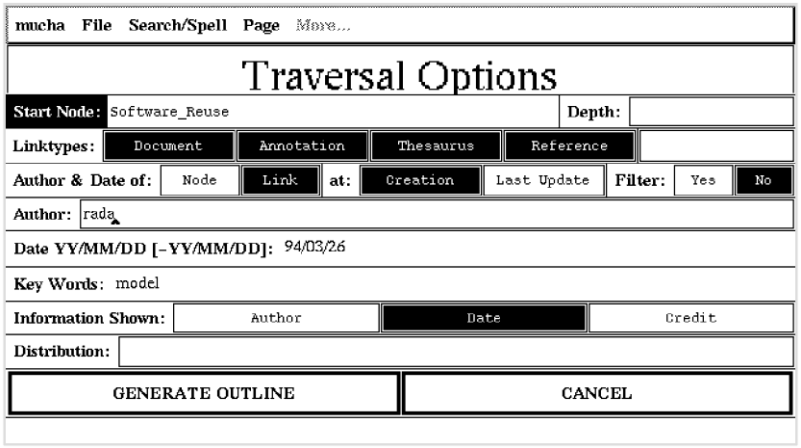


Abb. 29: Auswahl der Link-Typen in MUCH (aus Wang & Rada 1995).

In einigen der folgenden Hypertext-Systeme waren sowohl Links als auch *Knoten* typisiert. Beispielsweise bot das „*Decision Support System" gIBIS* die drei Knotentypen „Issue", „Position" und „Argument" und neun zugehörige Link-Typen, mit denen sich rhetorische Argumentationsketten zwischen den Knoten bilden ließen (Conklin & Begeman 1987; Bieber, Vitali et al. 1997a). Knoten und Links wurden mit ihrem Typ in einer Karte dargestellt (s. Abb. 53), die auch als primäre Schnittstelle zur Navigation und Bearbeitung diente (s. Abschnitt 5.3.4). Einen ähnlichen Ansatz verfolgte man im kooperativen Hypertext-System *Sepia*. Die Typisierung der Knoten beruhte auf einer vordefinierten Menge von Substantiven, und als Link-Typen dienten passende Verben. So ergaben sich kurze Phrasen, die den Benutzern die Bedeutung und Beziehungen der Objekte vermittelte (Thüring, Hannemann et al. 1995).

Streng typisierte Links weisen für bestimmte Anwendungsfelder Vorteile gegenüber der freien Typisierung auf. Bei der Erstellung eines Hypertextes unterstützen die Typen die *Strukturierung* der Informationen und stellen gleichzeitig eine *gemeinsame Basis* für die kooperative Nutzung des Systems dar. Zudem erleichtert ein festes Typschema das *Verständnis* eines Hypertextes, da der Leser vertraute Muster als Navigationshilfe vorfindet. Ein weiteres Potenzial besteht darin, dass streng typisierte Links zur *digitalen Verarbeitung* des Hypertextes herangezogen werden können, beispielsweise für die Linearisierung oder den Ausdruck. Darüber hinaus lassen sich streng typisierte Links nicht nur textlich, sondern auch durch platzsparende Symbole oder Icons repräsentieren (s. Abschnitte 5.2 und 5.3).

Auf der anderen Seite stößt eine strikte Typisierung schnell an ihre Grenzen, wenn ein Autor den gewünschten Typ nicht im Schema findet und er folglich die Bedeutung des Links nicht ausdrücken kann (Marshall & Rogers 1992). Das Aufstellen eines allgemeingültigen Typschemas ist eine besondere Herausforderung: Eine zu kleine Menge von Typen ist potenziell entweder *zu generisch,* um die Bedeutung der Links sinnvoll einzugrenzen, oder *zu spezifisch* und schränkt so das Einsatzgebiet des Hypertext-Systems ein. Zu viele Typen bergen das Risiko, unüberschaubar zu sein, sodass Autoren ähnliche Typen missverstehen und einen falschen Typ auswählen. So zog bereits Randal Trigg aus seiner Arbeit mit *TextNet* den

Schluss, dass wohl kein fester Katalog von Link-Typen jemals ausreichen würde, um alle Einsatzbereiche von Hypertext-Informationssystemen abzudecken (Trigg & Weiser 1986). Zumeist sollte das Typschema eines Hypertextes folglich durch die Autoren anpassbar sein (Rada 1990).

4.2.4 Semi-streng typisierte Links mit adaptivem Typschema

Eine Alternative zu beiden obigen Konzepten ist die *semi-strenge Link-Typisierung* mittels eines *adaptiven Typschemas*. Dabei hat der Autor den Link-Typ ebenfalls aus einer Menge von vorgegebenen, grundlegenden Typen auszuwählen; allerdings ist dieses Typschema für ihn erweiterbar. Auf diese Weise lässt sich ein Hypertext-System in unterschiedlichen Anwendungsgebieten einsetzen, da die Link-Typen an das Themengebiet anpassbar sind.[73]

Das Konzept semi-streng typisierter Links wurde bisher nur selten realisiert. Ein Beispiel für ein adaptives Typschema ist das „`role`"-Attribut von XLink (s. Anhang C.3.1). Sowohl Links als auch Anker in XLink sind hierbei mit einem maschinenlesbaren Typ in Form einer URN[74] zu versehen. Damit diese Information von den Applikationen ausgewertet werden kann, müssen entsprechende URN-Typen definiert werden.

Semi-streng typisierte Links bieten den Vorteil, dass in dem jeweiligen Hypertext ein konsistentes Schema von Link-Typen zur Verfügung steht und diese zur automatischen Auswertung und Verarbeitung des Hypertextes genutzt werden können. Das Typschema lässt sich dabei noch während des Erstellens der Dokumente und der Arbeit mit dem System erweitern und anpassen.

Zu Problemen kommt es, wenn die „Link-Ontologien" unterschiedlicher Hypertext-Autoren zueinander inkompatibel sind und sich Übersetzungsprobleme ergeben. Zudem besteht das Risiko, dass mehrere Autoren eines Hypertextes redundante Typen definieren.

4.3 Grenzen der Typisierung von Hyperlinks

Trotz der aufgeführten Vorteile ist die Typisierung von Links mit mehreren Herausforderungen für Leser und Autoren verbunden.

Aus *Leser-Sicht* können Link-Typen problematisch sein, wenn sie zur *Verständlichkeit* eines Links *nichts* beitragen. Berücksichtigt ein Autor bei der Typzuweisung nur sein eigenes Anwendungsfeld und seinen Wissenshintergrund und ignoriert er die potenziellen Aufgaben und Anforderungen der Leser, so kann der Link-Typ irreführend sein. Zudem hat sich gezeigt, dass es nicht bei allen Arten von Links notwendig und sinnvoll ist, detaillierte

---

[73] Zum Beispiel benötigt ein Hypertext-System mit technischen Inhalten andere Link-Typen als ein System mit medizinischen Dokumenten, um den unterschiedlichen Arten von Relationen zwischen den Dokumenten und den entsprechenden Fachtermini gerecht zu werden.

[74] Ein „Uniform Resource Name" ist ein „Uniform Resource Identifier" (URI), der als *dauerhafter* Bezeichner für eine Ressource dient. Er wird mit dem Schema „`urn:`" kenntlich gemacht. Ein Beispiel ist „`urn:isbn:3453146972`" für das Buch mit der entsprechenden ISBN-10-Nummer.

Typinformationen anzubieten. Beispielsweise erschließt sich meistens die Bedeutung *struktureller Links* wie „Zurück zur Homepage“ oder „Nächstes Kapitel“ aus dem Text des Link-Ankers. Für eindeutig verständliche Links können ergänzende Informationen potenziell hinderlich sein, da sie vom Benutzer zusätzlich berücksichtigt werden müssen, dies ihre Zeit und Aufmerksamkeit in Anspruch nimmt und sich so der „Cognitive Overhead“ weiter erhöht (s. Abschnitt 3.2.2). Sie können sogar zu messbar *schlechteren* Leistungen beim Erinnern von Informationen führen, wenn der Leser keine Notwendigkeit oder Motivation für die zusätzlichen Informationen erkennt (Jonassen 1993: 159-165). Solche Probleme lassen sich vermeiden, indem man den Link-Typ nur bei Bedarf einblendet („*Details-on-Demand*“), also wenn der Benutzer ein zusätzliches Informationsbedürfnis verspürt. In Kapitel 5 werden die unterschiedlichen Visualisierungsmöglichkeiten von Link-Markern und Link-Typen klassifiziert.

Es gibt auch Anwendungsbereiche von Hypertext, bei denen ambivalente Links gewünscht sind (s. Abschnitt 2.1.5). Wird z. B. ein System für die Lehre eingesetzt um das *explorative Lernen* zu unterstützen, so kann die Anzeige von Link-Typen dem gewollten *Serendipity-Effekt* entgegenwirken (s. Abschnitt 3.2.1) und das (scheinbar) zufällige Auffinden weiterer Interesse weckender Informationen verhindern (Kuhlen 1991; Warth 1999).

Aus *Sicht der Autoren* stellt die Auswahl des Link-Typs häufig eine kritische Herausforderung dar, die gerne vermieden wird. Dies zeigte sich während einer zweijährigen Studie mit über 200 Benutzern[75] des MUCH-Hypertext-Systems. Es verfügte über ein strenges Schema mit 13 Typen, von denen einer als Standard vorgegeben war (s. Abb. 29). Die Teilnehmer wählten bei nur ca. 3% der von ihnen erstellten Hyperlinks einen anderen Typ als den Vorgabewert aus, und selbst in diesen raren Fällen war die Auswahl häufig inkonsistent:

> “Contrary to the expectations of the builders of the MUCH system, the users did not exploit the ability to type semantic links. Typically authors used the default link type regardless of their semantic intentions. When a link type other than the default type was chosen, that choice was often inconsistent with the way another user would label a similar link” (Wang & Rada 1995).

Im Rahmen des ASK-Hypertext-Projektes wurden die *Ursachen* der Autorenprobleme bei der Erstellung typisierter Links genauer untersucht. Es stellte sich heraus, dass die Auswahl eines Link-Typs zeitaufwendig ist und ein präzises Wissen über die beteiligten Dokumente und das verfügbare Typschema voraussetzt. Zudem erstellten die Autoren immer wieder Strukturen, die entscheidend durch ihre Gewohnheiten und nicht durch die Inhalte des Hypertextes geprägt waren (Cleary & Bareiss 1996).

Bei einem *semi-strengen Typsystem* ist das Erstellen und Erweitern des Typschemas eine zusätzliche Schwierigkeit für die Autoren. Gerade bei mangelnder Erfahrung oder anfänglich nicht klar definiertem Gesamtinhalt des Hypertextes hat sich dies als entscheidender

[75] Alle 200 Teilnehmer waren auch als Autoren von Dokumenten und Links während der Studie aktiv.

Hinderungsgrund für die sinnvolle Nutzung der Link-Typisierung erwiesen (Marshall & Rogers 1992). Dabei ist insbesondere davon abzuraten, alle Link-Typen des Schemas *a priori* festzulegen (Marshall & Rogers 1992; Kopak 1999). Erfolgreicher ist der Ansatz, ausgehend von wenigen Standard-Typen, eine Taxonomie auf empirische Weise während der Erstellung des Hypertextes aufzubauen (Kopak 1999).

Aber auch die *freie Typisierung* von Links ist mit Autorenproblemen verbunden. Dies wurde bereits bei *NoteCards* festgestellt (s. Anhang B.9). Es ließ sich beobachten, dass die Mehrzahl der Autoren zumeist *keinen* Link-Typ angab, und wenn sie einen Typ wählten, war er häufig „falsch oder unnütz" (Bernstein 1997).

Vergleichbare Autorenprobleme zeigten sich bei der Definition von Typen für Hypertext-*Knoten*. Autoren äußerten „Frustration", wenn sie bereits bei dem Anlegen eines neuen Objekts einen Knotentyp festlegen mussten, da sie häufig noch keine genaue Vorstellung vom späteren Inhalt hatten. Sie fühlten sich behindert, da die Typisierung nicht ihrem gewohnten schrittweisen Vorgehen beim Entwickeln von Ideen und Strukturen entsprach (Marshall, III et al. 1994).

Die hier geschilderten Probleme bei der Typisierung von Links und Knoten in Hypertext-Systemen können im Zusammenhang mit der grundsätzlichen Beobachtung gesehen werden, dass die Kategorisierung von Daten für Menschen eine anspruchsvolle Aufgabe darstellt: oft fehlt die Übersicht über alle geeigneten Kategorien, und die meisten Informationen lassen sich nach unterschiedlichen Kriterien einordnen (Lansdale 1988; Kidd 1994).

Aufgrund der Autorenprobleme bei der Link-Typisierung wurde eine Reihe von Alternativen entwickelt: Eine Möglichkeit besteht darin, den Link-Typ *automatisch* aus Metadaten oder Typ-Informationen der am Link beteiligten Dokumente zu generieren (Cleary & Bareiss 1996). Ein weiterer Ansatz zur vereinfachten Link-Typisierung schränkt die Angabe der Relation zwischen Start- und Ziel-Dokument auf wenige *grundlegende* und einfach für den Autor anzugebende Dimensionen ein, beispielsweise den *Umfang* und die *Detailliertheit* der Informationen im Link-Ziel in Bezug zum aktuellen Dokument (Bernard, Crowder et al. 1995).

Weiterhin sollten *Link-Previews* (siehe Abschnitt 4.1.1) als Alternative oder Ergänzung zu typisierten Links stärker in Betracht gezogen werden. Sie haben den Vorteil, *bereits vorhandene Informationen* des Zielknotens zu nutzen. Somit entlasten sie den Hypertext-Autor und geben dem Leser dennoch Hilfen für die Navigation.

Die umfangreichen und in weiten Teilen positiven Erkenntnisse zur Typisierung von Links und zum Einsatz von Link-Previews wurden bisher im World Wide Web kaum berücksichtigt. Dies wird auch von vielen Experten kritisiert (siehe z. B. Bieber, Vitali et al. 1997a; Cailliau & Ashman 2000). Der folgende Abschnitt geht genauer auf den *Status quo* im Web ein und analysiert die existierenden Ansätze zur strengen und freien Link-Typisierung und zur Bereitstellung von Daten zum Zielknoten des Links im Link-Anker.

4.4 Link-Techniken und typisierte Links im World Wide Web

Auf den ersten Blick scheint das World Wide Web ein Hypertext-Informationssystem mit vergleichsweise simplen Links zu sein: Es gibt (fast) ausschließlich in die Dokumente *eingebettete* assoziative Links, ein Link führt (meist) zu *einem* anderen Dokument, das (in der Regel) *im selben* Browser-Fenster dargestellt wird (Oinas-Kukkonen 1999). Als Link-Marker dienen entweder kurze, farblich und durch Unterstreichung hervorgehobene Textabschnitte oder Grafiken, die vom Autor entsprechend gestaltet wurden (s. Abschnitt 4.4.1).

Eine genauere Analyse zeigt allerdings, dass es *mehrere* Möglichkeiten zur Link-Typisierung im Web gibt (Abschnitt 4.4.2), für Link-Previews lassen sich bereits im Startanker Eigenschaften des Link-Ziels spezifizieren (Abschnitt 4.4.3), und das Erscheinungsbild des Link-Markers kann von den Eigenschaften des Links und der Zieladresse abhängig sein (s. Abschnitt 4.4.4). Darüber hinaus haben zahlreiche Anker-Attribute Auswirkungen auf die Link-Aktion und das Verhalten des Browsers (s. Abschnitt 4.4.5). Programmatische Links bieten fast unbegrenzte Gestaltungs- und Interaktionsmöglichkeiten (s. Abschnitt 4.4.6).

Abgesehen von den assoziativen Links verfügt HTML über spezielle Elemente für *strukturelle Verweise* zwischen Dokumenten (s. Abschnitt 4.4.7) und zum Einbinden von Objekten (s. Abschnitt 4.4.8).

Der folgenden Analyse der vorhandenen Konzepte zur Typisierung von Links und der verschiedenen Link-Arten im Web liegen die aktuellen Standards HTML 4.01 (Raggett, Hors et al. 1999) und XHTML 1.1 Second Edition[76] (Stand November 2010; McCarron & Ishikawa 2010) als auch der kommende Standard HTML5 (Stand Sommer 2011)[77] zugrunde. Die hier beschriebenen Eigenschaften gelten somit für die meisten im Web angebotenen Dokumente (Wilson 2008). Zudem werden die Auswirkungen von CSS3 (Bos 2011) auf die Darstellung und Interaktion mit Links in (X)HTML-Dokumenten[78] berücksichtigt, da Browser sukzessive diesen Standard unterstützen.

4.4.1 Technische Grundlagen assoziativer Links im Web

Die allgegenwärtigen assoziativen Hyperlinks im Web werden in (X)HTML mittels des „a"-Elementes[79] definiert. Diese „Anker"-Elemente dienen als Start eines Links, wenn sie über

[76] Es gibt Bestrebungen, HTML durch verschiedene XML-Dialekte wie XHTML 1.1, XHTML 2.0 und XLink (siehe Abschnitt C.3) abzulösen. XHTML 1.1 ist mit HTML 4.01 weitgehend kompatibel und wird häufig eingesetzt, aber XHTML 2.0 ist in großen Teilen mit früheren HTML-Versionen inkompatibel und wird von aktuellen Webbrowsern nicht unterstützt. Aus diesem Grunde wurde die Entwicklung Ende 2010 eingestellt (Axelsson, Epperson et al. 2010), und es wird in dieser Arbeit nicht mehr berücksichtigt.

[77] Die Spezifikation von HTML5 liegt als „Working Draft" unter (Hickson 2011) vor und wird voraussichtlich erst 2014 fertiggestellt. Dennoch wird HTML5 bereits in großen Teilen eingesetzt und von aktuellen Browsern unterstützt. Kapitel 9.2 geht auf wichtige Neuerungen in HTML5 ein.

[78] Die Schreibweise (X)HTML steht im Folgenden stellvertretend für die beiden Sprachen HTML und XHTML.

[79] Das „a" ist eine Abkürzung für „Anchor", also „Anker eines Links".

das „href“-Attribut[80] mit einem URI als Wert verfügen, der das Ziel des Links spezifiziert. Anker-Elemente können auch als *Link-Ziel* innerhalb von Webseiten fungieren, wenn sie ein „id“-Attribut[81] aufweisen, dessen Wert den Zielanker spezifiziert. Solche Anker werden mittels des *Fragment-Teils*[82] in dem URI des Links adressiert (s. Abb. 30).

```
Mehr Informationen finden Sie in
<a href="/document.html#kapitel2">Kapitel 2 der Dokumentation</a>.
...
<h1><a id="kapitel2" name="kapitel2">Kapitel 2</a></h1>
```

Abb. 30: Ein Start- und ein Zielanker im HTML-Code einer Webseite.

Der Startanker eines Links muss immer einen Text und/oder weitere HTML-Elemente umschließen, die ein Objekt (z. B. eine Grafik) in die Seite einbinden (s. Abschnitt 4.4.8). Der Text bzw. die Grafik wird vom Browser als Link-Marker hervorgehoben. Folglich lassen sich Zeichen, Wörter, Grafiken[83] und Animationen als Link-Anker verwenden (Berners-Lee & Connolly 1995; Münz & Gull 2010).

Links im Web sind damit in der Regel *binäre*, *unidirektionale*, *gerichtete* und *nicht typisierte* Punkt-zu-Punkt-Verbindungen, die hinter den Möglichkeiten anderer Hypertext-Systeme mit *multiplen*, *bidirektionalen* und *typisierten* Links zurückbleiben (s. Abschnitt 2.1.4 und Tabelle 5). Bei genauerer Betrachtung sind die tatsächlichen Eigenschaften assoziativer Links im Web aber erheblich komplexer, als es auf diesen ersten Blick erscheint und oft kritisiert wird (beispielsweise durch: Bieber, Vitali et al. 1997a; Nelson 1999). Im Folgenden wird gezeigt, dass es durchaus Möglichkeiten zur *Typisierung* gibt, dass das *Aussehen* der Link-Marker und die *Interaktion* mit den Links definiert werden kann, und dass sich bereits im Startanker Angaben zum Zielobjekt machen lassen.

4.4.2 Methoden zur Typisierung assoziativer Links in HTML

Tim Berners-Lee wies bereits im ersten Projektantrag zum World Wide Web (s. Anhänge A.4 und B.15) auf die Notwendigkeit der Typisierung von Links in seinem System hin:

> “This example is basically a list, so the list of links is more important than the text on the node itself. Note that each link has a type ("includes" for example) and may also have comment associated with it” (Berners-Lee 1989).

---

[80] „href“ steht für „**H**yper **Ref**erence“ und drückt aus, dass Tim Berners-Lee diese Verknüpfungen als die assoziativen Hyperlinks des Webs konzipiert hatte (s. Berners-Lee 1989; Berners-Lee & Connolly 1995).

[81] Seit HTML 4 kann genau genommen jedes Element mit dem „id“-Attribut als Zielanker dienen. In älteren HTML-Versionen wurde stattdessen noch das „name“-Attribut benötigt, um Zielanker innerhalb von Dokumenten zu definieren. Dieses Attribut wird von vielen aktuellen Browsern nicht mehr unterstützt, daher werden in der Praxis oft beide Attribute verwendet, um eine Kompatibilität mit möglichst vielen Browser-Versionen zu erreichen (s. Abb. 30).

[82] Der „Fragment“-Teil einer URI ist nach RFC 3986 der Code nach dem #-Zeichen Berners-Lee, Fielding et al. 2005. Bei HTML-Links wird dies als Sprungadresse zu einem Zielanker innerhalb einer Webseite verwendet, beispielsweise führt http://www.scone.de/example.xml#**kaptitel2** zu dem Anker mit der ID „kapitel2“.

[83] Es besteht in HTML auch die Möglichkeit, Bereiche innerhalb von Grafiken mit Links zu versehen. Solche „Image Maps“ werden statt mit dem „a“-Element mit dem „map“-Element deklariert.

Er sah somit neben einer *strengen Typisierung* (Abschnitt 4.2.3) mittels eines Typschemas auch eine *freie Typisierung* (Abschnitt 4.2.2) von Hyperlinks durch Kommentare vor. Beides wurde bis heute aber in der Praxis unzureichend realisiert. (X)HTML bietet zwar Attribute zur Typisierung von Links, diese sind aber lediglich optional und werden nur selten genutzt:

- Die *freie Typisierung* von Links ist seit HTML Version 2.0 (Berners-Lee & Connolly 1995) mittels des „`title`"-Attributs möglich. Dieser „Link-Titel" wird seit Ende der 1990er Jahre von den meisten Browsern in einem Tooltip dargestellt (s. Abschnitte 5.3.7 und 5.5.2). Obwohl die Verwendung des Link-Titels, insbesondere für potenziell missverständliche Links, empfohlen wird (Nielsen 1998a; Chisholm, Vanderheiden et al. 1999; BITV2 2011), sind sie nur relativ selten zu finden. Bei einer 2002 von Bob DuCharme durchgeführten Studie mit über 20.000 zufällig ausgewählten Dokumenten aus den Suchdatenbanken von *Google* und *Yahoo!* wiesen von 440.000 untersuchten Link-Ankern weniger als 1% das „`title`"-Attribut auf (DuCharme 2002a). Bei einer im Dezember 2005 durchgeführten Analyse von knapp über einer Milliarde Dokumenten aus dem Google-Index waren es zwar bereits fast 21% (Google 2006). Dieser Wert stagniert aber seither, und bei einer Studie im Juli 2008 verfügten nur knapp 20% von über 3 Millionen Links über ein entsprechendes Attribut (Wilson 2008). Da nur ein Bruchteil der Links über einen Titel verfügen, weiß der Benutzer somit nicht, bei welchen Links er zusätzliche Informationen erhalten kann und ob sich das Warten auf einen Tooltip lohnt.[84]
- Nahezu ungenutzt und weitgehend unbekannt sind die Möglichkeiten zur *strengen Typisierung* von assoziativen Links in HTML, obgleich sie ebenfalls bereits in HTML 2.0 eingeführt wurden. Die zwei hierfür vorgesehenen Anker-Attribute „`rel`" und „`rev`" sollen durch den Browser interpretierbare Zusatzinformationen zum Link bereitstellen.[85] Diese Attribute werden aber gegenwärtig von keinem relevanten Webbrowser unterstützt. Im Standard HTML 4.01 wurde ein erweiterbares Schema mit 19 „sinnvollen" Werten als Vorgaben für beide Attribute definiert (Raggett, Hors et al. 1999), das allerdings erhebliche konzeptionelle Inkonsistenzen aufweist: Nur einige der Typen beschreiben die Art der Relation der verknüpften Objekte zueinander, wie „`next`" und „`previous`" für sequenziell aufeinanderfolgende Seiten. Andere Werte wie „`glossary`", „`index`" und „`contents`" definieren aber den Typ des *Zieldokuments* und nicht die Relation, die der Link zwischen Quelle und Ziel herstellt (Münz & Gull 2010). Bei der Analyse von Bob DuCharme (s.o.) wies keiner der untersuchten Link-Anker eines der beiden Attribute auf (DuCharme 2002a), und bei den Analysen von Google (Google 2006)

[84] Diese Problematik hat sich auch in den im Rahmen dieser Arbeit durchgeführten Studien bestätigt (vergl. Kapitel 6.2.1 und 6.5.3.1).

[85] Das Attribut „`rev`" wurde für HTML5 fallen gelassen, da es in der Praxis nahezu gar nicht eingesetzt wurde (Google 2006; Wilson 2008).

und Opera (Wilson 2008) waren es jeweils unter 3% der Anker.[86] In HTML5 wurde die Hälfte der in Version 4.01 vorgegebenen Werte fallen gelassen, da sie ungenutzt blieben (Wilson 2008) und durch 14 andere Attributwerte mit meist technischer Bedeutung ersetzt. Hierzu gehört „`prefetch`“, das Browser anweisen soll, ein Link-Ziel im Voraus zu laden, „`sidebar`“ zum Laden des Zieles im Sidebar des Browsers und „`external`“, das Links zu anderen Anbietern kennzeichnet.[87]

Inzwischen versuchen einige experimentelle Ansätze die strenge Typisierung von HTML nutzbar zu machen, beispielsweise XFN 1.1 („XHTML Friends Network“, siehe: Çelik, Mullenweg et al. 2006) aus dem Bereich des *Blogging*. Das „`rel`“-Attribut wird dabei verwendet, um Links mit unterschiedlichen „Tags“ im Sinne einer von Benutzern gebildeten „Folksonomy“ zu versehen (s. Abschnitt 3.1.6). Blogger können so beim „Verlinken“ ihrer Blogs die persönlichen Verhältnisse untereinander kenntlich machen.

4.4.3 Informationen zum Zielobjekt des Links

Die Sprache (X)HTML verfügt über drei Anker-Attribute, die bereits im Startanker Informationen zum Link-Ziel zur Verfügung stellen. Sie sollen dem Webbrowser die Möglichkeit geben, schon vor dem Folgen eines Links potenzielle Darstellungsprobleme auszumachen, sodass er entweder entsprechende Maßnahmen einleitet (z. B. eine zusätzliche Komponente installiert) oder den Benutzer auf das Problem hinweist (Raggett, Hors et al. 1999).

- Das „`charset`“-Attribut definiert, in welcher Zeichencodierung[88] das Zieldokument vorliegt.[89] Es werden die Codes der IANA („Internet Assigned Numbers Authority“) verwendet. Beispiele sind die Codierungen „ISO-8859-1“ und „UTF-8“ (IANA 2004).
- Das Attribut „`hreflang`“ beschreibt die Sprache des Link-Ziels. Es finden die international standardisierten Sprach-Codes der Network Working Group (Alvestrand 2001) Einsatz, wie „de“ für Deutsch im Allgemeinen und „de-CH“ für die Schweizer Ländervariante der deutschen Sprache.
- Das Anker-Attribut „`type`“ repräsentiert den MIME-Type des Zielobjekts (Freed & Borenstein 2006). Der Browser kann so den Benutzer auf das Dateiformat hinweisen.
- In HTML5 findet man zusätzlich das Attribut „`media`“, das als *Media Query*[90] den Medientyp bzw. die Geräteklasse angibt, für die das Zielobjekt optimiert wurde. So lassen sich

---

[86] Bei den Analysen war der mit großem Abstand am Häufigsten zu findende Attributwert `rel="nofollow"` (Google 2006; Wilson 2008) und somit keine inhaltliche Kennzeichnung des Links, sondern eine Angabe für Suchmaschinen, dass sie den Link-Anker beim Indexieren nicht berücksichtigen sollen.

[87] Die vollständige Liste findet man unter (Hickson 2011), http://www.w3.org/TR/html5/links.html#links.

[88] Hiermit ist kein *Schrifttyp* gemeint, sondern die *Binärcodierung* der einzelnen Zeichen, da landesspezifische Sonderzeichen zur korrekten Darstellung die Angabe der Zeichencodierung erfordern. Eine Alternative ist UTF-8, das seit einigen Jahren zunehmend zur Zeichencodierung verwendet wird und die Adressierung aller Zeichen des 16-Bit-Unicode-Standards erlaubt (mehr über Unicode findet man unter: http://unicode.org/).

[89] Das Attribut `charset` wird voraussichtlich in HTML5 fallen gelassen (Hickson 2011).

beispielsweise Links zu speziell angepassten Dokumenten für unterschiedliche Geräte unterscheiden (Hickson 2011).

Keines dieser Attribute wird von den gegenwärtigen Browsern unterstützt oder nennenswert im Web eingesetzt (DuCharme 2002a; Google 2006; Münz & Gull 2010). Eine fehlende Angabe der genannten Eigenschaften führt aber häufiger zu Problemen, beispielsweise wenn ein Link den Anwender zu einer Ressource in einer unbekannten Sprache führt oder auf seinem Computer die entsprechende Anwendung zur Darstellung des Dateiformats fehlt. Die Nutzung dieser Attribute und eine Erweiterung des Konzepts um weitere Eigenschaften des Link-Zieles wäre somit zweckmäßig, zumal sich solche Daten nicht nur technisch auswerten, sondern auch für Link-Previews einsetzen lassen.

4.4.4 Link-spezifische Darstellung von Link-Markern mit CSS

Das *Erscheinungsbild* von Link-Markern wird im Web primär mittels *Cascading Style Sheets (CSS)* und den zugehörigen „universellen" (X)HTML-Attributen `class` und `style` definiert (Münz & Gull 2010). Textliche Link-Anker lassen sich so beinahe beliebig formatieren: Neben Textfarbe und -stil sind auch Hintergrundfarben, Linien und Kästen als Link-Marker möglich (s. Abb. 31). Die meisten grafischen Browser unterscheiden dabei zwischen „*normalen*" Link-Ankern, Links zu *kürzlich bereits besuchten* URIs und „*aktiven*" (d. h. gerade ausgewählten) Link-Ankern (s. Abb. 31, Beispiele 1,2 und 5). Der Hinweis auf bereits besuchte Ressourcen ist einer der wenigen realisierten Ansätze für *Link-Previews* (siehe Abschnitt 4.1.1) im Web und wird von vielen Experten als sehr bedeutend für die Web-Navigation angesehen (Spool, Scanlon et al. 1998; Siegel 1999; Nielsen 2004a). Das sechste Beispiel in Abb. 31 hebt das **Ziel** eines Referenz-Links innerhalb der Seite (z. B. http://www.test.de/#**kapitel5**) in hellgelb mit schattiertem Rand hervor.

[90] *Media Queries* wurden ursprünglich für CSS entwickelt, um medien- und geräteabhängige Stile zu unterscheiden (Lie, Çelik et al. 2010). Beispiele sind „print and (resolution:600dpi)" für eine zum Ausdruck mit 600 DPI optimierte Datei oder „only screen and (max-device-width: 480px)" als charakteristische Definition für das iPhone.

```
/* Beispiele 1 und 2: "Pseudo-Klassen", die für alle */
/* Link-Marker des Dokuments gelten. */
a:link { color:blue; }
a:visited { color:purple; }
/* 3. und 4.: Weitere Möglichkeiten für die Hervorhebung von Link-Ankern. */
a.Boxed {
 border:  1px solid darkblue;
 background-color: #eeeeff;
 text-decoration: none;
}
a.Dotted {
 text-decoration:none;
 border-bottom: 1px dotted blue;
}
/* 5. Die Pseudo-Klasse „hover" gilt, sobald der Mauszeiger über dem Element ist. */
a:hover {
 color:red;
 background:#ffff80;
}
/* 6. Mittels der Pseudo-Klasse „target" wird festgelegt,  */
/* wie das Ziel eines Referenz-Links innerhalb einer Seite */
/* hervorgehoben werden soll. */
*:target {
 background:#ffffaa;
 box-shadow: 0 0 25px #ffffaa;
}
```

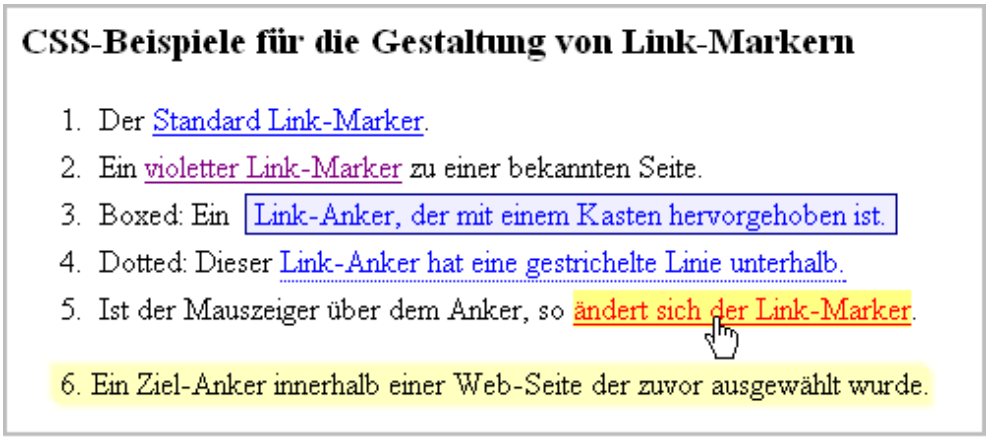


Abb. 31: Illustration zur Gestaltung von Link-Markern mit CSS. Oben der CSS-Code, unten die Darstellung eines Beispiel-Dokuments.

Der aktuelle Standard *CSS 3* bietet darüber hinaus Ausdrucksformen, um die Darstellung der Link-Marker von weiteren Attributen des Links und des Link-Zieles abhängig zu machen und den Benutzer somit automatisch auf Eigenschaften des Links hinzuweisen (Bos 2011). Das Beispiel in Abb. 32 zeigt eine Definition, die automatisch hinter allen Link-Ankern ein Icon einfügt, sofern die Ziel-Adresse mit „`http:`" anfängt[91] oder mit „`.pdf`" endet (für *Adobe-Acrobat*-Dateien, s. Abschnitt 4.5.3.1).

[91] Wobei es sich folglich um eine *absolute Adresse* handelt, die in der Regel einen *externen Link* zu einer anderen Website kennzeichnet (s. Abschnitte 2.2.2.1 und 4.5.1.2).

```
a[href^="http:"]:after {
 padding-left: 0.2em;
 content: url(/icons/extern.gif);
}
a[href*=".pdf"]:after {
 padding-left: 0.2em;
 content: url(/icons/pdf.gif);
}
```

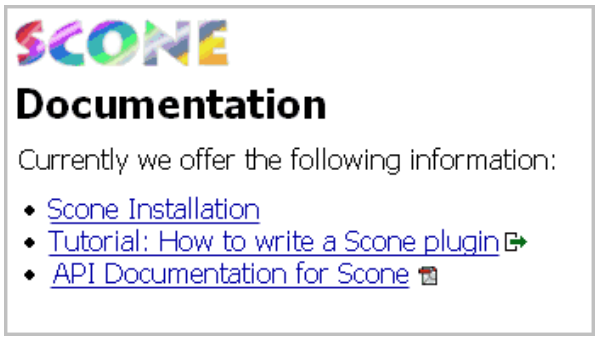


Abb. 32: CSS-Code, der eine Grafik hinter dem Link-Anker einfügt, die von dem Ziel-URI des Links abhängt.

4.4.5 Definition des Verhaltens von Links im Web

Das *Verhalten* von Links lässt sich in HTML mit unterschiedlichen Methoden beeinflussen. Mithilfe des „`target`"-Attributs kann im Link-Anker angegeben werden, *wo* das Ziel erscheinen soll. Beispielsweise führt der Wert „`_blank`" dazu, dass das Zielobjekt in einem neuen Fenster dargestellt wird. Durch die Angabe eines Namens lässt sich ein anderer Anzeigebereich (Frame) oder ein anderes Fenster als Ziel auswählen (Münz & Gull 2010).

Das *Hyperlink Presentation Module* von CSS3 erlaubt die Definition weiterer Link-Aktionen.[92] Der Browser kann angewiesen werden, für das Link-Ziel ein *modales*[93] Fenster zu öffnen oder einen *Browser Tab*[94] anzulegen. Weiterhin lässt sich angeben, ob das neue Fenster bzw. der neue Reiter im Vorder- oder Hintergrund erscheint (Çelik, Bos et al. 2004). Mit den in Abschnitt 4.4.4 vorgestellten Techniken sind solche Attribute gleichzeitig mit einem bestimmten Erscheinungsbild des Link-Markers zu verbinden; diese und weitere Möglichkeiten werden aber bis heute kaum unterstützt.

4.4.6 Programmatische Links im Web

Mithilfe von *Javascript-Anweisungen*, die in die Webseiten eingebunden sind und im Webbrowser ausgeführt werden, lassen sich *programmatische Links* erstellen und nahezu beliebige Verhaltensweisen für Links programmieren. Die meisten der erweiterten Link-Techniken anderer Hypertext-Systeme werden auf diese Weise realisierbar: Beispielsweise kann das Ziel des Links *dynamisch berechnet* werden, für *multiple Links* lässt sich ein Auswahlmenü einblenden (s. Anhang C.1.7) und mittels Ajax werden Teile einer Seite nachgeladen, um z. B. *Stretchtext-Links* anzubieten (s. Abb. 75 und Anhang B.7). Die Flexibilität heutiger Webbrowser ist dabei mit der von *NoteCards* (s. Anhang B.9) und *HyperCard* (s. Anhang B.11) vergleichbar.

---

[92] Genau genommen ist dies inkonsistent, da CSS primär die Darstellung von Elementen beschreibt.

[93] Ein modales Fenster liegt immer im Vordergrund einer Applikation. Erst nachdem es geschlossen wurde, lassen sich die Inhalte der anderen Fenster wieder bearbeiten.

[94] Alle neueren Browser bieten das sogenannte *Tabbed Browsing* an, bei dem innerhalb eines Browser-Fensters mehrere Dokumente in eigenen Bereichen als „Reiter" dargestellt werden.

Zum Aufruf der entsprechenden Funktionen dienen über 20 verschiedene HTML-Attribute, die als „*Event-Handler*“ auf jeweils eine bestimmte Benutzeraktion „*horchen*“ und beim Auftreten den im Attributwert angegebenen JavaScript-Code ausführen. Zum Beispiel reagiert das „`onClick`“-Attribut auf das Anklicken eines Elementes, „`onMouseOver`“ wird ausgelöst, sobald sich der Mauszeiger über dem Element befindet, und „`onKeyDown`“ wartet auf einen Tastendruck.

Die Integration von JavaScript ist mit einer ganzen Reihe neuer Herausforderungen verbunden. Aus *technischer Sicht* führt die mangelnde JavaScript-Kompatibilität[95] unterschiedlicher Browser häufig zu Entwicklungsproblemen. „Normalen“ *Autoren* bleiben viele dieser Techniken aufgrund ihrer Komplexität ganz verwehrt. Aus Sicht des *Anwenders* wird das Verhalten des Browsers oft unvorhersehbar: Anstatt nach einem Mausklick ein neues Dokument angezeigt zu bekommen, werden komplexe Abläufe programmiert, und ganze Web-Applikationen entstehen, die in Umfang und Funktionalität mit lokalen Desktop-Anwendungen vergleichbar sind (s. Kapitel 2.2.4). Es fehlen aber anerkannte Gestaltungsrichtlinien für eine konsistente Benutzungsoberfläche von Web-Applikationen, wie sie in anderen Bereichen der Softwareentwicklung seit Langem üblich sind,[96] sodass Anwender im Web regelmäßig mit neuen Benutzungskonzepten und Metaphern konfrontiert werden.

4.4.7 Strukturelle Links im Web

Relativ unbekannt und kaum genutzt sind *strukturelle Links* in HTML. Hiermit sind in diesem Abschnitt *nicht* die in die Dokumente eingebetteten Link-Anker gemeint, die (meist) in einem „Navigationsbereich“ der Seite dargestellt werden und zur hierarchischen Navigation innerhalb der Website dienen (s. Abschnitt 3.1.2 und Abb. 16 rechts), sondern die Möglichkeiten des „`link`“-Elements. Es wird im (unsichtbaren) Kopfbereich der Webseiten eingebunden und die Verweise werden mithilfe eines zusätzlichen Bedienelementes im Steuerungsbereich des Browsers zugänglich gemacht (s. Abb. 33 und Abb. 35).

---

[95] Genau genommen gibt es nicht „eine“ Sprache JavaScript, da unterschiedliche Browser und Browser-Versionen jeweils eigene JavaScript-Varianten implementieren. Da sie nur eingeschränkt zueinander kompatibel sind, führt dies regelmäßig zu Problemen bei der plattformunabhängigen Realisierung entsprechender Programme. Es gibt Bestrebungen, JavaScript zu standardisieren, die zu dem internationalen Standard ECMAScript geführt haben (1999). In der Praxis hält sich leider *kein* Browser-Hersteller vollständig an diese Spezifikation. Vereinfacht wird die JavaScript-Entwicklung inzwischen durch Ajax-Bibliotheken wie *jQuery* (http://jquery.com/) und *Prototype* (http://www.prototypejs.org/), die Browser-Inkompatibilitäten kapseln.

[96] Beispiele für entsprechende Standards sind die „Apple Human Interface Guidelines“ für das Max OS X der Apple-Computer (Apple Computer 2011) oder die „KDE Human Interface Guidelines“ (KDE 2011) des KDE-Projektes.



Abb. 33: Opera 11.5 zeigt strukturelle Links in einem Navigationsbalken an. Der Benutzer wählt gerade den Link zur Suche in der Website aus.

Das „link“-Element wurde bereits in HTML 2.0 (Berners-Lee & Connolly 1995) eingeführt und definiert hierarchische und sequenzielle Verknüpfungen. Die Art der Relation wird mittels des „rel“-Attributs festgelegt, wobei mehrere Attributwerte als Link-Typen vorgegeben sind. Beispielsweise definieren die Werte „start“, „contents“ und „index“ Verweise zur Homepage und zu Übersichtsseiten einer Website, „chapter”, „section” und „subsection” dienen zur Angabe von Kapiteln und Abschnitten und „prev“ und „next“ erlauben die sequenzielle Navigation.[97] Neben solchen strukturellen Referenzen lassen sich auch Beziehungen zu verwandten Objekten herstellen: Der Wert „alternate“ spezifiziert alternative Versionen des Dokuments, z. B. in anderen Sprachen und Dateiformaten. Der Autor beschreibt den Link mithilfe des Attributs „title“ und spezifiziert das Ziel mit dem „href“-Attribut (s. Abb. 34).

```
<link rel="start" title="Scone Homepage" href="/">
```

Abb. 34: Das „Link“-Element in HTML. Das Beispiel führt zur Homepage der Website.

Als erster Browser unterstützte 1995 „IXI Mosaic“ der „Santa Cruz Operation – SCO“ (Maloney & Quin 1995) das „link“-Element, und Ende der 1990er Jahre boten der zeilenorientierte Browser *Lynx* und der grafische Browser *iCab* für Macintosh-Rechner entsprechende Navigationselemente (s. Abb. 35). Die ersten populären Webbrowser mit einer Unterstützung dieser strukturellen Links waren *Mozilla 1.1* im Jahr 2001 (s. Abb. 33) und *Opera 7.0* (von 2003). Für die marktbeherrschenden Browser *Internet Explorer* und *Firefox* sind bis heute strukturelle Links aber nur mit speziellen Erweiterungen nutzbar, und entsprechende Daten werden folglich nur von wenigen Websites angeboten (Nahrath 2000).

[97] Das „link“-Element mit dem Attributwert „next“ wird von einigen Browsern ausgewertet, um die Folgeseite bereits im Hintergrund zu laden und dem Benutzer verzögerungsfrei anzuzeigen (Raggett, Hors et al. 1999).

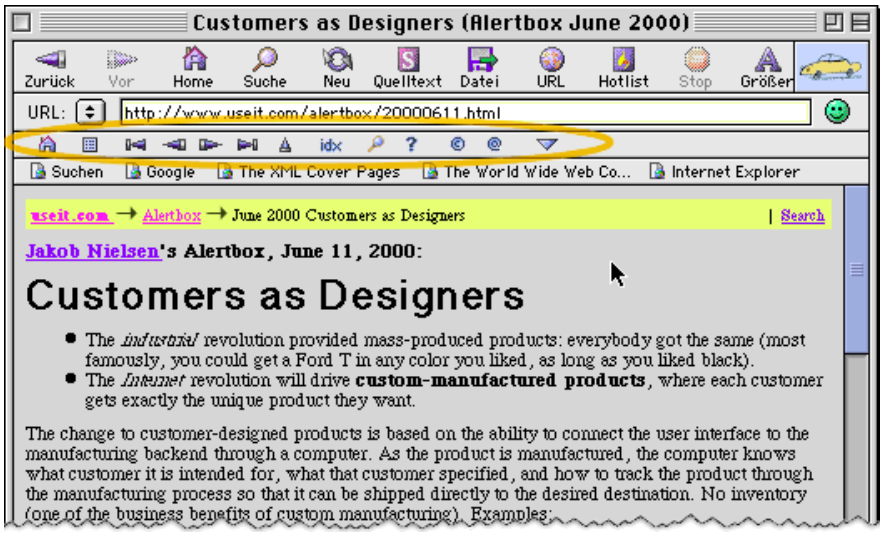


Abb. 35: Der Navigationsbalken in iCab zeigt strukturelle Links als Icons an.

Die Gebrauchstauglichkeit und Nützlichkeit dieses Navigationsmittels wurde bisher nicht evaluiert.

Da es für die Benutzung des Webs nur marginale Bedeutung besitzt, wurde es in dieser Arbeit ausgegrenzt.

4.4.8 „Transklusionen“: Verknüpfungen zum Einbinden von Objekten

Neben assoziativen und strukturellen Links gibt es in (X)HTML diverse Verknüpfungsmöglichkeiten zum Einbinden von Objekten in HTML-Dokumente. Der Hypertext-Pionier *Ted Nelson* sieht auch solche Verweise als ein entscheidendes Hypertext-Feature an, da sich mit ihrer Hilfe die Informationen mehrerer Autoren zu neuen Dokumenten *zusammenführen* lassen. Er nennt sie *Transklusionen*, allerdings hat er dabei wesentlich feingranularere Möglichkeiten im Sinn, als sie heute von HTML geboten werden (s. Anhänge A.3 und B.5).

Die meisten HTML-Elemente zum Einbinden von Objekten in Webseiten wurden Mitte der 1990er Jahre von den Browser-Herstellern als individuelle Erweiterungen ihrer Systeme eingeführt. Als Konsequenz gibt es eine Vielzahl von HTML-Elementen wie „`img`“, „`object`“, „`applet`“ und „`iframe`“, die eine jeweils eigene Syntax aufweisen.

Da keines dieser Elemente zur *Navigation* dient, wurden sie im Rahmen dieser Arbeit ebenfalls nicht genauer betrachtet.

4.4.9 Zusammenfassung der Defizite der Link-Typisierung im Web

Dieser Abschnitt hat einen Überblick über die vielfältigen technischen Möglichkeiten zur Verknüpfung von Objekten im Web gegeben. (X)HTML bietet neben Elementen für assoziative Links im Dokument auch Elemente für strukturelle Verweise und zum Einbinden anderer Ressourcen.

Für die Navigation im Web sind primär die assoziativen Links von Bedeutung. Sie stellen sogar das am Häufigsten genutzte Navigationsmittel von Webbrowsern dar (s. Abschnitte

2.2.4 und 3.1.8). Die Möglichkeiten, sie *explizit* zu typisieren, werden aber kaum eingesetzt: Die Attribute „`rel`“ und „`rev`“ zur strengen Link-Typisierung sind nahezu unbekannt. Obgleich das „`title`“-Attribut inzwischen von einigen Autoren genutzt wird, stellt es oft keine Navigationshilfe dar, zumal Benutzer nicht erkennen können, wann diese Informationen (in Tooltips) zur Verfügung stehen, welche Art von Informationen sie hier erhalten und ob diese bei der Navigation hilfreich sein könnten.

(X)HTML bietet darüber hinaus zahlreiche weitere häufig eingesetzte Link-Attribute, die Einfluss auf das Browser-Verhalten und die Funktion eines Links haben. Diese Eigenschaften bleiben dem Benutzer aber in der Regel bis zur Auswahl des Links verborgen und können zu unerwarteten oder ungewollten Aktionen führen. Es handelt sich somit um *implizite* Link-Eigenschaften, die zwar die Art und Funktion des Links ändern, aber nicht explizit als Mittel der Typisierung eingesetzt werden.

Die Grundlagen für *Link-Previews* sind in HTML ebenfalls vorhanden, liegen aber brach. Die drei Attribute „`charset`“, „`hreflang`“ und „`type`“ erlauben die Codierung wichtiger Eigenschaften des Link-Ziels bereits im Startanker, sie werden aber von aktuellen Browsern nicht berücksichtigt. Zudem handelt es sich hierbei um *technische* Eigenschaften des Zielobjekts, es fehlen semantische Attribute, die den Inhalt des Zieles beschreiben können.

Um die mit diesem Informationsdefizit verbundenen Probleme bei der Benutzung des Webs zu reduzieren, sollten Anwendern unerwartete Eigenschaften von Links und Zielknoten zugänglich sein, bevor sie einen Link auswählen (Harper, Yesilada et al. 2004; ISO9241-151 2008). Im folgenden Abschnitt werden daher die *impliziten* Eigenschaften von Links und Knoten im Web klassifiziert, um auf dieser Basis systematisch eine Technik zur automatischen Navigationsunterstützung zu entwickeln.

4.5 Eine Klassifikation impliziter Link-Informationen im Web

Nachdem im letzten Abschnitt die Möglichkeiten von Verknüpfungen in (X)HTML aus technischer Sicht betrachtet wurden, wird im Folgenden auf die tatsächliche Nutzung von Links im Web eingegangen. Dabei zeigt sich, dass die Eigenschaften assoziativer Links weitaus vielfältiger sind, als sich dies allein aus den verfügbaren Elementen und Attributen erschließt.

Ein entscheidender Faktor für die Bedeutung eines Links ist die referenzierte Ressource. Der URI als „universelles“ Adressierungsschema gestattete es nicht nur, Objekte unterschiedlicher Websites mit jedwedem Inhalt miteinander zu verknüpfen, es lassen sich auch beliebige Dateitypen adressieren und zahlreiche Protokolle verwenden. So können Links ebenfalls den Versand einer E-Mail-Nachricht initiieren, einen Download per FTP starten oder zu einer Interaktion mit einer Web-Applikation führen. Anwendern bleiben gegenwärtig oft relevante Details über die *Funktion*, *Bedeutung* und *Aktion* eines Links vor der Auswahl verborgen, obwohl diese Eigenschaften Einfluss auf seine Navigationsentscheidung haben

können. Solche Link-Eigenschaften sollten den Benutzern daher nicht verborgen bleiben (ISO9241-151 2008, 9.4). Zudem ist es für die *Erwartungskonformität* eines Software-Systems notwendig, dass es sich konsistent zu den Vorstellungen des Benutzers verhält; ein unerwartetes Verhalten steht dem entgegen (ISO9241-110 2006).

Damit Anwendern unerwartete, „unsichtbare" Eigenschaften von Links systematisch zugänglich gemacht werden können, erfolgt in diesem Abschnitt eine Klassifikation der *impliziten Eigenschaften* von Links und Knoten im Web. Sie berücksichtigt, ob und wie diese Daten als zusätzliche Link-Hinweise nutzbar sind und welche Methoden existieren, um sie automatisch zu berechnen und im Webbrowser zugänglich zu machen.

Diese Klassifikation basiert auf anderen Untersuchungen aus dem Hypertext- und Web-Bereich (u. a. Noirhomme-Fraiture & Serpe 1998; Oinas-Kukkonen 1999; Harper, Yesilada et al. 2004) und berücksichtigt die technischen Möglichkeiten von (X)HTML, URIs und JavaScript. Sie umfasst die folgenden Hauptkategorien:

- Die erste Hauptkategorie ist der *implizite Link-Typ* (Abschnitt 4.5.1). Hierzu gehört die *semantische Relation* zwischen Quelle und Ziel, die *Verteilung* der beteiligten Knoten zueinander und die *Aktion* des Links.
- Die zweite Gruppe fasst grundlegende *inhaltliche Meta-Informationen* zu den Objekten im Hypertext zusammen (s. Abschnitt 4.5.2).
- Als Drittes erfolgt eine Klassifikation der *Typen von Ressourcen* im Web. Hierzu gehört eine Einordnung nach *technischen* und *medialen* Eigenschaften sowie der *topologischen* Bedeutung in Bezug auf die Navigation im verteilten Hypertext (s. Abschnitt 4.5.3).
- *Statusinformationen* bezüglich der *Verfügbarkeit* der Ressource und der *Verzögerung* beim Zugriff bilden die vierte Hauptkategorie (s. Abschnitt 4.5.4).
- Fünftens werden Daten zur *Benutzung* der Objekte klassifiziert (s. Abschnitt 4.5.5).
- Der *URI* als Identifikator von Ressourcen besitzt eine besondere Bedeutung, die über die rein technischen Aspekte hinausgehen. Seine Rolle bei der Benutzung des Webs wird in Abschnitt 4.5.6 behandelt.

Hieraus ergeben sich 6 Haupt- und 17 Unterkategorien, die im Folgenden vorgestellt spezifiziert und analysiert werden. Die Klassifikation dient als Basis für die im Rahmen dieser Arbeit erstellten und evaluierten Konzepte und Prototypen mit dem Namen *HyperScout* (s. Kapitel 5ff).

4.5.1 Kategorien impliziter Link-Typen im Web

Links können im Web verschiedenartige Beziehungen zwischen Ressourcen herstellen. In diesem Abschnitt wird untersucht, welche Kategorien solcher impliziter Link-Typen im Web identifizierbar sind und inwieweit sie sich automatisch bestimmen lassen.

Mehrere Studien haben sich damit beschäftigt, welche Klassen von *semantischen Relationen* Links im Web ausdrücken. Abschnitt 4.5.1.1 geht auf drei entsprechende Studien ein, identifiziert die wesentlichen Link-Typen im Web und definiert Methoden zur Analyse des Typs. Zweitens wird in Abschnitt 4.5.1.2 der Charakter von Web-Links bezüglich der *Verteilung der Daten* zwischen Link-Start und Ziel beschrieben. Die dritte Kategorie impliziter Link-Typen im Web bezieht sich auf die *Link-Aktion* (Abschnitt 4.5.1.3).

4.5.1.1 Der semantische Link-Typ

Die ***explizite*** Typisierung von Links (s. Abschnitt 4.2.1) hat im Web bisher keine besondere praktische Relevanz erlangt (s. Abschnitt 4.4.2). Gleichwohl werden assoziative Hyperlinks ohne die explizite Typisierung für viele unterschiedliche Zwecke eingesetzt und stellen dabei ***implizit*** verschiedenartige *inhaltliche Beziehungen* her. Mehrere Experten haben die Verwendung von Links genauer analysiert und darauf basierende Typschemata aufgestellt (Conklin 1987: 33ff; Bannon & Grønbæk 1989; Frei & Stieger 1992: 103; Baron, Tague-Sutcliffe et al. 1996; DeRose 1998; Noirhomme-Fraiture & Serpe 1998). Zur weiteren Eingrenzung werden nachfolgend drei auf das Web bezogene semantische Klassifikationen von Link-Typen vorgestellt.

1. Eine Klassifikation von Links im Web auf der Grundlage früherer Typsysteme

Paul Thistlewaite erstellte bereits 1997 eine Klassifikation der Links im Web, die auf mehreren früheren Hyperlink-Typsystemen basierte (Thistlewaite 1997). Sein Forschungsprojekt hatte zum Ziel, alle Dokumente des australischen Parlamentes mit automatisch berechneten Hyperlinks zu versehen. Er betrachtete daher gleichzeitig die Möglichkeiten zur Generierung von Links und Methoden zur Darstellung der unterschiedlichen Link-Typen. Sein Schema setzt sich aus vier Typen zusammen:

- Der häufigste Link-Typ im Web ist der s*trukturelle Link* (*Structural Link*). Er setzt die Objekte des Informationssystems in eine hierarchische Beziehung. Solche Strukturen müssen in der Datenbasis des Informationssystems vorhanden sein und sollten in Listenform angeboten werden.
- *Referential Links* verknüpfen einen Begriff oder ein Konzept innerhalb eines Dokuments mit seinem „*Referenten*"; dies kann beispielsweise eine Quellenangabe, eine Definition oder die Homepage einer Person sein.[98] Sie sollten einfach hervorgehoben (z. B. unterstrichen) im Fließtext erscheinen.

[98] Zu Projektbeginn vertrat Thistlewaite die Meinung, dass *Referential Links* sich automatisch generieren lassen. Es stellten sich aber wesentliche Beschränkungen der Automation heraus, da häufig kontextuelle Informationen zum korrekten Anlegen eines Links benötigt wurden. Beispielsweise erfordert ein *Referential Link* vom Begriff „*Prime Minister*" ein Wissen über den zeitlichen Zusammenhang, um gemäß der Legislaturperiode die korrekte Person als Ziel zu definieren (Thistlewaite 1997).

- *Semantic Links* verknüpfen Dokumente, die gemeinsame Inhalte teilen. Links dieser Kategorie sind nicht automatisch zu generieren, da ihre Erstellung in der Regel *Wissen über die Thematik* voraussetzt. Solche Links sind somit selbst ein Informationsträger.
- Die vierte Klasse sind *Contingent*[99] *Links.* Sie setzen zwei Objekte in eine Beziehung, die nur unter gewissen Voraussetzungen oder bei bestimmten Aufgaben nützlich ist. Ihr Sinn erschließt sich nicht jedem Leser, und sie sollten daher nur der vorgesehenen Anwendergruppe dargestellt werden.

2. Eine Link-Klassifikation ausgehend vom Auftreten von Links auf Webseiten

Stephanie Haas und Erika Grams erstellten 1998 eine Hyperlink-Klassifikation auf Basis einer Analyse von 75 zufällig ausgewählten Webseiten verschiedener Anbieter und den knapp 1500 enthaltenen Links. In mehreren Iterationsschritten wurden drei Kategorien von Link-Typen identifiziert, denen sich 99% der gefundenen Links zuordnen ließen (Haas & Grams 1998a; Haas & Grams 1998b).

- Der häufigste Typ war mit einem Anteil von 48% der *Navigational Link*. Solche Verknüpfungen repräsentieren die *hierarchische Struktur* einer Website.
- *Expansion Links* (ca. 35%) verweisen zu detaillierteren Informationen in Bezug zum Text des Link-Ankers. Dies können unter anderem eine Illustration, eine Begriffsdefinition oder ein erläuternder Text sein.
- Die dritte Kategorie sind *Resource Links.* Sie führen zu inhaltlich verwandten, ergänzenden oder anderweitig lesenswerten Seiten und stellen häufig eine Empfehlung des Autors dar, falls der Leser weitere Informationen zu einem Thema wünscht.

3. Fünf Link-Typen im Fließtext von Webseiten

Im Jahr 2002 extrahierten Timothy Miles-Board, Leslie Carr und Wendy Hall aus einem Pool von ca. 750.000 Webseiten programmatisch 576 Dokumente mit längeren Textpassagen und relativ vielen im Fließtext eingebetteten Links. Auf diese Weise wollten sie die Verwendung von „wirklichen" assoziativen Hyperlinks im Web analysieren, da ihrer Beobachtung nach die meisten Webseiten fast ausschließlich strukturelle Links in speziell vorgesehenen „Navigationsbereichen" anböten, wodurch es dem Web an „echtem Hypertext-Charakter" mangele. Die im Textbereich eingebetteten Links werteten sie manuell aus und kamen zu einer Klassifikation mit fünf grundlegenden Link-Typen (Miles-Board, Carr et al. 2002):

- *Reference Links* treten innerhalb von Fließtext am häufigsten auf. Sie verknüpfen Substantive und Namen mit Dokumenten, die genauere Informationen zu dem Begriff enthalten, wie einer Produktbeschreibung oder einer persönlichen Homepage.

[99] Contingent wird hier als Adjektiv im Sinne von „ungewiss, zufällig, unvorhergesehen" verwendet.

- *Deep Links* verknüpfen Dokumente zu bestimmten Schlüsselwörtern oder Konzepten und helfen dem Leser, ein genaueres Verständnis der Sachverhalte zu erlangen.
- *Citation Links* werden verwendet, um Zitate oder Fußnoten mit der Quelle zu verknüpfen.
- *Glossary Links* verweisen von einzelnen Begriffen auf Glossar- oder Wörterbucheinträge.
- *Structural Links* dienen der strukturellen Navigation in Hierarchien oder Sequenzen von Dokumenten. Im Fließtext von Seiten sind sie nur selten zu finden.

Methoden zur automatischen Bestimmung des semantischen Link-Typs im Web

Die drei Klassifikationen weisen mehrere Überschneidungen auf. Ihnen ist gemeinsam, dass sie ***strukturelle Links*** (auch „Navigation Links" genannt, siehe: (Haas & Grams 1998a), die zur Navigation in der hierarchischen Struktur einer Website dienen, als häufigsten Link-Typ im Web identifizierten. Zudem gibt es jeweils einen Link-Typ, der von einem Begriff oder Konzept im Link-Anker zu entsprechenden Detailinformationen verweist (*Referential, Expansion* bzw. *Reference Links,* nachfolgend ***referenzielle Links*** genannt). Große Ähnlichkeit zueinander haben *Semantic, Resource* und *Deep Links* der drei Klassifikationen, da sie jeweils inhaltlich verwandte Dokumente miteinander verknüpfen und dem Anwender die Möglichkeit geben, weiterführende Informationen zu einem Thema zu erlangen (in dieser Arbeit ***semantische Links*** genannt).

Es gibt auch Unterschiede: Beispielsweise werden *Citation Links* und *Glossary Links* nur von (Miles-Board, Carr et al. 2002) ausgewiesen. Vergleichbare Typen findet man aber in frühen (pre-Web) Hyperlink-Klassifikationen wie der von Bannon und Grønbæk, die diese Verweistypen als *Bibliographic References* und *Keyword Links* bezeichneten (Bannon & Grønbæk 1989).

Eine automatische Bestimmung des Link-Typs ist selbst für die drei grundlegenden Link-Typen des Webs (*strukturelle*, *referenzielle* und *semantische Links*) schwierig. Anwender können hingegen oft aufgrund des Link-Ankers erkennen, mit welcher Art von Link sie es zu tun haben: Die Analyse von Haas und Grams hat einen bedeutenden Zusammenhang zwischen dem *Ort* eines Link-Ankers und dem Link-Typ ergeben: Sie stellten eine Korrelation von über 80% zwischen einer *isolierten Platzierung* und *strukturellen Links* fest sowie eine über 50-prozentige Übereinstimmung von *referenziellen Links* und einer im Text eingebetteten Darstellung, wobei der Link-Anker meistens ein (erweitertes) Substantiv war.

Strukturelle Links lassen sich zudem häufig durch einen Vergleich der Adressen von Ausgangs- und Zieldokument bestimmen. Viele Websites spiegeln die hierarchische Struktur der Site in den URIs wider, wobei die Hierarchiestufe durch den Pfad in dem URI repräsentiert wird. Eine einfache Auswertung der URIs zeigt dann, ob ein Link innerhalb der Struktur auf-

oder abwärts führt oder es sich um einen Querverweis handelt[100] (Nielsen 1999b; Kaasten, Greenberg et al. 2001).

Eine weitere Möglichkeit zur Identifikation von Link-Typen sind *Data-Mining-Verfahren für semistrukturierte Daten*. Solche Techniken werden häufig eingesetzt, um bedeutsame Objekte zu bestimmten Themen aus einem System vernetzter Informationen zu ermitteln (Kleinberg 1999; Lempel & Moran 2000), oder um bestimmte Strukturen herauszufiltern (Nestorov, Abiteboul et al. 1998; Ramakrishan 2000). Im Web lassen sich auf diese Weise *strukturelle Links* ermitteln, da sie innerhalb einer Site nach einem bestimmten Muster auftreten und meist denselben Anker-Text haben. Im Kontrast dazu stehen einmalig oder selten auftretende *semantische Links* mit individuellen Anker-Texten, die manuell erstellt werden und inhaltliche Beziehungen herstellen. Zur Identifikation der Link-Typen sind Daten über die hierarchische Struktur der Website hilfreich.

Insgesamt sind damit die Möglichkeiten zur automatischen Bestimmung des impliziten semantischen Link-Typs begrenzt. Sie beschränken sich im Wesentlichen auf heuristische Verfahren zur Ermittlung der drei aufgeführten Basistypen. Im Rahmen dieser Arbeit wurden Verfahren zur Kennzeichnung struktureller Links berücksichtigt und evaluiert, da die Hoffnung bestand, dass dies ihre Verständlichkeit verbessern könnte (s. Abschnitt 5.5.4.1).

4.5.1.2 Die Verteilungscharakteristik des Links

Im Gegensatz zu den meisten früheren Informationssystemen war das Web von Anfang an als global verteiltes System konzipiert. Die Möglichkeit der „Verlinkung" zwischen beliebigen Websites führt dazu, dass zwischen *lokalen Hyperlinks (*oder *within-site* bzw. *Site-spezifischen Links)* und *externen Hyperlinks (*auch *between-site* bzw. *Site-übergreifenden Links)* unterschieden werden kann (Catledge & Pitkow 1995; Tognazzini 1998). Ein lokaler Link führt zu einem Objekt auf demselben Server, wogegen bei einem externen Link[101] die URIs des Ausgangs- und Zieldokuments unterschiedliche Domain-Namen aufweisen[102] (s. Abschnitt 4.5.1.2) – Links haben somit eine bestimmte Charakteristik bezüglich der Verteilung der beteiligten Ressourcen.

Externe Links leiten den Leser zu Informationen eines anderen Anbieters. Als Folge stehen sie häufig in einem anderen inhaltlichen Kontext als das Ausgangsdokument: Der neue Anbieter kann andere Interessen verfolgen und eine abweichende Zielgruppe adressieren. Diese

---

[100] Dieses Verfahren versagt häufig, beispielsweise wenn eine Website restrukturiert wurde, ohne die Adressen anzupassen, oder wenn ein Content-Management-System mit parametrisierten URIs eingesetzt wird.

[101] Im Forschungsbereich offener Hypertext-Systeme wird der Term *externer Link* auch für Verweise verwendet, die nicht im Dokument eingebettet sind, sondern in einer davon unabhängigen Link-Datenbank gespeichert sind (Carr, De Roure et al. 1996). Diese Nomenklatur wird hier nicht verwendet, zumal diese Technik im Web nicht unterstützt wird.

[102] An dieser Stelle könnte noch ein weiterer Link-Typ aufgeführt werden: der W*ithin-Document-Link (bzw. Dokumenten-spezifische Link).* Er wird innerhalb dieser Klassifikation aber als eigene *Link-Aktion* aufgeführt (s. Abschnitt 4.5.1.3).

Faktoren können zu Verständnisproblemen führen, insbesondere wenn ein externer Link nicht als solcher erkennbar ist (Nielsen 1999a).

Studien haben ergeben, dass Web-Nutzer in der Regel nicht damit rechnen, dass ein Link zu einer anderen Site führt, sofern dies nicht explizit aus dem Link-Anker hervorgeht (Spool, Scanlon et al. 1998: 45ff). Häufig ist dies aber nicht der Fall, und der Benutzer muss der URI des aktuellen Dokuments – oben im Steuerungsbereich des Browsers – mit dem URI des referenzierten Objekts – unten in Statusbereich des Browsers – vergleichen (s. Abschnitt 5.3.5). Die hierdurch verursachte zusätzliche kognitive Belastung ließe sich einfach vermeiden, wenn Webbrowser externe Links adäquat kennzeichnen würden.

Aus technischer Sicht sind externe Links indes nicht immer an den Domain-Namen vom Ausgangsdokument und dem Link-Ziel zu erkennen, da einige Anbieter mehrere Domains verwenden oder sich auch mehrere Anbieter eine Domain teilen können. Häufig können der Inhaber des Domain-Namens aus dem DNS-Eintrag weiterhelfen, um solche Fragen zu ermitteln (s. Abschnitt 4.5.2.2), ein in der Praxis aber gegenwärtig aufwendiges Unterfangen.

4.5.1.3 Die Link-Aktion

Die Auswahl eines Hyperlinks kann zahlreiche unterschiedliche Aktionen zur Folge haben (vergl. Abschnitt 4.4). Die meisten Link-Aktionen wurden bereits für frühere Hypertext-Systeme entwickelt und benannt; einige der Bezeichnungen sind jedoch nicht eindeutig (vergl. Anhang B).

- Die charakteristische Link-Aktion im Web ist der sogenannte *Replacement Link,* bei dem das Zieldokument die Ausgangsseite im selben Fenster ersetzt (Bannon & O'Malley 1984; Nielsen 1993a: 96ff). Viele frühere Hypertext-Systeme ohne grafische Desktop-Oberflächen, wie *ZOG/KMS* (s. Anhang B.4) und *HyperTies* (s. Anhang B.8), boten nur diesen Link-Typ.
- *Popup Links* öffnen für das Zielobjekt ein neues Fenster (Gloor 1990: 30). Der Vorteil liegt darin, dass das Ausgangsdokument im Hintergrund auf dem Bildschirm zugreifbar bleibt. Andererseits verlieren Benutzer bei vielen geöffneten Fenstern schnell die Übersicht (s. Anhänge B.6 und B.9; Halasz 1988).
- *Note Links* öffnen ebenfalls ein neues Fenster; es bleibt aber nur so lange sichtbar, wie der Benutzer den Link aktiviert hat, also beispielsweise den Mausknopf gedrückt hält[103] (s. Anhang B.7; Brown 1987; Bannon & Grønbæk 1989).
- *Jumps* (auch *Reference Links* genannt) verweisen zu einer anderen Stelle im selben Dokument (Brown 1987; Nielsen 1993a: 96ff). Die meisten Hypertext-Informations-

[103] Ein Tooltip, der zusätzliche Informationen in einem Fenster darstellt, solange die Maus über dem Linkanker ruht (s. Abschnitt 5.3.7), ist nach dieser Definition *kein* eigenes Link-Ziel, da eine *explizitere Auswahlaktion* des Benutzers notwendig sein soll, um dem Verweis zu folgen.

systeme, die für Knoten (Dokumente) das Konzept der „*Schriftrolle*“ einsetzen, unterstützen solche Verknüpfungen (s. Abschnitt 2.2.1).

- Eine weitere Link-Variante ist der sogenannte *Stretchtext* (s. Anhang B.5). Bei der Auswahl eines Links wird der referenzierte Text unterhalb des Link-Ankers eingefügt. Dieses Konzept wurde bereits mit dem Hypertext-System *Guide* eingeführt (hier „Replace Button“ genannt (Brown 1987) und findet häufig im Bereich adaptiver Hypermedia-Systeme Einsatz (Brusilovski 1996).
- *Multiple Links* (auch „*multi-tailed Links*“ genannt, siehe: Ladd, Capps et al. 1994) verknüpfen mehrere Knoten miteinander (s. Abschnitt 2.2.1). Meistens wird dem Benutzer ein *Auswahlmenü* angeboten, aus dem er das gewünschte Ziel-Objekt auswählen kann. In der Variante des „*Fat Link*“ werden gleichzeitig mehrere Fenster mit unterschiedlichen Link-Zielen geöffnet (Nielsen 1993a: 115). Im Web findet man solche Fat Links beispielsweise in Form von „Popup-Fenstern“ mit Werbung, die sich automatisch zur Hauptseite öffnen.
- Inzwischen sind *programmatische Links* (auch „*Action Link*“ [Bannon & Grønbæk 1989] oder „*Command Link*“ genannt [Keep, McLaughlin et al. 1993]) von sehr großer Bedeutung im Web. Sie führen bestimmte Aktionen aus, starten eine Anwendung oder übergeben Parameter an ein Programm. Im Web wird aus einem Link ebenfalls ein *Action Link*, wenn im URI ein anderes Protokoll („Schema“) als „`http:`“ angegeben ist: Bei „`mailto:`“-Links wird der E-Mail-Client gestartet und bei „`ftp:`“ ein Download ausgelöst. Da HTTP zur Übertragung beliebiger Dateitypen verwendet wird, können darüber hinaus auch HTTP-Links zum Starten eines externen Viewers oder eines Dateitransfers führen. Mit noch komplexeren Link-Aktionen sind Benutzer häufig bei der Verwendung von Web-Applikationen konfrontiert. Dabei lösen Links bestimmte Operationen aus, wie das Hinzufügen eines Produktes zum Warenkorb oder das Abschließen eines Bestellvorgangs.

Alle Link-Aktionen außer dem *Replacement Link* können im Web eine unerwartete Interaktionsform für den Anwender bedeuten und daher potenziell Benutzungsprobleme zur Folge haben. Benutzbarkeitstests haben gezeigt, dass andere Aktionen des Browsers Benutzer oft irritieren, wenn nicht bereits vor der Auswahl aus dem Link-Anker hervorgeht, was passieren wird (Spool, Scanlon et al. 1998; Ojakaar & Spool 2001).

Gegenwärtig kann der Benutzer die Link-Aktion zumeist nicht sicher vorhersehen. Dies stellt beispielsweise gegenüber dem *Guide*-System einen Rückschritt dar, da hier der Benutzer anhand des Mauszeigers die Link-Aktion erkennen konnte (s. Abschnitt 5.3.6). Aus technischer Sicht wäre es häufig möglich, automatisch genauere Informationen zur Link-Aktion durch den Webbrowser anzuzeigen zu lassen, da sich die meisten entsprechenden Daten im HTML-, CSS- oder JavaScript-Code der aktuellen Seite befinden. Etwas problematischer ist die Auswertung, wenn sich der Code für die Link-Aktion im Zielobjekt befindet, also z. B. JavaScript-Code beim Laden der Seite ausgeführt wird, um Popup-Fenster

zu öffnen oder einen Download zu starten. In diesen Fällen müssen bereits vor dem Folgen des Links Daten zum Zieldokument vorliegen.

4.5.2 Kategorien inhaltlicher Informationen zu Ressourcen im Web

In einem Link-Preview sollten inhaltliche Informationen zum Link-Ziel abrufbar sein, sofern für den Benutzer Zweifel über die Bedeutung des Links bestehen (s. Kapitel 4.1.1 und Harper, Yesilada et al. 2004). Für gedruckte als auch digitale Dokumente gibt es mehrere grundlegende Meta-Informationen, die zur Identifikation von Dokumenten dienen und gleichzeitig Hinweise über den Inhalt vermitteln. In den nächsten Unterkapiteln werden die wesentlichen dieser Meta-Informationen charakterisiert.

4.5.2.1 Der Titel des Dokuments

Nahezu jedes gedruckte Dokument hat einen Titel oder eine Überschrift, und digitale Dokumente in digitalen Bibliotheken verfügen ebenfalls fast ausnahmslos über dieses Attribut (s. Weibel, John Kunze et al. 1998; Felfoldi 2005).

Dies gilt nur bedingt für Webseiten. Jedes syntaktisch korrektes HTML-Dokument *muss* an sich im „`head`"-Bereich ein „`title`"-Element mit einem Seitentitel aufweisen, das den Seiteninhalt auf „möglichst kontextreiche Weise" beschreibt (Raggett, Hors et al. 1999) und der auch außerhalb des Kontextes der Website verständlich sein soll, da er gerade in diesem Umfeld häufig genutzt wird: Der Seiten-Titel wird bei den *Ausgabeseiten von Suchmaschinen*, in der *History* des Browsers sowie bei *Bookmarks* (s. Abschnitt 3.1.7) zur Beschreibung der Ressource verwendet. Zudem ist ein sprechender Titel für die Navigation zwischen mehreren Browser-Fenstern und Browser-Tabs unverzichtbar, denn auch hier wird der Titel angezeigt.

Auf der anderen Seite muss sich der Autor bei der Länge des Titels beschränken, da seine primären Einsatzfelder nur wenig Platz zur Anzeige lassen und zumeist kein Zeilenumbruch möglich ist. *Google* zeigt beispielsweise in Ergebnislisten nur (ungefähr) die ersten 64 Zeichen des Titels an, die Darstellung in History und Bookmarks ist oft ähnlich stark beschnitten, und Browser Tabs oder die Taskleiste des Betriebssystems geben (je nach verfügbarem Platz) zumeist sogar nur die ersten 20 bis 40 Zeichen des Titels wieder (s. Abschnitt 3.1.7).

In der Praxis des Webs findet man leider häufig *nicht* die geforderten kurzen, kontextreichen und inhaltsbeschreibenden Titel vor (siehe Harper, Yesilada et al. 2004). Ein nicht zu vernachlässigender Anteil der Dokumente hat sogar gar kein entsprechendes Element oder der Inhalt des Elementes ist lediglich ein aussageloser Text wie „Einleitung" oder „Meine Homepage". Noch häufiger findet man unvollständige, inakkurate oder sogar falsche Informationen und Websites, die für jede Seite denselben Titel anbieten (Cockburn & Greenberg 1999; Kaasten, Greenberg et al. 2001). Eine Ursache für diesen häufigen Mangel liegt darin, dass der Inhalt des „`title`"-Elements bei den meisten Browsern lediglich im Titelbalken des Browsers erscheint und damit für den Autor längst nicht so auffällig ist, wie das Dokument selbst.

4.5.2.2 Autor und Anbieter

Der Autor bzw. der Herausgeber eines Dokuments ist eines der wichtigsten Charakterisierungsmerkmale gedruckter Werke. Er dient nicht nur zu ihrer Identifikation – z. B. bei Literaturverweisen –, sondern kann dem informierten Leser etwas über den Inhalt und die Güte eines Werkes sagen. Dieser Grundsatz scheint im World Wide Web wenig Relevanz zu haben, denn bei einem Großteil der Webseiten lassen sich weder Autor noch Herausgeber einfach feststellen. Nicht nur im Dokumenttext selbst fehlt diese Information allzu oft, auch wird das entsprechende *Meta-Attribut* zur Angabe des Autors nur selten eingesetzt (s. Abschnitt 4.5.2.6 und Google 2006). Darüber hinaus spricht aufgrund der Erfahrungen mit den in gedruckten Dokumenten verwendeten Literaturverweisen einiges dafür, diese Informationen dem Benutzer bereits vor dem Zugriff auf ein Dokument (also vor Auswahl des Links) zugänglich zu machen, sofern es sich um das Dokument eines anderen Anbieters als der Ausgangsseite handelt.

Der deutsche Gesetzgeber hat die rechtliche Problematik der Anonymität von Web-Ressourcen erkannt und im §5 des „Telemediengesetzes (TMG)“ als „Allgemeine Informationspflichten“ festgelegt, dass Name und Anschrift des „Diensteanbieters […] leicht erkennbar, unmittelbar erreichbar und ständig verfügbar zu halten“ ist (TMG 2007). Diese Information ist aber für Link-Previews kaum zu nutzen, da ihre Lokalisierung (welcher URI, welche Position im Dokument) in einer Site nicht standardisiert ist und sie sich daher nur schwer automatisch aus einer Website extrahieren lässt.

Grundsätzliche Informationen über den Anbieter einer Information im Web gibt meistens der Domain-Name der Site (vergl. Abschnitt 3.3.3), weshalb erfahrene Web-Nutzer bei *externen Links* auch häufig auf die Domain in dem URI achten (Spool, Scanlon et al. 1998). Präzisere Informationen als in der Domain alleine sind in der Regel – insbesondere bei größeren Firmen und Organisationen – im Titel und im Kopfbereich der Startseite einer Site zu finden. Alternativ können per *whois-Abfrage* die Eintragungen zum Domain-Namen herangezogen werden, die standardisiert und recht detailliert sind (Roessler 2002). Weitere Auskünfte über eine Site, die Art des Angebotes und die Anbieter geben Dienste wie *Alexa* und *DomainTools*.[104]

4.5.2.3 Datum der Erstellung und Aktualisierung

Bei gedruckten Dokumenten ist das Ausgabedatum zumeist eine ebenso bedeutende Angabe wie der Titel und der Autor. Bei Web-Dokumenten kann das Datum der Erstellung gleichermaßen relevant sein, allerdings ist dies wesentlich von der Art der angebotenen Information abhängig: Beispielsweise sind bei Börsenkursen und damit verbundenen wirtschaftlichen Informationen bereits Minuten entscheidend, bei der Suche nach wissenschaftlichen Doku-

[104] Die Dienste Alexa und *Domaintools* findet man unter www.alexa.com bzw. www.domaintools.com.

menten kommt es oft nur auf das Erscheinungsjahr an, und für Angebote im Unterhaltungssektor ist das Alter oft irrelevant.

Obwohl Jakob Nielsen bereits vor über 10 Jahren nicht gekennzeichnete, veraltete Dokumente als eines der „zehn schwerwiegendsten Probleme“ bei der Benutzung des Webs identifizierte (Nielsen 1996), scheinen sich die meisten Web-Autoren der Bedeutung des Alters ihrer Information oft nicht bewusst zu sein. Auf vielen Webseiten findet man weder das Datum der Erstellung noch der letzten Aktualisierung. Es gibt auch keinen bindenden Standard, der eindeutig regelt, wie diese Information im Dokument aufzuführen ist oder wie entsprechende Meta-Informationen auszusehen haben (siehe Abschnitt 4.5.2.6). Lediglich einige Web Design Guidelines weisen auf die Notwendigkeit der Angabe von Aktualisierungs- und Änderungsdaten hin (Weinreich 1997; Lynch & Horton 2002).

Bei offenen, verteilten Hypertext-Informationssystemen ist das Datum der letzten Aktualisierung der Dokumente noch für einen weiteren Aspekt bedeutsam: Wenn das Ziel eines Links überarbeitet wird, kann es den Link *semantisch inkorrekt* machen, da er sich auf eine ältere Version des Objekts bezieht. Dieses Problem tritt bei allen Informationssystemen auf, die keine Mechanismen zur Versionierung und Konsistenzkontrolle bieten, also auch beim Web (Nelson 1999). Verbyla nennt dieses Phänomen inhaltlich veraltender Links „*Semantic Decay*“ (vergl. Abschnitt 3.3.4; Verbyla 1999).

Aus technischer Sicht lässt sich im Web die letzte Aktualisierung eines Dokuments häufig aus dem Attribut „`Last-Modified`“ im HTTP-Header der Server-Antwort ermitteln. Hier soll das Datum der letzten Änderung einer Datei angegeben werden, damit Browser und Web-Proxies erkennen können, ob eine lokal gespeicherte Version der Seite noch aktuell ist (Fielding, Gettys et al. 1999). Diese Information ist aber bei Systemen mit dynamisch erstellten Seiten oft fehlerhaft, wenn hier beispielsweise das letzte Änderungsdatum eingebetteter Werbung herangezogen wird, nicht aber die dem Dokument zugrunde liegenden Informationen, die beispielsweise aus einer Datenbank stammen. Insgesamt kann die Situation im Web momentan somit als unbefriedigend bezeichnet werden.

4.5.2.4 Zusammenfassung des Inhalts

Damit Menschen einen schnellen Eindruck vom Inhalt eines Dokuments erhalten können, verfügen gut aufgebaute Dokumente in der Regel neben einer aussagekräftigen Überschrift über eine *Zusammenfassung* am Textanfang.

Im Web bieten nur wenige Seiten eine inhaltliche Zusammenfassung, zumal viele der Dokumente, verglichen mit gedruckten Dokumenten, recht kurz und trivial erscheinen. Um schnell einen Gesamteindruck von einer Webseite zu erhalten, „*scannen*“ daher die meisten Benutzer sie häufig nach den gesuchten Informationen, anstatt den Text sequenziell zu lesen. In einer Studie von Morkes und Nielsen scannten 79% der Teilnehmer *jede* beteiligte Webseite, bevor sie sie genauer lasen (Morkes & Nielsen 1997). Eine im Rahmen dieser Arbeit durchgeführte Studie hat gezeigt, dass Benutzer auf den meisten Webseiten nur sehr kurz

verweilen (Weinreich, Obendorf et al. 2006b), und sie in der Regel in der Zeit höchstens 28% des Inhalts lesen können (Nielsen 2008).[105] Medien-angemessen gestaltete Webseiten setzen aufgrund dieser verbreiteten Gepflogenheit auf relativ kurze Abschnitte mit aussagekräftigen Überschriften (Spool, Scanlon et al. 1998).

Sofern ein Dokument nicht über eine Zusammenfassung verfügt, besteht die Möglichkeit, automatisch eine Übersicht zu erstellen. Die Herausforderung liegt dabei darin, die für das Verständnis relevanten Teile eines Dokuments automatisch herauszufiltern. Bei multimedialen Inhalten sind die bereits erwähnten „Thumbnails" oft eine praktikable Lösung (Kaasten, Greenberg et al. 2001), bei textlichen Inhalten ist hingegen eine schriftliche Zusammenfassung zu bevorzugen, da verkleinerte Texte zumeist unlesbar sind (Woodruff, Faulring et al. 2001).

Das Erstellen von „Text Summaries" ist eine relativ komplexe Materie und wird unter anderem im Forschungsbereich der künstlichen Intelligenz behandelt. Gängige Methoden basieren in der Regel auf statistischen Auswertungen des Textes und lernenden Algorithmen.[106] Bisher gelten diese Techniken aber lediglich für längere Dokumente und entsprechend längere Zusammenfassungen als brauchbar; bei kurzen Texten – wie sie in Hypertext-Systemen üblich sind – ist die Güte solcher Inhaltsangaben gegenwärtig hingegen nicht sichergestellt und sehr von der Art des Dokuments abhängig (Kupiec, Pedersen et al. 1995; Salton, Singhal et al. 1997).

Automatische Inhaltsangaben für *Link-Previews* bieten als weitere Herausforderung, dass die textlichen Vorschauen ausgesprochen knapp sein sollten: Ein Exzerpt von 1-3 Zeilen wird für Link-Previews als angemessen angesehen (Conklin 1987; Utting & Yankelovich 1989). Diese „Zusammenfassung" kann somit nur wenige, kurze Kernsätze enthalten oder alternativ Schlüsselwörter aufführen. Bisher gibt es nur wenige Untersuchungen zur Gebrauchstauglichkeit automatisch erstellter Zusammenfassungen als Navigationshilfe im Hypertext, und es ist nicht bekannt, welche Methode sich am besten eignet und welcher Umfang für die darzustellenden Informationen optimal ist (vergl. Kaczmirek 2003; Paek, Dumais et al. 2004).

4.5.2.5 Sprache des Dokuments

Webseiten existieren in nahezu allen Schriftsprachen der Welt.[107] Englisch ist zwar bis heute die am weitesten verbreitete Sprache im Web, aber der Anteil nicht-englischsprachiger Seiten nimmt zu (Langer 2001; Pimienta, Lamey et al. 2001).

---

[105] Jakob Nielsen kam durch eine erweiterte Auswertung der Daten aus Weinreich, Obendorf et al. 2008 zu diesem Ergebnis.

[106] (Marcu 1997) gibt einen Überblick zu den wichtigsten etablierten Techniken zur automatischen Zusammenfassung von Dokumenten.

[107] Für Webseiten kann als Zeichencodierung das UTF-8-Format eingesetzt werden (Yergeau 1998), mit dem sich alle Zeichen des Unicode-Standards adressieren lassen (s. http://www.unicode.org/).

Die zunehmende sprachliche Diversität führt zu der Konsequenz, dass Anwender unerwartet auf Dokumente in einer für sie unbekannten Sprache stoßen können. Da solche Seiten zumeist nicht weiterhelfen, sollte der Benutzer *vor* dem Verfolgen eines Links auf potenzielle Sprachbarrieren hingewiesen werden. Dieses Problem wurde bereits vor Längerem erkannt und im Standard HTML 4 berücksichtigt: Er bietet ein entsprechendes Attribut für Hyperlinks mit dem Namen „`hreflang`“ (Raggett, Hors et al. 1999), das die Sprache des Zieldokuments beschreiben soll, aber ungenutzt blieb (s. Abschnitt 7.5.1; DuCharme 2002a). Gegenwärtig kann die Sprache eines Dokuments entweder der HTTP-Antwort eines Servers[108] oder mittels statistischer Verfahren aus dem Dokumenteninhalt ermittelt werden (Dunning 1994).

4.5.2.6 Weitere standardisierte Meta-Informationen in Webseiten

Bereits 1997 wurde in der Spezifikation von HTML 3.2 (Raggett 1997) die Syntax für ein generisches „`meta`“-Element definiert,[109] mit dem sich im „`head`“-Bereich eines Dokuments weitere Meta-Informationen ergänzen lassen. Inzwischen gibt es hierauf basierend eine kaum überschaubare Vielfalt von Standards, die entsprechende Attribute für das Element definieren (Daviel 1996; Felfoldi 2005): Neben vergleichsweise häufigen Attributnamen wie „*Description*“, „*Keywords*“ und „*Author*“, die ursprünglich von Suchmaschinenbetreibern eingeführt wurden (Sullivan 2007), haben einige Interessengruppen auch umfangreichere Spezifikationen entwickelt. Zwei nennenswerte Beispiele sind das „Dublin Core Metadata Element Set“ der Dublin Core Metadata Initiative (Weibel, John Kunze et al. 1998) und „GILS“ der US-Regierung, das aus dem Z39.50-Standard entstanden ist (Daviel 1996; Christian 2000). Neben inhaltlichen Beschreibungen können so weitere Meta-Informationen zum Dokument ergänzt werden, wie die Herausgeber des Dokuments, die Sprache und das Datum der letzten Aktualisierung (s. vorherige Abschnitte).

Leider konnte sich keiner der Standards bisher global durchsetzen, man findet nur auf einem Bruchteil der Webseiten entsprechende Elemente mit sinnvollen Informationen, zumal aktuelle Webbrowser dieses Element nicht berücksichtigen. Gegenwärtig ist die Situation somit wenig ergiebig, und auch für Link-Previews stehen selten brauchbare Informationen zur Verfügung.

Eine Hoffnung ruht auf der vom W3C geleiteten *Semantic Net Initiative*, in deren Rahmen unter anderem Standards für Meta-Informationen spezifiziert werden. Als Formate dienen das *Resource Description Framework RDF* (Berners-Lee, Hendler et al. 2001) und die *Ontologiesprache OWL* (McGuinness & Harmelen 2004). Der Durchbruch dieser Techniken ist bis heute

---

[108] Einige Websites bieten Informationen in mehreren Sprachen an. Damit der Server die Seiten automatisch in der bevorzugten Sprache des Anwenders darstellen kann, übermittelt der Browser die Systemsprache in der HTTP-Anweisung `Accept-Language`. Der Web-Server versucht dieser Anfrage zu genügen und teilt mittels des HTTP-Kommandos `Content-Language` mit, in welcher Sprache das Dokument bereitgestellt wurde.

[109] HTML selbst definiert nicht, welche konkreten Meta-Angaben zu Dokumenten hinzugefügt werden sollen oder können, sondern lediglich die Syntax. Dabei wird für allgemeine Meta-Informationen das „`meta`“-Element mit den Attributen „`name`“ (für die Art des Attributes) und „`content`“ (für den Wert) verwendet.

aber ausgeblieben, und es werden unter anderem ebenfalls Autorenprobleme wie bei der Typisierung von Dokumenten und Links befürchtet (s. Abschnitt 4.3; Marshall & Shipman III 2003).

4.5.2.7 Eignung inhaltlicher Informationen für Link-Previews im Web

Den in diesem Abschnitt aufgeführten Meta-Informationen zu Ressourcen im Web ist gemein, dass sie nur in den Ziel-Dokumenten selbst zu finden sind. Sie sind daher für *Link-Previews* nur eingeschränkt nutzbar, da dem Browser vor dem Folgen eines Links keine detaillierten Informationen über das Link-Ziel zur Verfügung stehen. Kapitel 7 stellt daher ein neues Konzept zur effizienten Bereitstellung solcher Daten im Web vor.

4.5.3 Typisierung der Ressourcen im Web

Der *Typ des Zielknotens* ist die dritte Hauptkategorie von Informationen, die für Benutzer bei der Navigation in verteilten Hypertext-Informationssystemen oft von Bedeutung ist. Im Rahmen dieser Arbeit wurden, basierend auf früheren Arbeiten aus dem Hypertext-Bereich, drei Kriterien zur Klassifikation von Web-Ressourcen identifiziert.

Offene Hypertext-Systeme erlauben die Integration beliebiger Dateiformate (s. Abschnitt 2.2.2.2). Der *technische Typ* eines Objekts ist aus Benutzersicht wichtig, da viele Dateitypen spezifische Software- und Hardware-Anforderungen haben und mit ihnen teilweise eine spezifische Form der Interaktion verbunden ist (Abschnitt 4.5.3.1).

Das zweite Klassifikationskriterium geht von einer medialen Betrachtungsweise aus: Ausschlaggebend für den *Medientyp* eines Objekts ist die Art der Darstellung und die sinnliche Wahrnehmung durch den Anwender. Beispielsweise ist für den Benutzer bei der Navigation oft von Interesse, ob ihm ein Text-Dokument, eine Illustration oder ein Video erwartet. Für das Informationsbedürfnis ist das Dateiformat – also ob es sich um eine MPEG-, Quicktime- oder RealVideo-Animation handelt – sekundär (Abschnitt 4.5.3.2).

Das dritte Klassifikationsmerkmal bezieht sich auf die Funktion eines Knotens bezüglich der Navigation. Da es in engem Zusammenhang mit der Graphenstruktur des Hypertextes steht, wird es als *topologischer Typ* bezeichnet (Abschnitt 4.5.3.3).

4.5.3.1 Der technische Typ: das Dateiformat des referenzierten Objekts

Offene Hypertext-Systeme zeichnen sich dadurch aus, dass Benutzer alle ihre Dokumente in beliebigen Formaten in das System integrieren und mit Links versehen können (Pearl 1989; Hall 1997), vergl. auch Abschnitt 2.2.2.2). Beim World Wide Web wurde dieses Ziel nur teilweise erreicht, und es lassen sich zumindest beliebige Dateien als *Link-Ziel* definieren, sofern die Ressource per URI adressierbar ist (s. Abschnitt 2.2.3ff). Zur Darstellung wird je nach Dateiformat oft eine spezifische Applikation benötigt, die die Datei darstellen kann; die Objekte lassen sich somit aus *technischer* Sicht einem bestimmten Typ zuordnen.

Die Integration unterschiedlicher Dateitypen kann eine ganze Reihe von Benutzbarkeitsproblemen zur Folge haben. Im Web ist HTML das originäre Datenformat, daher erwarten Benutzer normalerweise auch ein HTML-Dokument als Zielobjekt. Ein Link kann aber beispielsweise ebenso auf ein Word- oder PDF-Dokument verweisen. Unterschiedliche technische Formate sind häufig mit spezifischen Anforderungen und Möglichkeiten verbunden, selbst wenn sie denselben Inhalt verkörpern (s. Abb. 36).

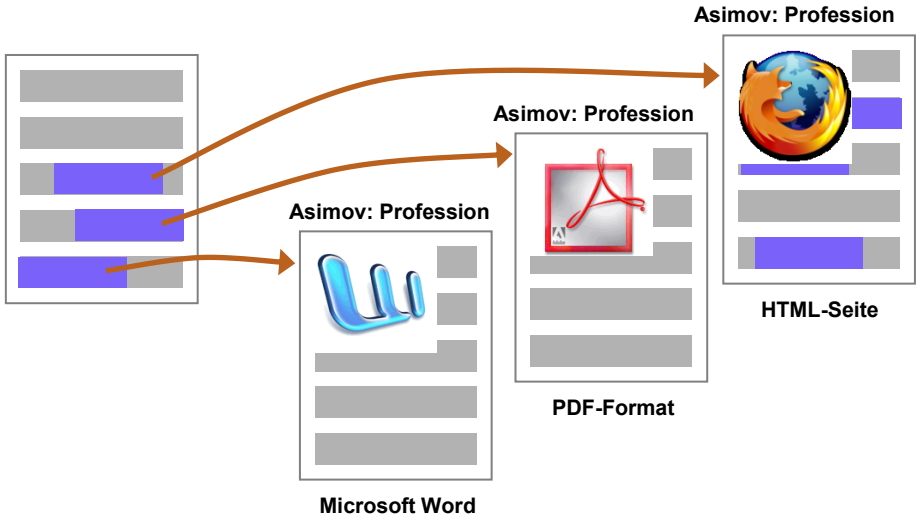


Abb. 36: Drei Links auf Text-Dokumente in unterschiedlichen Dateiformaten.

Aus technischer Sicht ergeben sich hieraus zahlreiche Schwierigkeiten: So muss die Hardware den Minimalanforderungen genügen. Es ist beispielsweise sicherzustellen, dass Bildschirmauflösung, Farbtiefe und Rechenleistung ausreichend sind. Ein Standard, der sich mit der Problematik technischer Systemanforderungen in verteilten Informationssystemen auseinandersetzt, ist „*Composite Capability/Preference Profiles (CC/PP)*“ des W3C (Butler, Hjelm et al. 2004). Dabei teilt der Client dem Server seine technischen Eigenschaften mit, woraufhin der Server die Daten in angemessener Form zur Verfügung stellt. Dieser Standard wird bisher primär im Bereich mobiler Geräte eingesetzt, beispielsweise in Form des darauf aufbauenden *UAProf* der *Open Mobile Alliance*,[110] welches unter anderem *RIMs Blackberry-Systeme*[111] verwenden. Die populären Webbrowser für PCs berücksichtigen diesen Standard nicht.

Bei modernen Arbeitsplatzrechnern stehen beim Zugriff auf unterschiedliche Dateiformate zumeist die Software-Anforderungen im Vordergrund. Fast jedes Format setzt eine eigene Applikation oder ein eigenes Browser-Plug-in zur Wiedergabe voraus. Für den Anwender kann dies zum Problem werden, wenn die benötigte Software nicht bereits auf seinem Computer installiert ist. Wird als Folge das Zielobjekt nicht oder inkorrekt angezeigt, oder

[110] Informationen zur *Open Mobile Alliance* finden sich unter http://www.openmobilealliance.org/.

[111] Blackberry bezeichnet den erfolgreichen proprietären Standard der Firma „Research In Motion Limited“, der Mitarbeitern eines Unternehmens den mobilen, kabellosen Zugriff auf ihre persönlichen Geschäftsdaten – wie E-Mails, Termine und Kontakte – ermöglicht.

muss der Anwender zusätzliche Programme installieren, so reduziert dies die Gebrauchstauglichkeit des Systems wesentlich.

Im Web zeigt sich eine weitere Problematik der Integration unterschiedlicher Dateitypen. Wenn ein Dokument eine eigene Applikation zur Darstellung benötigt, wird es zumeist außerhalb der Hypertext-Standardanwendung (dem Webbrowser) geöffnet. Die gewohnte Navigation des Benutzers wird dadurch unterbrochen, und er kann z. B. nicht mehr mit dem Back-Button zur ursprünglichen Seite zurückkehren oder ein Bookmark erstellen. Bietet der Dateityp keine eigenen Links, so wird das Dokument darüber hinaus zur „Sackgasse".

Web-Gestaltungsrichtlinien wie (Levine 1996; Lynch & Horton 2002; BITV2 2011) empfehlen, solche unerwarteten technischen Eigenschaften von Link-Zielen sichtbar zu machen. Wie dies geschehen soll, ist aber nicht festgelegt, und es fehlen Standards für eine erwartungskonforme Benutzungsschnittstelle.

Aus technischer Sicht ist der Dateityp einer Web-Ressource relativ einfach zu ermitteln. Erste Hinweise ergeben sich bereits aus dem URI des Link-Zieles, da sich der Dateityp *zumeist* in der Erweiterung des Dateinamens widerspiegelt. Der tatsächliche Dateityp wird dem Browser bei der Übertragung des Objekts im HTTP-Header als *Mime-Media-Type* (Freed & Borenstein 2006) übermittelt. Eine weitere Möglichkeit besteht daher darin, diese Information im Vorfelde mittels eines HTTP-„HEAD"-Befehles vom Server abzufragen, was die Übertragung von einigen Hundert Byte pro Link erfordert (Stanyer & Procter 1999). Erheblich effizienter ist das „`type`"-Attribut des Anker-Elements von HTML, in dem der Mime-Media-Type des Ziels angegeben werden kann (vergl. Abschnitt 7.5.1), das bisher aber ungenutzt ist (DuCharme 2002a; Google 2006). Eine Schwierigkeit wurde bei der Konzeption dieses Attributs allerdings offensichtlich übersehen: Wenn das Ziel eines Links ein HTML-Dokument ist, so kann es weitere Dateitypen einbinden, wodurch wiederum einige der bereits geschilderten Probleme auftreten. Sinnvoller wäre es daher, bereits die Typen aller eingebundenen Objekte der Zielseite im Link-Anker zugänglich zu machen.

4.5.3.2 Der Medientyp

Beim Medientyp des Dokuments geht es im Gegensatz zum technischen Typ nicht darum, in welchem Binärformat die Datei vorliegt, sondern in welcher medialen Form die Informationen wiedergegeben und vom Anwender wahrgenommen werden. Das Spektrum reicht von einfachen Text-Dateien über Illustrationen bis hin zu Filmen und interaktiven Animationen. Der Medientyp des Objekts kann bei der Informationsrecherche von Interesse sein, wenn der Benutzer beispielsweise eine Illustration benötigt oder eine Animation die Präsentationsform der Wahl ist (Abb. 37).

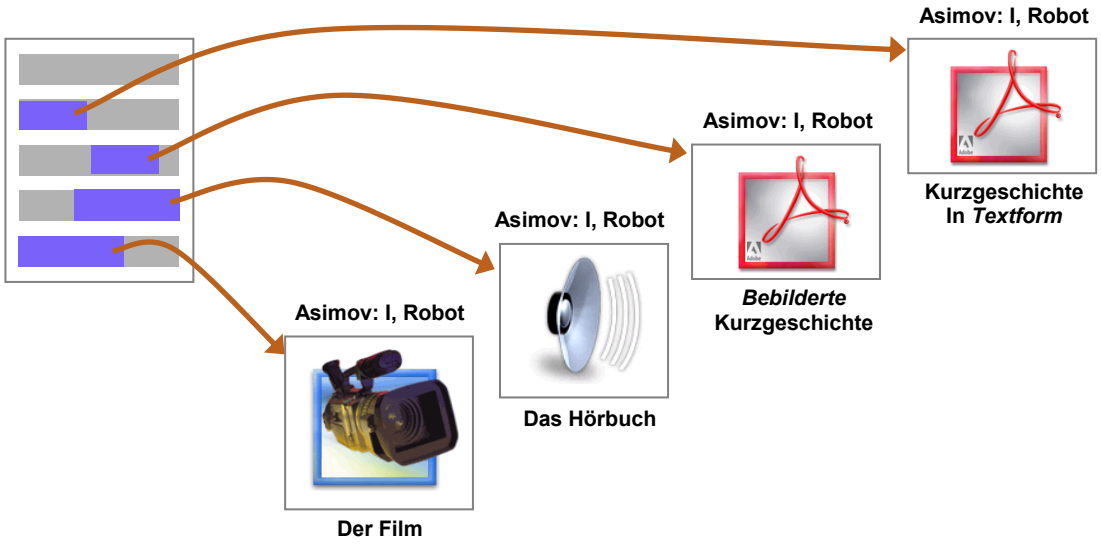


Abb. 37: Drei unterschiedliche Medientypen mit derselben Kurzgeschichte als Inhalt

Eine automatische Identifikation des Medientyps kann vergleichsweise problematisch sein. Zumeist ist aus dem technischen Typ bereits der *typische* Medientyp ableitbar, viele Dateiformate sind aber auch für unterschiedliche Medientypen einzusetzen (siehe z. B. Nation 1998). Beispielsweise kann eine PDF-Datei ausschließlich aus Bildern bestehen oder eine Flash-Datei zur Anzeige von Text verwendet werden. Zudem erlauben es viele Formate – wie auch HTML – weitere Objekte einzubinden, wodurch sich eine eindeutige mediale Klassifikation eines Dokuments weiter erschwert.

Eine Möglichkeit, um dem Anwender Hinweise auf den zu erwartenden Medientyp zu geben, besteht in Form einer „komprimierten" Vorschau: Ein Thumbnail – das verkleinerte Abbild des Dokuments – vermittelt häufig einen Eindruck des medialen Inhalts (Kaasten, Greenberg et al. 2001; Woodruff, Faulring et al. 2001). Es lassen sich auch Animationen in Thumbnails wiedergeben, und Audioausgaben können in methodischer Analogie kurz angespielt werden (s. unter anderem Albers & Bergman 1995; Goldman-Segall, Jonesson et al. 1996). Die automatische Generierung solcher Vorschauen für das Web ist aber nicht trivial und war bereits Schwerpunkt mehrerer wissenschaftlicher Arbeiten (Kopetzky & Mülhäuser 1999; Zühlsdorff 2002).

4.5.3.3 Der topologische Typ des Dokuments

Einige frühere Hypertext-Systeme boten dem Anwender zur Unterstützung der Navigation und Orientierung zusätzliche Werkzeuge in eigenen Fenstern. Zum Beispiel zeigten der „Session Manager" von *Hyper-G* (s. Anhang B.14) und der „Overview"-Client von *Intermedia* (s. Anhang B.10) immer eine Übersicht der lokalen hierarchischen Struktur. Demgegenüber verwendeten Systeme wie *HyperTies*, *ZOG/KMS* oder *HyperCard* denselben Anzeigebereich für Inhalte *und* Navigation (s. Anhänge B.4, B.8 und B.11).

Beim Web verhält es sich ähnlich: Die im Browser-Fenster angezeigten HTML-Seiten repräsentieren sowohl die Inhalte des Web und dienen gleichzeitig zur Navigation und

Orientierung. Allerdings haben die meisten Seiten einen bestimmten Schwerpunkt. Die Rolle einer Webseite bei der Systembenutzung hängt eng mit der Graphenstruktur des Hypertextes zusammen: Es wird nachfolgend daher vom *topologischen Typ* des Knotens gesprochen (Gibson, Kleinberg et al. 1998). Diese Klassifikation basiert auf zahlreichen Arbeiten aus dem Hypertext-Bereich und der Web-Forschung (Botafogo & Shneiderman 1991; Kahn 1995; Pirolli, Pitkow et al. 1996; Tauscher 1996: 73; Haas & Grams 1998b; Takano & Winograd 1998; Modha & Spangler 2000; Miles-Board, Carr et al. 2002).

Bei einem verteilten Informationssystem ist der topologische Typ aus *lokaler* (also bezüglich der Navigation innerhalb einer Website) und *globaler* Sicht (für die Site-übergreifende Navigation) zu unterscheiden (Pirolli, Pitkow et al. 1996; Amitay, Carmel et al. 2003). In Bezug auf die *lokale* Navigation sind zusammenfassend die folgenden Knotentypen charakterisierbar:

- *Reference Nodes* verfügen über einen umfangreichen Fließtext und haben zumeist mehr ein- als ausgehende Links. Da sie die Inhalte des System repräsentieren, spricht man auch von *Inhaltsseiten (Content Pages*, siehe: Botafogo & Shneiderman 1991).
- *Index Nodes (Indexseiten)* weisen im Verhältnis zum Textumfang viele Links zu anderen Seiten auf und dienen primär zur Navigation. Sie werden daher auch *Navigationsseiten* genannt (Botafogo & Shneiderman 1991; Pirolli, Pitkow et al. 1996). Dieser Typ ist weiter unterteilbar:
  - Die *Head Node* ist die Einstiegsseite in den Hypertext (Pirolli, Pitkow et al. 1996). Von hier aus erlangt der Benutzer in der Regel eine Übersicht über das gesamte Angebot. Im Web wird diese Seite häufig als *Homepage* oder *Startseite* bezeichnet.
  - *Landmark Pages* sind Übersichtsseiten von *Clustern* innerhalb der Hypertext-Struktur. Cluster sind thematisch zusammengehörige, eng vernetzte Gruppen von Knoten, die häufig einen Zweig innerhalb der hierarchischen Struktur des Systems repräsentieren (Tauscher 1996: 73; Takano & Winograd 1998). In der Regel verweisen viele strukturelle Links der Website auf eine *Landmark Page* (Kahn 1995).
- Ein weiterer Typ ist die *Functional Page*. Sie dient primär zum Zugriff auf bestimmte Dienste und erfordert Benutzereingaben, auf die vom System eine dynamische Ausgabe generiert wird (Tauscher 1996: 73) Beispiele sind die Eingabemaske einer Suchmaschine oder der Warenkorb des Online-Shops.

Der lokale topologische Typ eines Dokuments kann mittels einer Analyse der Link-Struktur und der Dokumenten-Inhalte einer Website bestimmt werden (Modha & Spangler 2000). Dabei wird in der Regel zwischen Links im Navigationsbereich der Seiten (wo sich primär *strukturelle Links* befinden, s. Abschnitt 4.5.1.1) und anderen Links unterschieden. Einige Verfahren teilen dafür die Dokumente in *Mikroseiten* auf, um die Bereiche und Links nach ihrer Funktion getrennt auszuwerten (Haas & Grams 1998b; Miles-Board, Carr et al. 2002).

Aufgrund der *Site-übergreifenden* Links wurden im Web zusätzlich verschiedene *globale topologische Typen* identifiziert (Amitay, Carmel et al. 2003). Die zwei bedeutendsten Typen sind *Authorities* und *Hubs*. Auf eine *Authority-Seite* verweisen besonders viele Links von anderen Websites, und *Hubs* verfügen über viele externe Links, insbesondere auf *Authorities* (Kleinberg 1999). Zur Berechnung dieser Seiten wird ein iterativer Algorithmus verwendet, der ausgehend von einer Ausgangsmenge von Webseiten (z. B. der Ergebnisliste einer Suchmaschine) zu einem *bestimmten Stichwort* derartige *besonders themenrelevante* Seiten bestimmt. Die so gewonnenen Dokumente erwiesen sich bei Evaluationen mit Anwendern als zumeist ebenso relevant und hochwertig wie die in Web-Verzeichnissen (z. B. *Yahoo!* oder *DMOZ*) redaktionell zusammengestellten Seiten (vergl. Abschnitt 3.3.3).

4.5.4 Informationen zum Zugriff auf Ressourcen im Web

Das Internet bietet aufgrund seiner paketorientierten Datenübermittlung und der zumeist redundanten Verbindungen zwischen den Netzwerkknoten einen insgesamt schnellen und zuverlässigen Zugang zu den Ressourcen des Webs, dennoch kommt es aus unterschiedlichen Gründen immer wieder zu Zugriffsproblemen. Dabei lässt sich grundlegend unterscheiden, ob ein Objekt *gar nicht verfügbar* ist (Abschnitt 4.5.4.1) oder der Abruf lediglich mit einer größeren *Latenzzeit* verbunden ist (Abschnitt 4.5.4.2).

4.5.4.1 Die Verfügbarkeit der Ressource

Zahlreiche technische Ursachen können den Zugang zu Objekten im Web verhindern. Da Anwender nicht gewarnt werden, bevor sie versuchen, auf ein (gerade) nicht verfügbares Dokument zuzugreifen, kommt es bei der Navigation relativ häufig zu Fehlermeldungen (s. Abschnitt 3.3.4). Aus software-ergonomischer Sicht ist ein solches Systemverhalten unbefriedigend und anachronistisch: Ein System sollte versuchen, Fehlerursachen im Vorfelde zu vermeiden und dem Anwender aktiv bei der Problemlösung helfen, anstatt ihn mit wenig hilfreichen Fehlermeldungen zu konfrontieren (Shneiderman 1997: 76ff).

Aus technischer Sicht setzt ein verbesserter Umgang mit nicht zugreifbaren Objekten voraus, dass dem Hypertext-Client entsprechende Daten zu den Link-Zielen rechtzeitig zur Verfügung stehen. Die Bereitstellung dieser Daten stellt im Web aufgrund der verteilten, losen Architektur eine Herausforderung dar, zumal unterschiedliche Ursachen den Zugang zu Informationen verhindern können:

- *Veraltete Links:* Das Web ist ständigen Änderungen unterworfen. Dokumente, die kürzlich noch unter einem bestimmten URI zu finden waren, sind plötzlich nicht mehr zugreifbar, wenn ein Anbieter sie ersetzt, umbenennt oder löscht. Es fehlen dem Web aber (standardisierte) Mechanismen zur Konsistenzsicherung von Links, Suchsystemen und anderen Navigationsmitteln wie History und Bookmarks (s. Abschnitt 3.3.4).
- *Die Erreichbarkeit des Dienstes:* Ein weiterer Grund für Zugriffsprobleme im Web sind nicht erreichbare Server. Es kann sich um eine temporäre Störung oder auch um ein dauer-

haftes Problem handeln, wenn z. B. eine Firma schließt. Temporäre Störungen haben meist technische Ursachen: Server werden gewartet, haben einen Defekt oder Netzwerkprobleme an entscheidenden Knotenpunkten verhindern die Datenübertragung.

- *Beschränkter Zugang:* Eine weitere Zugriffshürde können „geschützte Bereiche" stellen, die nur von Rechnern mit bestimmten IP-Adressen oder mit einem *Benutzerkonto* zugänglich sind. Solche Dokumente sind für Personen nutzlos, die nicht zur zugelassenen Anwendergruppe gehören.
- *Gebührenpflichtige Ressource:* Zahlreiche Dokumente und Dienste im Web sind gebührenpflichtig. Links zu solchen Seiten oder Diensten sind ebenfalls nicht ohne Weiteres zugreifbar.

Die unterschiedlichen Gründe für den fehlenden Zugriff sind auf verschiedene Art ermittelbar. „*Absolute*" Zugriffsprobleme wie dauerhaft gelöschte Dokumente oder abgeschaltete Webserver sind von keinem System aus erreichbar. Solche Fehlerursachen können daher von beliebigen Computern im Internet ermittelt und zur Verfügung gestellt werden. *Temporäre* oder *regionale* Probleme, die beispielsweise auf Netzwerkausfälle zurückzuführen sind oder von der IP-Adresse des persönlichen Computers abhängen, müssten zeitnah und lokal vom Web-Client oder einem System im Subnetz des Benutzers ermittelt werden. Kapitel 7.2 geht genauer auf mögliche Verfahren ein.

Ungelöst ist bis heute, wie bezüglich der Benutzungsschnittstelle mit Links umzugehen ist, deren Ziel nicht zugreifbar ist. Eine Möglichkeit besteht darin, „fehlerhafte" Links auszublenden, wie dies bei Hyper-G gemacht wurde (Dalitz & Heyer 1995). Ein solcher Umgang hat sich aber im Web als problematisch erwiesen, da Web-Nutzer bestimmte Objekte oder Texte häufig alleine aufgrund ihrer Positionierung oder ihrer grafischen Gestaltung als Link-Anker auffassen (Weinreich 1997; Spool, Scanlon et al. 1998).

Um den Anwender vor zeitweise unerreichbaren Objekten zu warnen, wurde stattdessen die Anzeige einer *Wahrscheinlichkeit* der Zugreifbarkeit vorgeschlagen. Diese Informationen könnten z. B. als *Prozentsatz* oder *grafisch* in Form eines Thermometers dargestellt werden (Fogg & Nielsen 2000). Weitere Konzepte zur Benutzungsschnittstelle fehlerhafter Links wurden im Rahmen dieser Arbeit entwickelt und evaluiert (s. Abschnitte 5.5.4.6 und 6.4ff).

4.5.4.2 Die Latenzzeit bei der Navigation

Die Reaktionszeit eines Software-Systems ist ein entscheidendes Kriterium für seine Benutzbarkeit. Bereits Antwortzeiten von über 2 bis 4 Sekunden führen zu messbar schlechteren Leistungen der Anwender. Für das Web können aber entsprechend geringe *Latenzzeiten* auf Benutzeraktionen *nicht* sichergestellt werden (s. Abschnitt 3.3.1).

Lange Wartezeiten gehen zudem von der produktiven Arbeitszeit des Benutzers verloren. Johnson betrachtet daher Verzögerungen bei der Web-Navigation als *mikroökonomischen Kostenfaktor* für Anwender und Unternehmen. Als Konsequenz fordert er, dass Nutzer die zu

erwartende Übertragungszeit und die daraus folgenden „Kosten“ einschätzen können müssen, *bevor* sie einem Link folgen (Johnson 1997a). Studien haben gezeigt, dass viele Personen den Transfer einer Seite abbrechen, wenn ihnen die Übertragung subjektiv zu lange dauert (Spool, Scanlon et al. 1998); als Konsequenz ist das entsprechende Objekt für den Leser sogar wertlos.

Die *Latenzzeit* ist aus diesen Gründen eine weitere bedeutsame Objekteigenschaft im Web. Insbesondere unerwartet lange Verzögerungen sollten dem Anwender *vor* dem Zugriff auf ein Objekt zugänglich sein: Er kann dann selbst in Erwägung ziehen, ob er die Wartezeit in Kauf nehmen möchte oder lieber nach einer Alternative Ausschau hält. Allerdings lässt sich die konkrete Antwortzeit im Web meist nur schwer vorhersagen, da sie von zahlreichen Variablen abhängig ist. Die drei folgenden Faktoren sind zumeist ausschlaggebend (vergl. Abschnitt 3.3.1):

- Die *Größe des Objekts* bestimmt die Menge der zu übertragenden Daten und beeinflusst damit die Dauer des Transfers. Die Dateigröße kann vom Webserver abgefragt werden und wird im HTTP-Header als „`Content-Length`“ angegeben (Fielding, Gettys et al. 1999). Bei Webseiten ist neben der HTML-Datei allerdings auch der Umfang aller eingebundenen Objekte zu berücksichtigen, deren Umfang sich nur einzeln abfragen lässt.
- Der zweite entscheidende Faktor ist die *aktuell verfügbare Bandbreite*. Sie hängt wiederum von anderen Variablen ab, wie dem Routing der einzelnen Pakete und der aktuellen Netzwerklast.[112]
- Der dritte wichtige Faktor ist die *Verzögerung (delay)*, die zwischen der Anfrage des Clients und der Antwort des Servers vergeht. Sie ist abhängig von der physikalischen Entfernung zwischen Client und Server sowie der Anzahl der Router auf dem Weg. Zusätzlich kann eine hohe Auslastung bereits von einem der beteiligten Systeme die Bearbeitung der Anfrage merklich verzögern.

Es gibt noch eine ganze Reihe weiterer Variablen, die die Latenzzeit im Web beeinflussen, wie die Zeit, die der Browser zum Rendern einer Seite benötigt.

Bedauernswerterweise ist, abgesehen von der Größe der Dateien, keiner dieser Faktoren zuverlässig vorherzusehen, zumal das gegenwärtig gebräuchliche *Network-Layer-Protokoll* des Internets *IPv4* keine *Quality-of-Service*-Merkmale bietet,[113] und die verfügbaren Bandbreiten

[112] Da die einzelnen Datenpakete unterschiedliche Pfade nehmen können, ist der Sachverhalt eigentlich noch komplexer. Dies stellt lediglich eine im Rahmen dieses Abschnitts zweckmäßig vereinfachte Sichtweise dar. Als weiterführende Literatur sei hier z. B. auf (Tanenbaum 2003, S. 431ff) verwiesen.

[113] In Kopf von IPv4-Datenpaketen gibt es ein 8-Bit-Feld mit der Bezeichnung „Type of Service“ (TOS), mit dem spezifiziert werden kann, welche Art von Dienst das Paket verkörpert. Vier dieser acht Bit wurden auch definiert, um für Pakete beispielsweise den *Delay* zu minimieren oder die *Zuverlässigkeit* zu maximieren; aber dies wurde kaum praxisrelevant, da die Angaben nur ungenau sind und zudem nur wenige Router sie überhaupt berücksichtigten. In dem sich langsam verbreitenden Nachfolger IPv6 ist dies allerdings besser geregelt, und es lassen sich u. a. Prioritäten für Pakete angeben (Postel 1981; Deering & Hinden 1998; Tanenbaum 2003, S. 434ff).

von einer unbekannten Anzahl von Systemen gemeinsam genutzt werden. Die einzige Möglichkeit besteht daher gegenwärtig darin, die zu erwartende Antwortzeit durch zeitnahe Tests abzuschätzen.

Ein Forschungsprojekt, das sich dieses Konzept bereits zunutze gemacht hat, ist die Browser-Erweiterung „Traffic-Lights“ (s. Abschnitt 5.3.2 und Abb. 51). Die Antwortzeit eines Servers wurde dabei geschätzt, indem für alle externen Links je ein HTTP-HEAD-Request zur Homepage der Site erfolgte. Die benötigte Antwortzeit wurde als Schätzung für alle weiteren Objekte der Site herangezogen. In einer Evaluation erwies sich die Darstellung dieser Information bei den Links für die Suche nach Informationen als hilfreich (Campbell & Maglio 1999).

4.5.5 Benutzungsinformationen im Web

In der physischen Welt spielt die frühere Nutzung von Informationsquellen wie Büchern und Zeitschriften oft eine bedeutende Rolle: Menschen kategorisieren und ordnen ihre Literatur bewusst oder unbewusst auf eine bestimmte Weise und können anhand von Merkmalen wie Lesezeichen, Eselsohren und Anmerkungen frühere Informationen einfacher erinnern und wiederfinden.

Bei den öffentlich zugänglichen Druckwerken einer Bibliothek zeigt sich ein weiterer Aspekt der Benutzung: Oft gelesene Publikationen zeigen Nutzungserscheinungen, die dem Leser Hinweise auf die Popularität eines Werkes geben, und neue Bücher sind oft auch als solche erkennbar. Dies kann die Entscheidung des Lesers zur Auswahl einer Lektüre beeinflussen. Bei gemeinsam genutzten Unterlagen kennzeichnen schriftliche Anmerkungen bedeutende Abschnitte und weisen auf Fehler und Ungenauigkeiten hin.

In der digitalen, virtuellen Informationswelt sind derartige direkte und indirekte *Nutzungsartefakte* bisher für die normalen Anwender nur selten zugänglich. Sowohl die Aufzeichnung der Benutzeraktionen als auch die Aufbereitung und Visualisierung von Benutzungsdaten setzen eine spezielle Softwareunterstützung voraus, deren Konzeption und Realisierung bereits Forschungsgegenstand zahlreicher Projekte war. Ebenso wie bei gegenständlichen Schriftstücken gibt es zwei Aspekte der Benutzung: die persönliche Nutzung (*Personal History*, folgender Abschnitt) und die Benutzung durch andere (*Social Navigation*, Abschnitt 4.5.5.2).

4.5.5.1 Personal History: die persönliche Benutzung des Informationssystems

Daten über die eigene Benutzung eines Systems sind für Menschen oft erst dann wichtig, wenn sie früher schon einmal zugegriffene Dokumente wiederfinden möchten. Mehrere Studien haben belegt, dass dies auch für Benutzer des Webs relevant ist, da sie recht häufig zu bereits besuchten Webseiten zurückkehren (Catledge & Pitkow 1995; Tauscher & Greenberg 1997: 113; Cockburn & McKenzie 2001). Webbrowser bieten dementsprechend

mehrere Werkzeuge, die den Zugang zu zuvor besuchten Seiten vereinfachen (s. Abschnitt 3.1.7).

Oft möchten Benutzer aber auch erkennen können, auf welche Dokumente sie bereits zugegriffen haben, um sie *nicht* noch einmal zu inspizieren oder „im Kreise" zu navigieren. Webbrowser zeigen daher an, welche Seiten der Anwender bereits aufgerufen hat, indem sie die Farbe der entsprechenden Link-Marker von Blau zu Violett ändern (s. Abschnitte 5.1.3 und 5.3.3). Diese Unterscheidbarkeit wird als ausgesprochen bedeutsam für die Benutzbarkeit des Webs angesehen, zumal es sich um eine der wenigen „Vorab-Informationen" handelt, die für die Navigation zur Verfügung stehen (Spool, Scanlon et al. 1998; Siegel 1999; Nielsen 2002; ISO9241-151 2008, 9.4.8). Es unterstützt das Gefühl der *Kontrollierbarkeit*, wenn ein System die Aktionen des Menschen widerspiegelt (Nielsen 2004a).

Die Methode der verschiedenfarbigen Link-Marker weist allerdings einige Schwächen auf: Viele Web-Autoren ändern das Farbschema von Webseiten, sodass dieser Link-Hinweis dem Anwender nicht mehr zur Verfügung steht, und bei grafischen Link-Ankern fehlt die farbliche Kennzeichnung ganz. Des Weiteren ist nicht zu erkennen, *wann* und *wie oft* ein Zielobjekt bereits angesehen wurde – es besteht ein wesentlicher Unterschied, ob man vor wenigen Minuten eine Seite besucht hat und zyklisch navigiert, oder ob ein Dokument vor mehreren Tagen gelesen wurde und genau die benötigte Information enthält.

Aus technischer Sicht lassen sich detaillierte Daten zur eigenen Web-Benutzung problemlos aufzeichnen und zugänglich machen.[114] Zahlreiche Projekte versuchen, die *Personal History* von Webbrowsern zu erweitern und den Zugriff auf früher besuchte Dokumente zu vereinfachen (s. Abschnitt 3.1.7); die Möglichkeiten zur Integration erweiterter History-Informationen in die Benutzungsschnittstelle von Links wurden bisher aber nicht untersucht.

4.5.5.2 Social Navigation: Aktionen und Bewertungen anderer Personen

Unter dem Begriff *Social Navigation* werden Konzepte zusammengefasst, bei denen sich Anwender während ihrer Interaktion mit Informationssystemen an dem Verhalten und den Hinweisen anderer Nutzer orientieren können (vergl. Abschnitt 3.1.6). Diese Hinweise entstehen entweder *indirekt* als Spuren vergangener Navigationsaktivitäten oder *direkt*, indem Benutzer bewusst im Informationsraum Artefakte wie Bewertungen oder Kommentare hinterlassen. Neben derartigen *asynchronen* Kollaborationsformen gibt es auch *synchrone* Social-Navigation-Konzepte, bei denen die Anwender direkt online miteinander kommunizieren. Beispielsweise kann ihnen ein Online-Chat angeboten werden, wenn sie sich aufgrund ihrer Aufgaben oder der besuchten Seiten in virtueller „Nähe" zueinander befinden (Dourish & Chalmers 1994; Dieberger, Dourish et al. 2000).

---

[114] Bereits *NoteCards* protokollierte alle Aktionen der Leser und Autoren. Dabei konnten die Anwender spezifizieren, welche Benutzungs-Ereignisse aufgezeichnet werden sollten (Nielsen 1993a).

Social Navigation macht das Potenzial gemeinschaftlichen Agierens auch in digitalen Informationssystemen zugänglich. Die bisher rein menschliche Kompetenz zur inhaltlichen und qualitativen Wertung von Informationen wird so in virtuelle Informationsräume eingebracht. Aus technischer Perspektive sind zwei Basiskonzepte zur Realisierung solcher Systeme etabliert. Erstens wurden zahlreiche Social-Navigation-Anwendungen für einzelne Webserver realisiert. Dort sind die Aktionen und Eingaben sämtlicher Benutzer des Systems technisch verfügbar, und sie lassen sich relativ einfach aufzeichnen und auswerten. Ein populäres Beispiel ist das Online-Kaufhaus *Amazon*, das Benutzern für alle Produkte Links zu weiteren Waren anbietet, die andere Kunden mit ähnlichem Profil erworben haben und dass zudem die Möglichkeit bietet, Rezensionen zu den angebotenen Produkten abzugeben.

Die zweite technische Realisierungsmöglichkeit benötigt einen speziellen Dienst, der zum Datenaustausch dient und den serverübergreifenden Einsatz des Social-Navigation-Systems ermöglicht. Drei Beispiele zeigen die Potenziale solcher Dienste:

- *Annotea* ist ein Forschungsprojekt des W3C, das das Web um gemeinsame Metadaten in Form von Bookmarks und Annotationen erweitert (Kahan & Koivunen 2001). Es basiert auf offenen Standards wie RDF (Berners-Lee, Hendler et al. 2001) und XPointer (DeRose, Daniel et al. 2002). Implementiert wurde es für den W3C-Referenz-Browser *Amaya*[115] und als Firefox-Erweiterung.[116]
- *CritLink* benötigt keine Browser-Erweiterung, sondern basiert auf einem Intermediary (s. Abschnitt 8.4), der alle Dokumente des Webs filtert und zusätzliche Kommentare und Links anderer Benutzer ergänzt (s. Abb. 42; Yee 2002).
- Das *Social Web Cockpit* bietet zusätzlich Möglichkeiten zur *synchronen* Kommunikation. Die Benutzer ordnen sich bei diesem Dienst bestimmten Interessengruppen zu, die mit Websites assoziiert werden können. Wenn immer ein Benutzer auf eine entsprechende Seite gelangt, kann er über das Social Web Cockpit mit anderen Gruppenmitgliedern in Kontakt treten und direkt Informationen und Daten austauschen, wie Anmerkungen und Links zu weiterführenden Dokumenten (Gräther & Prinz 2001).

Eine Reihe von Herausforderungen erschwert allerdings die Einführung neuer Social-Navigation-Anwendungen. Das *Kaltstartproblem* bezeichnet den Umstand, dass bei Einführung eines neuen Dienstes nur wenige hilfreiche Informationen vorliegen und somit wenige Anwender eigenen Nutzen darin sehen. Das *Sparcity-Problem* geht auf den Umstand ein, dass bei Informationssystemen mit sehr vielen Objekten sich nur selten Benutzer mit ähnlichen Profilen im System „treffen“. Diese und weitere Schwierigkeiten sowie mögliche Lösungsansätze wurden in (Baier, Weinreich et al. 2004) eingehender diskutiert.

---

[115] Informationen zum Amaya-Browser werden vom W3C angeboten: http://www.w3.org/Amaya/. Weitere Informationen zum Annotea-Projekt befinden sich unter http://www.w3.org/2001/Annotea/.

[116] Das Projekt nennt sich Annozilla: http://annozilla.mozdev.org/.

Einige der Schwächen des Webs, wie die mangelnde Qualitätssicherung und die Probleme bei der Suche nach nützlichen und zuverlässigen Informationen, deuten auf die Bedeutsamkeit solcher Social-Navigation-Hinweise für die Web-Navigation hin. Viele Nutzer tragen gerne zu Gestaltung und Qualitätssicherung des Webs bei, wie sich in der großen Popularität von *Web-Foren*, *WikiWikiWebs* und den vielen *Social-Network-Plattformen* im Web zeigt (Leuf & Cunningham 2001; Obendorf 2004).

4.5.6 Bedeutung des URI als Element der Benutzungsschnittstelle

Der *URI (Uniform Resource Identifier*, siehe: Berners-Lee, Fielding et al. 2005) dient zur Identifikation aller Ressourcen im Web. Er wird in dieser Klassifikation als eigene Kategorie impliziter Objekt-Eigenschaften aufgeführt, da er besondere Bedeutung bei der Benutzung des Webs hat. Obwohl der URI primär ein technisches Adressierungsschema ist (Berners-Lee, Fielding et al. 2005), wurde er von Tim Berners-Lee so konzipiert, dass er auch für Menschen lesbar ist und dem erfahrenen Anwender verschiedene Hinweise zum Zielobjekt gibt (Berners-Lee 1996). Hierzu gehört, dass:

- der Domain-Name auf den *Anbieter* der Informationen hinweist (vergl. Abschnitte 3.3.3 und 4.5.2.2),
- der Pfad (*Path*) häufig die *Position* innerhalb der Struktur der Website widerspiegelt,
- der Dateiname (*File*) Hinweise auf den *Inhalt* des Dokuments gibt und
- die Erweiterung des Dateinamens (*Extension*) den Dateityp spezifiziert.

Der URI eines Link-Zieles wird von den meisten Browsern unten im Browser-Fenster dargestellt, sobald der Benutzer mit dem Mauszeiger über einen Link fährt (s. Abschnitt 5.3.5). Da dies zumeist die einzige *zusätzliche* Information ist, die dem Anwender zum Zielobjekt vorliegt, sehen ihn insbesondere erfahrene Benutzer als wichtige Hilfe für die Navigation an (Spool, Scanlon et al. 1998; Weinreich & Lamersdorf 2003; Lin, Greenberg et al. 2011). Jakob Nielsen hatte daher bereits 1996 „komplexe“ und missverständliche URIs als einen der „Top-Ten-Fehler“ im Web-Design beschrieben (Nielsen 1996) und später Richtlinien für die Gestaltung *„ergonomischer“* URIs aufgestellt (Nielsen 1999b).

Neben der Navigationsunterstützung für Links gibt es weitere wichtige Gründe für lesbare URIs: Sie werden häufig notiert, abgedruckt, wieder eingetippt oder per E-Mail versandt (Jones, Dumais et al. 2002). Der URI erlaubt es darüber hinaus, Dokumente wiederzuerkennen. In Benutzerstudien haben sich verständliche URIs zum Erinnern von Dokumenten als ähnlich nützlich erwiesen wie der Titel oder ein Thumbnail der Seite (Cockburn & Greenberg 1999); (Kaasten, Greenberg et al. 2001). Zahlreiche Benutzer achten insbesondere auf den Domain-Namen, um gefälschte „*Phishing-Sites*“ zu erkennen (Lin, Greenberg et al. 2011).

Leider verwenden immer noch viele Content-Management-Systeme „kryptische“ URIs, in denen nur technische Parameter verwendet werden. Sind zusätzlich in der Adresse Benut-

zungs- und Sitzungsinformationen codiert, so ändert sogar sich der URI bei jedem Besuch und viele Navigationswerkzeuge versagen.

4.6 Zusammenfassung

Die in diesem Abschnitt vorgestellte Klassifikation impliziter Eigenschaften von Links und Knoten-Objekten im Web hat die Vielfältigkeit der Verknüpfungsmöglichkeiten aufgezeigt. Die Klassifikation dient als eine Grundlage für die in dieser Arbeit vorgestellten Methoden zur Erweiterung der Link-Benutzungsschnittstelle und für die darauf basierenden Prototypen. Die Klassifikation lässt sich – bis auf den URI – weitgehend auf andere verteilte Hypertext-Informationssysteme übertragen, zumal sie auf grundlegenden Erkenntnissen der Informationswissenschaft und auf früheren Arbeiten aus dem Hypertext-Bereich aufbaut. Zahlreiche der vorgestellten Daten sind als Attribute des Links und des Link-Ziels bereits verfügbar oder mit vertretbarem Aufwand softwaretechnisch zu ermitteln und können somit für Link-Hinweise und Previews herangezogen werden. Wenn dies automatisch (z. B. durch den Webbrowser) geschieht, stellen diese Informationen nicht nur eine potenzielle Hilfe für die Nutzer, sondern auch für die Autoren dar, weil sie dann entsprechende Angaben zu „besonderen“ Link-Zielen – beispielsweise den Dateityp oder die Größe eines Downloads – nicht mehr selbst zu ergänzen haben.

Die hier vorgestellten Link- und Knoten-Eigenschaften sind aller Voraussicht nach von unterschiedlicher Bedeutung für die Benutzung des Webs. Daher wurden sie mit Thinking-Aloud-Tests zweier Prototypen und in einer Online-Umfrage evaluiert (s. Kapitel 6). Zuvor wird im folgenden Kapitel 5 auf die Benutzungsschnittstelle von Links aus einer anderen Perspektive eingegangen: Es werden Konzepte zur Visualisierung von Link-Ankern und zur Darstellung zusätzlicher Link-Informationen vorgestellt. Diese Analyse dient als eine weitere Basis für die in dieser Arbeit entwickelten Methoden zur Erweiterung der Link-Benutzungsschnittstelle.

5 Konzeption einer erweiterten Benutzungsschnittstelle für Hyperlinks

Der vorherige Abschnitt befasste sich mit den grundlegenden konzeptionellen Möglichkeiten zur Reduktion der Benutzbarkeitsprobleme von Links in verteilten Hypertext-Informationssystemen. Bisher wurde aber nur am Rande darauf eingegangen, wie sich diese Konzepte bezüglich der *Benutzungsschnittstelle von Links*, also der **Gestaltung von Link-Ankern** und der **Interaktion mit Links**, konkretisieren lassen. Dieses Kapitel beschäftigt sich daher systematisch mit den Anforderungen und Zielen bei der Gestaltung der Benutzungsschnittstelle von Hyperlinks und stellt Klassifikationen für zwei wesentliche Aspekte der Link-Schnittstelle vor (Landow 1987; Weinreich, Obendorf et al. 2001): Die Methoden zur *Hervorhebung von Link-Ankern* und die Möglichkeiten zur *Visualisierung von* (zusätzlichen) *Link-Informationen*, wie einem Link-Typ oder Link-Preview. Am Beispiel des Webs werden die Stärken und Schwächen der jeweiligen Methoden für die Anwendung in verteilten Hypertext-Informationssystemen analysiert und auf Basis dieser Ergebnisse neue Schnittstellenkonzepte für Hyperlinks entwickelt.

Der erste Abschnitt 5.1 identifiziert die Ziele für die Gestaltung benutzbarer Hyperlinks in Informationssystemen und spezifiziert die sich hieraus ergebenden wesentlichen Anforderungen. Diese Anforderungen werden mit den Eigenschaften des aktuellen *De-facto-Standards* für Link-Marker und der *ISO-Norm 9241-151* zur Gestaltung von Benutzungsschnittstellen im Web aus dem Jahr 2008 verglichen. Danach werden die verfügbaren Methoden zur Visualisierung von Link-Ankern (Abschnitt 5.2) und zur Darstellung zusätzlicher Link-Informationen kategorisiert (Abschnitt 5.3) und ihre jeweiligen Stärken und Schwächen analysiert. Aus den Analysen geht ein neues erweitertes Schnittstellenkonzept für Hyperlinks mit dem Namen *HyperScout I* hervor, das in Abschnitt 5.5 vorgestellt wird.

5.1 Die Benutzungsschnittstelle von Hyperlinks

Die Methode zur Hervorhebung von Link-Ankern ist ein bedeutsamer Faktor für das Erscheinungsbild eines Hypertext-Dokuments und die einfache Identifizierbarkeit der Links. In den unterschiedlichen Hypertext-Systemen (s. Anhang B) wurden und werden verschiedene Konzepte zur Kennzeichnung von *Link-Ankern* durch die sogenannten *Link-Marker* eingesetzt (s. Kapitel 2.2.1).

Dieses Kapitel geht erst auf die *Ziele* bei der Gestaltung der Benutzungsschnittstelle von Hyperlinks ein. Es geht dabei nicht um Fragen eines ansprechenden Designs, sondern um die Angemessenheit für die Kriterien offener, verteilter Hypertext-Informationssysteme, die besondere Anforderungen an die Link-Benutzungsschnittstelle stellen. Den aktuellen De-facto-Standard für Link-Marker prägten die ersten grafischen Webbrowser und er ist bis heute allgegenwärtig. Die farbigen und oft unterstrichenen Link-Anker des Webs weisen

aber zahlreiche Schwächen für die Verwendung in verteilten Hypertext-Informationssystemen auf, die sich auch in den zusätzlichen Anforderungen der ISO-Norm 9241-151 aus dem Jahr 2008 zur Gestaltung von Web-Benutzungsschnittstellen widerspiegeln.

5.1.1 Ziele bei der Gestaltung von Link-Markern für eingebettete, assoziative Links

Bei der Gestaltung von Link-Markern müssen mehrere zum Teil gegensätzliche Ziele miteinander vereinbart werden: Auf der einen Seite sollte eine möglichst *originalgetreue* Darstellung des Dokuments erfolgen und die *Lesbarkeit* des Textes nicht eingeschränkt werden. Die Hervorhebung von im Fließtext eingebetteten Link-Ankern kann dem entgegenstehen und den Lesefluss des Benutzers stören. Es ist zudem zu vermeiden, dass Benutzer zum häufigen Klicken auf Links animiert werden, da sie dies vom effizienten Lesen eines Textes abhält („*click happy behavior*“, siehe: Roby 1999: 98). Eine im Rahmen dieser Arbeit durchgeführte Studie hat gerade dieses Verhalten bei den immer sichtbaren und auffallend hervorgehobenen Link-Ankern im Web aufgezeigt (Obendorf & Weinreich 2003). In einer weiteren Studie wurde die häufig sehr kurze Verweilzeit auf Webseiten belegt (Weinreich, Obendorf et al. 2006b), die auf das nach Links „scannende“ Verhalten der Benutzer im Web hinweist, das auch von anderen Forschern beobachtet wurde (Vora, Helander et al. 1994; Morkes & Nielsen 1997; Nielsen 2008).

Auf der anderen Seite müssen Link-Anker eindeutig identifizierbar sein, damit Benutzer sie lokalisieren und auswählen können. Landow sieht dabei eine Analogie zwischen der Hypertext-Navigation und einem Reisenden und fordert als Konsequenz eine *Rhetoric of Hypertext*. Diese teilt er in zwei Kernaspekte auf: Eine *Rhetoric of Departure* ist erforderlich, um den Leser beim Startpunkt eines Links auf das Folgen der Verweisung vorzubereiten, indem er entsprechend informiert wird. Die *Rhetoric of Arrival* soll ihm hingegen bei der Ankunft am Link-Ziel ein „Gefühl des Willkommens“ geben. Hierzu gehört, dass er das *Link-Ziel* eindeutig identifizieren kann[117] und dass ihm vermittelt wird, welche Teile des Dokuments für das Verständnis des Ziels relevant sind. Dies kann beispielsweise durch einen *Zielkontext* erreicht werden, der einen erweiterten, inhaltlich relevanten Textabschnitt zum Zielanker kennzeichnet (Landow 1987; Landow 1997: 150).

Landow führte später die Idee der *Rhetoric of Departure* mit Kahn und Peters weiter aus. Sie propagierten eine *visuelle Rhetorik* für Hypertext-Systeme, damit eine „visuelle Kommunikation“ zwischen dem System und dem Benutzer entsteht (Kahn, Peters et al. 1995). Die zwei grundlegenden Elemente dieser visuellen Rhetorik nannten sie *Link Presence* und *Link Destination*:

[117] Dies bezieht sich (auch) auf Zielanker, die innerhalb von Dokumenten liegen. Heutige Webbrowser und einige frühere Hypertext-Systeme verzichten zumeist auf eine solche Hervorhebung von Zielankern in Dokumenten. Eine Ausnahme sind spätere Versionen des *Guide-Systems* (s. Anhang B.7), das einen kurz aufblinkenden Rahmen um den Zielanker zeichnete. Bei Webbrowsern gab es eine Hervorhebung von Zielankern lediglich bei Mosaic 2.0, der den Anker als invertierten Text darstellte. Erst mit der Einführung von CSS3 werden wieder ähnliche Möglichkeiten für Webbrowser eingeführt (s. Abschnitt 4.4.4).

- *Link Presence* beschreibt, *wo* die Link-Anker im Dokument sind: Der Benutzer soll genau erkennen können, an welcher Stelle ein Link-Anker anfängt, wo er aufhört und welchen Bereich er anklicken kann.
- *Link Destination* dient als Navigations- und Orientierungshilfe und soll dem Anwender vermitteln, *wohin* ein Link führt. Der Benutzer muss eindeutig über Funktion und Ziel des Links informiert werden, damit er entscheiden kann, ob er ihm folgen will oder nicht. Dabei weisen die Autoren bereits auf die zusätzlichen Herausforderungen von komplexen Verknüpfungsmöglichkeiten, z. B. bei multiplen Links, hin.

Die *Rhetoric of Departure* mit *Link Presence* und *Link Destination* dienen in dieser Arbeit als wesentliche Konzepte der Benutzungsschnittstelle von Links. In den Abschnitten 5.2 und 5.3 werden die verfügbaren Schnittstellentechniken beschrieben und ihre jeweiligen Potenziale und Grenzen in Bezug auf die im folgenden Abschnitt 5.1.2 dargestellten Anforderungen offener, verteilter Hypertext-Informationssysteme analysiert. Dabei stehen jeweils Benutzer ohne physische, situative oder technische Einschränkungen im Fokus, die einen Computer mit einer grafischen Benutzungsoberfläche verwenden. Für Anwender mit physischen Einschränkungen (wie einer Sehschwäche) oder mit speziellen Arbeitsvoraussetzungen (z. B. bei der Nutzung von Mobilgeräten) wären jeweils weitere Faktoren zu berücksichtigen gewesen, die aufgrund der anderen Thematik in dieser Arbeit ausgegrenzt wurden.

5.1.2 Anforderungen an die Link-Schnittstelle in offenen, verteilten Hypertext-Informationssystemen

Eine ganze Reihe von konzeptionellen Herausforderungen müssen bei der Gestaltung der Benutzungsschnittstelle von Links berücksichtigt werden, wenn sie den im Folgenden aufgeführten Anforderungen offener, verteilter Hypertext-Informationssysteme genügen sollen (vergl. Abschnitt 2.2.2.2 und Anhang B). Dies bezieht sich insbesondere auf die Kennzeichnung von Lnk-Ankern durch die sogenannten Link-Marker. Hierzu gehört, dass Link-Marker differenzierbar sein müssen, wenn sie *aneinandergrenzen*, über *mehrere Zeilen gehen* oder sich mehrere Anker-Bereiche *überlappen*. Link-Marker dürfen auch bei *hoher Link-Dichte* das Dokument nicht zu sehr dominieren und müssen den Anforderungen *erweiterter Link-Techniken* genügen. Einige Systeme erfordern zudem eine Unterscheidung bei der Darstellung von *Link-Ankern* und *Link-Kontexten*.

Angrenzende Link-Anker

Wenn zwei Link-Anker aneinandergrenzen, muss für den Benutzer dennoch erkennbar sein, wo die Link-Marker jeweils beginnen und enden. Das Problem ist auch im Web zu beobachten: Da Anfang und Ende von Link-Markern nicht explizit gekennzeichnet sind, ist die Trennung zweier aufeinanderfolgender Anker schwierig (Spool, Scanlon et al. 1998; s. auch ISO9241-151 2008, 9.4.3). Ein verwandtes Problem ergibt sich bei Link-Ankern, die sich über

mehrere Zeilen erstrecken, da nicht zu erkennen ist, ob ein Anker fortgeführt wird oder in der nächsten Zeile ein neuer Anker beginnt (s. Abb. 38).

Erster Link-Anker Zweiter Link-Anker Dritter
Link-Anker und etwas Text.

Abb. 38: Ein Text mit angrenzenden und mehrzeiligen Ankern und einer hohen Anker-Dichte.

Überlappende Link-Anker

Eine weitere Herausforderung bei der Gestaltung von Link-Markern sind *überlappende Link-Anker*. Eines der wenigen Informationssysteme, das überlappende Anker unterstützte, war *Hyper-G* (Keep, McLaughlin et al. 1993). Der Client *Harmony* verwendete verschiedene Hintergrundfarben, die bei Überschneidungen in horizontale Streifen aufgeteilt wurden (s. Abb. 39).

You are now accessing the Hyper–G server at the IICM (Institute for Information Processing and Computer Supported New Media) of Graz University of Technology, Austria. The IICM information server is the home server of J.UCS (Journal of Universal Computer Science).

Abb. 39: Link-Marker für zwei überlappende Link-Anker in Hyper-G (nach Andrews 1996).

Überlappende Anker gibt es im Web bisher nicht, da HTML sie syntaktisch nicht unterstützt. Bei offenen Hypertext-Systemen kann es aber relativ leicht zu überlappenden Ankern kommen, beispielsweise wenn mehrere Autoren Links zu einem Dokument hinzufügen.

Die Dichte der Link-Anker

In einem offenen, verteilten Hypertext-Informationssystem können die einzelnen Dokumente mit sehr vielen ein- und ausgehenden Links verknüpft sein. Hierzu kommt es beispielsweise, wenn viele Autoren Links zu einem Dokument hinzufügen. Gebrauchstaugliche Link-Marker sollten daher auch bei einer *hohen Dichte* von Link-Ankern effizient einsetzbar sein.

Eine hohe Link-Dichte wurde bei mehreren Hypertext-Projekten als Problem identifiziert, da eine Hervorhebung der Anker keinen Sinn mehr macht, wenn sie überhandnimmt: „After all, if everything is highlighted, then nothing is really highlighted anyway“ (Nielsen 1993a: 114). Daher empfahlen einige Hypertext-Experten bereits in den 1990er Jahren, nicht mehr als 7-10 Links pro Bildschirmseite anzubieten (Tomek & Maurer 1992: 114; Ansel Suter 1995: 140) oder maximal 10% des Textes durch Link-Marker hervorzuheben[118] (Kappe 1991). Ansonsten besteht die Gefahr des *Link Overload* (vergl. Abschnitt 3.3.2 und ISO9241-151 2008, 9.4.15).

[118] Die Mehrzahl der Webseiten hat zwar deutlich mehr als 7-10 Links, diese befinden sich aber zumeist in gesonderten Navigationsbereichen und nicht im Fließtext der Seite (Haas & Grams 1998a; Miles-Board, Carr et al. 2002).

Die heute gebräuchliche Methode zur Hervorhebung von Link-Ankern (s. Abschnitt 5.1.3) ist für eine hohe Anker-Dichte im Fließtext wenig geeignet, da sie die Lesbarkeit des Textes reduziert (s. typografische Link-Marker 5.2.1). Für das Web wird daher eine Beschränkung der Anzahl von Links im Inhaltsbereich der Seiten empfohlen (Lynch & Horton 1997; Lynch & Horton 2002). Als Konsequenz führen diverse Anbieter (beispielsweise einige populäre Online-Magazine[119]) Verweise auf externe Quellen in gesonderten Bereichen oder erst am Ende des Dokuments auf, statt sie im Fließtext einzubetten.

Erweiterte Hyperlink-Techniken

Zusätzliche Herausforderungen bei der Gestaltung von Link-Markern stellen die erweiterten Möglichkeiten fortschrittlicher offener Hypertext-Systeme (s. Abschnitt 2.2.2.2) und der Hypertext-Standards (s. Anhang C). Ungeklärt ist bisher beispielsweise, wie Link-Marker für *multiple Links* (s. Abschnitt 2.2.1) gestaltet werden können, damit Anwender erkennen, dass sie eine Auswahl erwartet.[120] Bei *bidirektionalen Links* (s. Abschnitt 2.2.1) ist es notwendig, Start- und Zielanker zu unterscheiden, sofern sie gerichtet sind. Weitere Herausforderungen stellen *typisierte Links*, da die Link-Typen für Benutzer nur dann bei der Navigation hilfreich sind, wenn diese Zusatzinformation auch verfügbar sind. Ebenso sollte der Anwender Links aus verschiedenen Link-Datenbanken oder von mehreren Autoren unterscheiden können.

Der Link-Kontext

Eine weitere Schwierigkeit stellt die Unterscheidung von *Link-Ankern* – als dem aktiven, anklickbaren Bereich – und dem sogenannten *Link-Kontext*[121] dar (s. Abschnitt 2.2.1). Eine derartige Erweiterung des Konzepts von Hyperlinks ist insbesondere bei im Fließtext eingebetteten Ankern sinnvoll, da es dem Leser eine genauere Idee über das Link-Ziel vermittelt (vergl. Weinreich, Obendorf et al. 2001). Sie sind darüber hinaus für Zielanker nützlich, um dem Benutzer im Sinne der *Rhetoric of Arrival* den zum Verständnis relevanten Bereich eines Dokuments anzuzeigen. Die Unterscheidbarkeit von Link-Anker und Link-Kontext setzt zwei Arten der Hervorhebung voraus.

5.1.3 Der gegenwärtige „Standard" für Link-Marker

Obwohl die Visualisierungsmethode für Link-Anker entscheidenden Einfluss auf das Erscheinungsbild eines Hypertext-Dokuments und die Benutzbarkeit des Informationssystems

---

[119] Beispiele sind hierfür *Spiegel Online* (www.spiegel.de) und der Heise Newsticker (www.heise.de/newsticker/).

[120] Im Web findet man eine Variante solcher multipler Links inzwischen im Navigationsbereich von Websites relativ häufig: Sobald der Benutzer mit dem Mauszeiger über einen aktiven Bereich kommt, klappt ein Menü mit einer Auswahl zusätzlicher Links auf. Diese Technik kann Navigationsschritte einsparen, gleichzeitig sind Probleme bei der Benutzung solcher „Drop-Down"-Menüs im Web bekannt, sofern sie für Benutzer nicht eindeutig als solche zu identifizieren sind (Ojakaar 2001).

[121] Der Link-Kontext ist ein inhaltlich zum Link-Anker gehöriger Bereich, den ein Benutzer berücksichtigen sollte, um Sinn und Funktion eines Links zu verstehen (s. Abschnitte 2.2.1 und 5.1.2). Als bisher einziges System bot *MacWeb* eine solche Unterscheidung (Nanard & Nanard 1993).

haben, setzten sich bisher nur wenige Untersuchungen mit der Gestaltung der Schnittstelle von Hyperlinks auseinander. Darüber hinaus existieren für Link-Marker keine systematisch entwickelten oder offiziellen Standards (vergl. Anhang C).

Der momentane *De-facto-Standard* für die Hervorhebung von Link-Ankern ist auf die Dominanz des Webs zurückzuführen: Gegenwärtig verwenden alle populären grafischen Webbrowser für textliche Link-Anker unterstrichene, farbige (zumeist blaue) Schrift.[122] Die Omnipräsenz des Webs hat dazu geführt, dass sich inzwischen auch viele andere Systeme an dieser Methode orientieren und aktive Textbereiche entsprechend kennzeichnen. Hierzu gehören Hilfesysteme sowie Werkzeuge von grafischen Benutzungsschnittstellen.[123]

Dabei ist dieser Standard eher ein Zufallsprodukt als das Ergebnis eines wohlüberlegten und systematisch vollzogenen Designprozesses. Es geht bis auf den ersten Web Browser „Nexus" von Tim Berners-Lee zurück, der Links auf seiner monochromen *NeXT-Workstation* durch Unterstreichung hervorhob (s. Abb. 40; Berners-Lee & Fischetti 2000). *Mosaic* führte mehrere Jahre später zusätzlich für Farbdisplays die blaue Textfarbe für Link-Marker ein (s. Abschnitt 5.2.2 und Abb. 55). Der Grund für diese Arten der Hervorhebung von textlichen Links war dabei höchstwahrscheinlich technisch begründet:

- Unterstrichener Text eignete sich für die damals weitverbreiteten Schwarz/Weiß-Bildschirme bei Workstations zur Hervorhebung von Text. Zudem war diese Methode einfach zu implementieren, da es als Textattribut von den gängigen Grafikbibliotheken (für *NEXTStep* und *X Windows*) zur Verfügung gestellt wurde (Berners-Lee & Fischetti 2000).
- Systeme mit Farbbildschirmen boten zu dieser Zeit häufig nur eine vordefinierte Palette von 16 Grundfarben, von denen Blau die am dunkelsten erscheinende Farbe war, die neben dem schwarzen Text am wenigsten herausstach und vor hellem Hintergrund am besten lesbar war (vergl. NCSA 1996; Weinreich, Obendorf et al. 2001).

[122] Es gibt einige Studien, die sich mit diesem und anderen De-facto-Standards im Webdesign auseinandersetzen: (Nielsen 1999c; Kangas 2001; Adkisson 2002; Bernard 2002; Kalbach & Bosenick 2003; Nielsen 2004c). Nach einer Studie von Microsoft soll ein bestimmtes Blau (#0044CC) sogar von Benutzern erheblich häufiger angeklickt werden als andere Farben und so Microsofts Suchmaschine Bing pro Jahr 80 bis 90 Millionen Dollar zusätzliche Werbeeinnahmen bescheren (Chan 2010).

[123] Sogar der Datei-Explorer von Microsofts Windows XP lässt sich so konfigurieren, dass sämtliche Dateinamen unterstrichen dargestellt werden. Programme und Dokumente lassen sich dann – ebenso wie Links auf Webseiten – mit einem einfachen Klick statt einem Doppelklick öffnen.

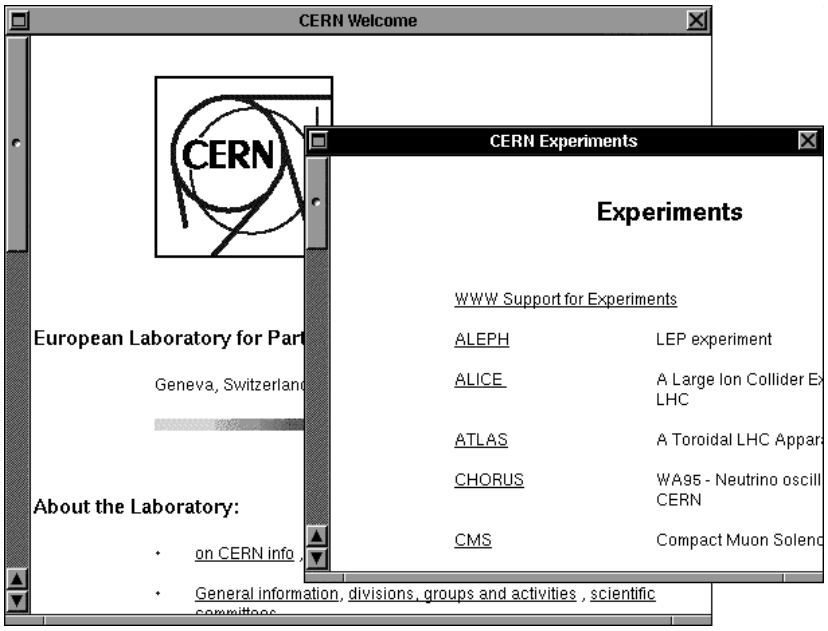


Abb. 40: Hyperlinks in Browser Nexus von Tim Berners-Lee (nach CERN 2008).

Obwohl eine systemübergreifend konsistente Benutzungsschnittstelle begrüßenswert ist, weist dieser Standard einige Nachteile und Einschränkungen auf: Unterstreichungen und blaue Textfarbe können die Lesbarkeit eines Textes reduzieren (s. Abschnitte 5.2.1 und 5.2.2), zudem werden viele Anforderungen an Link-Marker für offene, verteilte Hypertext-Systeme mit komplexen Links nicht erfüllt (s. Abschnitt 5.1.2); so lassen sich beispielsweise nebeneinanderliegende Links optisch kaum trennen, überlappende Link-Anker sind nicht darstellbar und es existieren keine standardisierten Möglichkeiten zur Kennzeichnung unterschiedlicher Link-Typen. Auf diese Weise behindert der De-facto-Standard in gewissem Maße sogar die Einführung neuer Verknüpfungsmöglichkeiten im Web.

5.1.4 ISO 9241-151: Norm zur Gestaltung von Schnittstellen im Web

Im Jahr 2008 wurde eine Norm für die Gestaltung von Schnittstellen im Web verabschiedet, die neben zahlreichen allgemeinen Anforderungen für die Umsetzung webbasierter Informationsangebote auch Kriterien für benutzergerechte Hyperlinks definiert (ISO9241-151 2008). Die Norm greift dabei mehrere der im vorherigen Abschnitt aufgeführten Defizite von Hyperlinks im Web auf und versucht, das Informationsdefizit der Benutzer bei der Link-Navigation (s. Abschnitte 3.2.1 und 3.4) zu reduzieren.

Zum einen wird gefordert, dass Links für Benutzer einfach und eindeutig zu erkennen sein müssen, angrenzende Links optisch voneinander getrennt sind und sich verschiedene Link-Arten (z. B. Navigations-Links gegenüber Aktions-Links in Online-Applikationen, vergl. Abschnitt 4.5.1.1) gut unterscheiden lassen (ISO9241-151 2008, 9.4.2ff). Auf welche Weise dies zu erreichen ist, wird allerdings nicht definiert.

Des Weiteren fordert die Norm, dass Links selbsterklärende Link-Hinweise („*link cues*“) bieten, sodass der Benutzer vor Auswahl eines Links weiß, was ihn erwartet (ISO9241-151 2008, 9.4.5ff). Konkret bedeutet dies, dass Links zu *besonderen Dateitypen*, Links, die *neue*

Fenster öffnen und „*Transaktions-Links*“, also solche, die Aktionen in Online-Anwendungen ausführen, gekennzeichnet werden.

Die ISO 9241-151 richtet sich primär an die Anbieter von Websites, sie fordert also keine Änderungen an den Browsern. Dabei könnten viele der aufgeführten Kriterien bereits automatisch durch den Webbrowser erfüllt werden. Dies hätte den Vorteil, dass sowohl die eindeutige Kennzeichnung von Link-Ankern als auch die Darstellung der zusätzlichen Link-Hinweise konsistent für alle Websites erfolgen würde, und Benutzer zuverlässig und auf vertraute Weise auf diese Informationen zugreifen könnten (s. auch Kapitel 9.2). Auf welche Weise sich diese beiden Aspekte der Link-Schnittstelle realisieren lassen, wird in den folgenden Abschnitten 5.2 und 5.3 als *Link Presence* und *Link Destination* systematisch untersucht.

5.2 Link Presence: Kategorisierung der Methoden zur Kennzeichnung von Link-Ankern

Dieses Teilkapitel widmet sich dem ersten Element der visuellen Hypertext-Rhetorik (s. Kapitel 5.1.1), der „*Link Presence*“. Es werden unterschiedliche Konzepte zur Kennzeichnung von Link-Ankern kategorisiert und die jeweiligen Vor- und Nachteile der Methoden analysiert. Die Kategorisierung legt den Schwerpunkt auf textliche Link-Anker, geht aber auch auf grafische und multi-mediale Anker ein (s. Abschnitt 5.2.9 und 5.2.10). Die Ergebnisse basieren sowohl auf Untersuchungen anderer Hypertext-Systeme (s. Anhang B) und Standards (s. Anhang C) als auch auf Forschungsergebnissen aus dem Bereich der Web-Usability.

5.2.1 Typografische Kennzeichnung

Unterschiedliche Schrifttypen und Schriftstile sind eine etablierte Methode zur Hervorhebung bedeutsamer Textabschnitte. Sie eignet sich ebenfalls zur Kennzeichnung von Link-Ankern im Text und geht bis auf das *Guide-System* (s. Anhang B.7) in den 1980er Jahren zurück, das fette und kursive Schrift für Link-Marker einsetzte (Brown 1987).

Ein Nachteil dieser Technik liegt darin, dass die verwendeten typografischen Stile „überladen“ werden, sofern man sie ebenfalls zur *inhaltlichen* Hervorhebung einsetzt. Evenson et al. propagieren daher die Entwicklung einer alternativen *visuellen Sprache* zur Gestaltung von Link-Markern, die neuartige Schriftstile verwendet. Als Beispiel schlagen Sie schattierten Text zur typografischen Kennzeichnung von Link-Ankern vor (Evenson, Rheinfrank et al. 1989).

Die heute gebräuchliche Unterstreichung von Link-Ankern ist neben der typografischen Überladung des Schriftstiles auch aus anderen Gründen problematisch: Diese Hervorhebungsmethode rührt ursprünglich aus der Gewohnheit her, etwas per Hand zu kennzeichnen (Unterstreichen zur Annotation). Später wurden Unterstriche bei Schreibmaschinen zur Hervorhebung verwendet, da diese Geräte in der Regel nur eine Schriftart zur Verfügung stellten (Evenson, Rheinfrank et al. 1989). Unterstreichungen betonen den Text aber *übermäßig stark* und *reduzieren seine Lesbarkeit*, da der Unterstrich mit den *Unterlängen* einiger

Buchstaben wie „g“, „p“ und „y“ interferiert. Studien haben die Wichtigkeit dieser Unterlängen („*Descenders*“) für die effiziente Lesbarkeit von Texten belegt (Mills & Weldon 1987). Unterstreichungen sollten daher nur eingesetzt werden, wenn keine anderen Möglichkeiten zur Hervorhebung verfügbar sind.

5.2.2 Farbliche Hervorhebungsmethoden

HyperTies vermied das Problem der Überladung von Schriftstilen, indem es eine charakteristische *Textfarbe* für Link-Marker verwendete[124] (Shneiderman & Kearsley 1989). Text- und Hintergrundfarben sind seitdem von diversen Hypertext-Systemen zur Hervorhebung von Link-Ankern eingesetzt worden (s. Abb. 41). Ein Vorteil liegt darin, dass farbiger Text kaum Einfluss auf die Lesbarkeit hat, sofern Schrift- und Hintergrundfarbe aufeinander abgestimmt sind (ISO9241-8 1998), ein hoher Helligkeitskontrast gewährleistet wird und entweder Text *oder* Hintergrund farbig ist (Mills & Weldon 1987). Allerdings können Menschen mit Farbfehlsichtigkeit Probleme haben, wenn nur die Schriftfarbe für die Hervorhebung genutzt wird, weshalb in der (ISO9241-151 2008, 9.3.9) auch hiervon abgeraten wird. Zudem stellt die dunkelblaue Link-Farbe gegenwärtiger Webbrowser eine suboptimale Wahl dar, da das Auge aufgrund der chromatischen Aberration Farben am Rande des sichtbaren Spektrums schlecht fokussieren kann und somit das Lesen solcher Texte erschwert wird. Hinzu kommt, dass ältere Menschen aufgrund einer zunehmenden Vergilbung der Augenlinse weniger empfindlich für Blautöne sind, und sie dunkelblaue Schrift nur schwer von schwarzer unterscheiden können (Lythgoe 1979).

Eine Alternative besteht darin, die *Hintergrundfarbe* und *-helligkeit* zu ändern. Harmony, der Client des Systems Hyper-G, verwendete diese Methode zur Kennzeichnung von Link-Ankern (Andrews 1996) und war so auch in der Lage, überlappende Anker darzustellen (s. Abb. 39, Abb. 41, und Anhang B.14).

Ein Beispiel-Text, der Schriftfarben zur Hervorhebung von Textbereichen verwendet.
Ebenso lässt sich die Hintergrundfarbe ändern.
Auf diese Weise können sogar überlappende Bereiche gekennzeichnet werden.

Abb. 41: Text- und Hintergrundfarben zur Hervorhebung von Textbereichen.

Farbiger Texthintergrund hebt Link-Anker potenziell stärker hervor als Schriftfarben. Bei Benutzertests, die im Kontext dieser Arbeit durchgeführt wurden, empfanden viele Teilnehmer Hintergrundfarben für Link-Marker als störend, da die Hervorhebungen wie farbige „Kästen“ im Texthintergrund erschienen und sehr dominant wirkten (Obendorf & Weinreich 2003).

Grundsätzlich sind Textfarben bei durchdachter Farbwahl eine bewährte Methode für einfache Link-Marker, die die Lesbarkeit eines Textes nur geringfügig beeinflussen und mit der

[124] Genau betrachtet ist dies wiederum eine technisch naheliegende Lösung, da bei der IBM-PC-Version von HyperTies aufgrund der verwendeten Hardware gar nicht die Möglichkeit bestand, den Schrifttyp zu ändern. Die damaligen Geräte boten aber bereits 16 Schrift- und Hintergrundfarben an.

sich einige Link-Attribute unterscheiden lassen (s. Abschnitt 5.3.3). Dem stehen aber Beschränkungen bei der Darstellung von überlappenden und aneinandergrenzenden Link-Ankern gegenüber. Dies wird durch Hintergrundfarben möglich, die dafür Lesbarkeit und Erscheinungsbild der Dokumente stärker beeinflussen.

5.2.3 Link-Symbole und Icons

In mehreren früheren Hypertext-Systemen wurden spezielle Symbole als Link-Marker verwendet. Beispielsweise nutzte bereits *KMS* einen kleinen Kreis „•“, um den Textanfang eines Ankers zu kennzeichnen (s. Anhang B.4 und Abb. 173 und Akscyn, McCracken et al. 1988). Intermedia stellte Link-Anker mithilfe eines kleinen Pfeils in einem Kästchen „▭“ dar, der zwischen den Textzeilen eingefügt wurde (s. Anhang B.10 und Abb. 184). Diese Hypertext-Systeme konnten so zwar den Anfang eines Ankers anzeigen, nicht aber den Umfang des zum Link-Anker gehörenden Textes (Yankelovich, Haan et al. 1988).

Das *MUCH-Hypertext-System* und *Emacs Info* lösten das Problem, indem sie den Link-Anker mit Sonderzeichen wie *Klammern* und *Sternchen* umschlossen (Wang & Rada 1995). Auf diese Wiese sind sogar überlappende Link-Marker darstellbar, wenn unterschiedliche Link-Symbole paarweise verwendet werden. Allerdings wird diese Methode bei zahlreichen und längeren Link-Ankern unübersichtlich, wie es bereits das Web-Annotationssystem *CritSuite*[125] demonstrierte (s. Abb. 42).

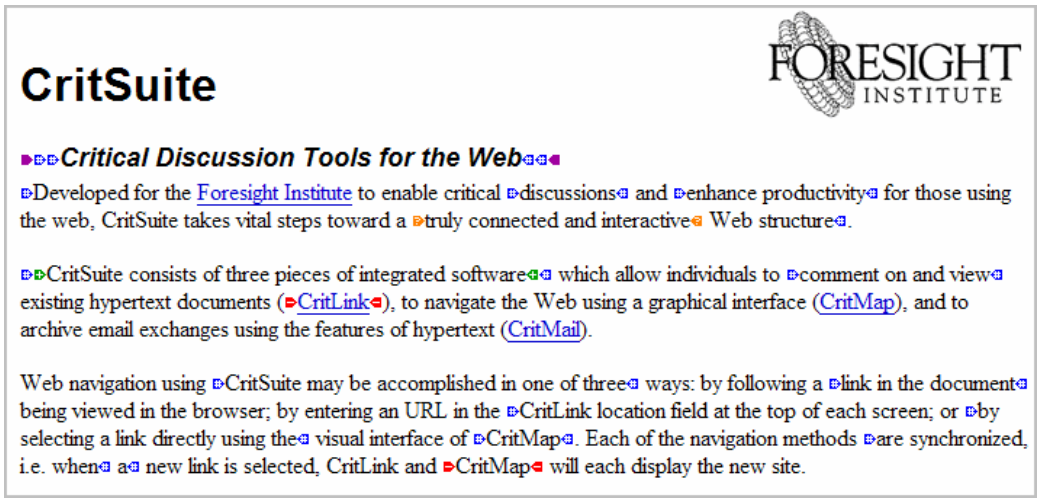


Abb. 42: Die *CritSuite* nutzte kleine bunte Pfeile, um Anfang und Ende von zusätzlich eingefügten Link-Ankern anzuzeigen. Zu sehen ist die ehemalige Homepage des Projektes unter http://www.crit.org/.

Ein Vorteil dieser Methode ist, dass sich Link-Typen durch verschiedene Icons symbolisieren lassen (Evenson, Rheinfrank et al. 1989; Noirhomme-Fraiture & Serpe 1998). Auf der anderen Seite können die eingefügten Link-Symbole den Lesefluss stören, selbst wenn sie relativ klein sind (Irler & Barbieri 1991). Allzu kleine Symbole sind schwer anklickbar.

---

[125] Das System *CritLink* wurde von Ka-Ping Yee an der University of California entwickelt. Der Dienst **war** unter der Adresse http://www.crit.org/ zu erreichen. Er wurde bereits Anfang 2004 eingestellt, der Screenshot stammt aus dem Jahr 2002. Die Open-Source-Software ist aber weiterhin unter der Adresse http://zesty.ca/crit/ erhältlich.

Sollen Links nachträglich zu einem Dokument hinzugefügt werden – beispielsweise durch ein offenes Hypertext-System –, so wird zusätzlicher Platz für die Link-Symbole benötigt. Entweder sie müssen dann in den Text eingeschoben werden oder sie verdecken Teile des Dokuments. Das Einfügen von Elementen ist aber selbst bei so flexiblen und plattformübergreifenden Dokumentformaten wie (X)HTML nur eingeschränkt möglich, da sich Einschübe häufig negativ auf das Erscheinungsbild und das Layout des ursprünglichen Dokuments auswirken (vergl. Abschnitte 5.3.2 und 5.5.1).

5.2.4 Hervorhebung durch Umrahmung

Eine ebenfalls in vielen Informationssystemen verwendete Kennzeichnungsweise für Hyperlinks sind grafische Kästchen, die den aktiven Anker-Bereich umrahmen. Diese Methode ist sowohl für Link-Anker im Text als auch für Grafiken und Animationen einsetzbar (s. Abschnitte 5.2.9 und 5.2.10).

Beispielsweise zeichneten *Hypergate, NoteCards* und das *Neptune Hypertext System* (s. Abb. 43) dünne Rahmen um die verknüpften Textbereiche. Gegenwärtig wird diese Visualisierungstechnik im *Adobe Acrobat Reader* verwendet.

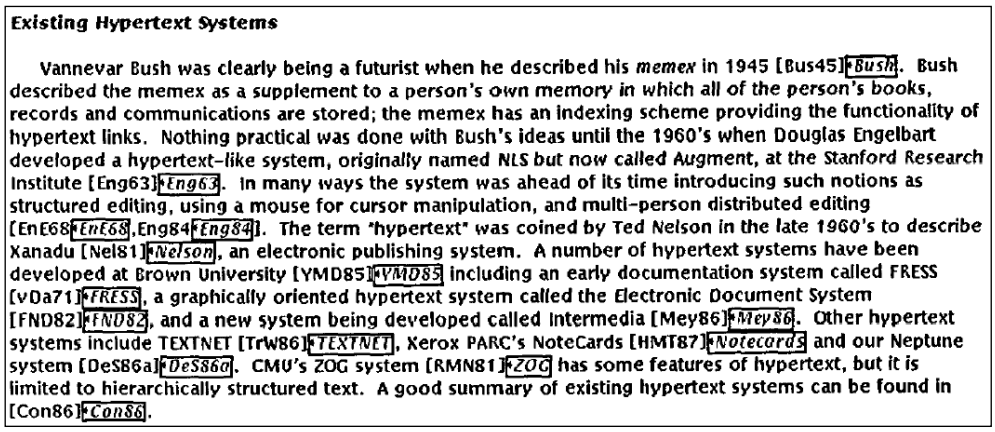


Abb. 43: Link-Marker im *Neptune Document Viewer* (aus Delisle & Schwartz 1987).

Varianten dieser Methode sind bei frühen Webbrowsern zu finden. *Cello* verwendete gestrichelte Rahmen (s. B.15 und Abb. 196) und *UdiWWW* unterschiedlich helle Linien für die Umrahmung, die wie hervorstehende Knöpfe im Text erschienen (s. Abb. 44; Berners-Lee & Fischetti 2000).

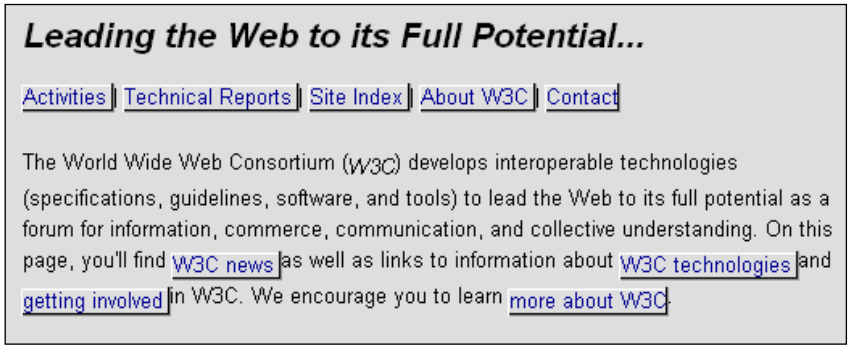


Abb. 44: Link-Marker im Webbrowser UdiWWW aus dem Jahr 1996.

Die Umrahmung weist für textliche Link-Anker ähnliche Schwächen bezüglich der Lesbarkeit wie die die Unterstreichung auf (s. Abschnitt 5.2.1), da die Linien mit dem Text interferieren. Dieser störende Effekt ist reduzierbar, indem ein schwacher Kontrast zwischen Rahmenfarbe und Hintergrund gewählt wird (Kahn, Peters et al. 1995). Durch den Einsatz verschiedener *Linienfarben* und *Linienstile* (*durchgängig*, *gestrichelt*, *gepunktet*, *gedoppelt*, siehe: Noirhomme-Fraiture & Serpe 1998) lassen sich zusätzlich unterschiedliche Link-Eigenschaften vermitteln (s. Abb. 47). Solche „dezenten" mehrfarbigen Rahmen wurden beispielsweise bereits in (Irler & Barbieri 1991) zur Annotation von Online-Texten favorisiert.

5.2.5 Overlays

Eine Methode zur grafischen Kennzeichnung von Bereichen, die für beliebige Bildschirmobjekte einsetzbar ist, sind sogenannte *Overlays* (Weinreich, Obendorf et al. 2001). Es handelt sich dabei um durchscheinende Flächen, die einen Bildschirmbereich farbig oder durch Helligkeitsunterschiede hervorheben. Die Konzepte „transluzenter" Interface-Elemente sind bei grafischen Benutzungsschnittstellen erst seit einigen Jahren verbreitet, obgleich Benutzbarkeitsstudien bereits vor über einem Jahrzehnt auf die Potenziale solcher Methoden hingewiesen haben (Harrison, Kurtenbach et al. 1995; Harrison & Vicente 1996; Cox, Chugh et al. 1998). Im Hypertext-Bereich wurden solche Techniken bisher fast gar nicht eingesetzt.

Das Erscheinungsbild farbiger Overlays zur Markierung von Text-Ankern weist Parallelen zu farbigem Texthintergrund auf mit ähnlichen Stärken und Schwächen (s. Abschnitt 5.2.2). Overlays eignen sich darüber hinaus gut zur Hervorhebung größerer Textbereiche, indem die *nicht* zum Anker gehörigen Regionen leicht ausgegraut werden. Beispielsweise verwendete der Postscript-Viewer von *Hyper-G* diese Technik zur Darstellung von Zielankern (s. Anhang B.14). In (Weinreich, Obendorf et al. 2001) wurde eine Erweiterung dieser Methode vorgeschlagen, um mittels zweier Helligkeitsstufen sowohl den *Zielanker* als auch den *Zielkontext* eines Links anzuzeigen (s. Abb. 45).

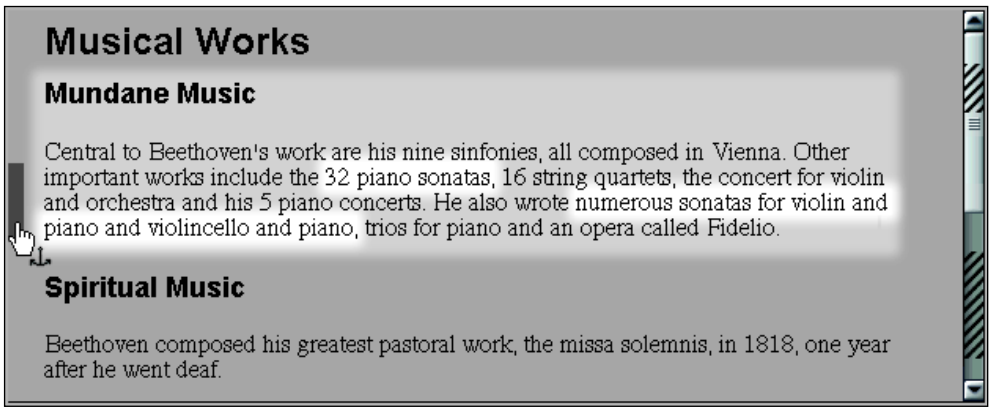


Abb. 45: Laterale Link-Marker, Schattierungen für den Link-Kontext und Markierungen im Scrollbalken für im Link verteilte Link-Anker (aus Weinreich, Obendorf et al. 2001).

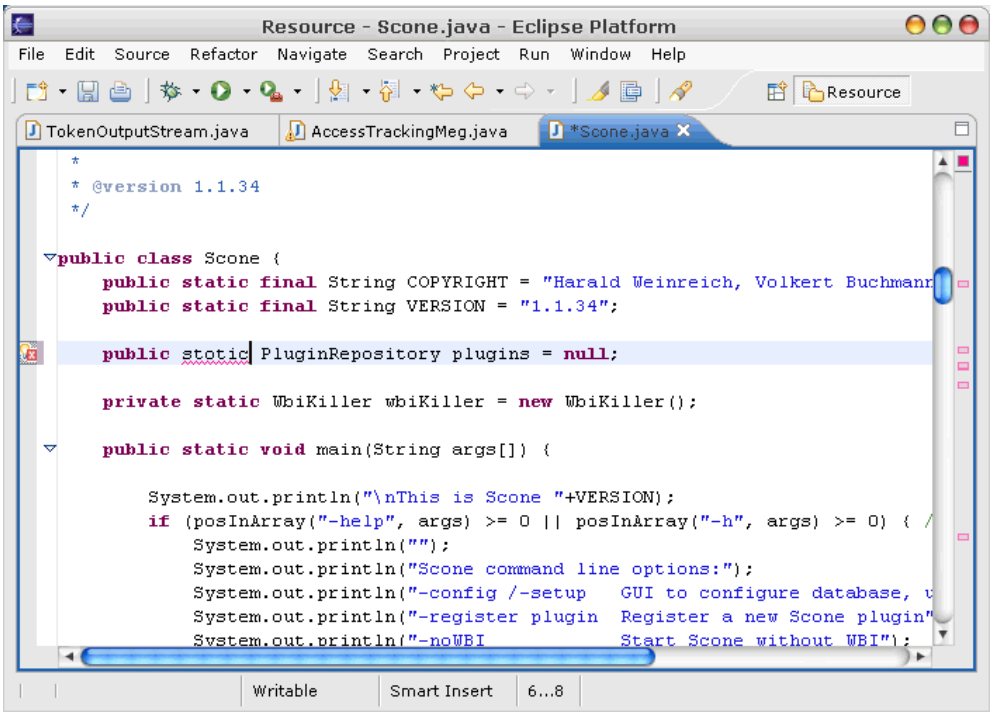


Abb. 46: Bei der Entwicklungsplattform *Eclipse* werden rechts neben dem Scrollbalken Fehler im Quellcode als rote Kästchen hervorgehoben.

5.2.6 Laterale Link-Marker

Eine für umfangreichere Link-Anker prädestinierte Methode sind *laterale Markierungen*. Das können zum einen Linien sein, die seitlich neben dem Text erscheinen und so den zum Anker gehörigen Textabschnitt kennzeichnen (s. Abb. 45 links) oder auch Markierungen im (s. Abb. 45 rechts) oder neben dem Scrollbalken (s. Abb. 46 rechts) des Fensters.

Erstmals wurde dieses Konzept zur Kennzeichnung von Textabschnitten von Donald Byrd für die Anzeige von Suchergebnissen vorgeschlagen, wobei alle Treffer innerhalb eines Dokuments gleichzeitig als Linien im Scrollbalken erschienen, damit der Benutzer einen Gesamteindruck über alle Fundstellen im Dokument erhielt (Byrd 1999). Inzwischen hat diese Technik in diverse Software-Entwicklungsumgebungen Einzug gehalten. Beispielsweise nutzt *Eclipse*[126] einen Bereich beim Scrollbalken zur Hervorhebung von Fehlern im Quellcode (s. Abb. 46).

Im Hypertext-Bereich wurde diese Methode bisher kaum eingesetzt. Eines der wenigen Beispiele ist das *Devise System*, das so Zielanker von eingehenden Links kennzeichnete (s. Abb. 47).

[126] Das Open-Source-Projekt *Eclipse* ist unter http://www.eclipse.org/ zu finden.

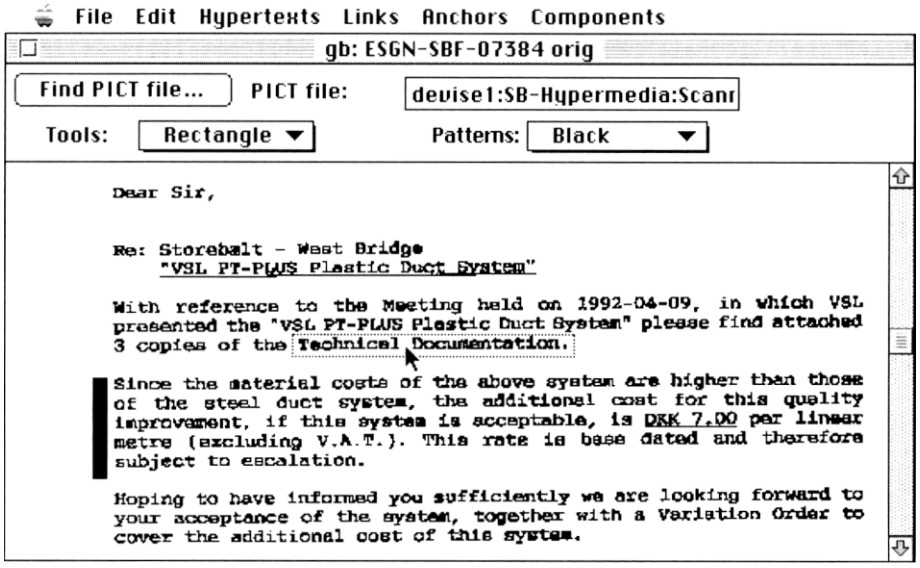


Abb. 47: Der seitliche schwarze Balken und der gepunktete Rahmen symbolisieren Zielanker von Links (aus Grønbæk & Trigg 1999).

Die Methode weist einige Stärken auf, die insbesondere bei offenen, verteilten Hypertext-Informationssystemen mit erweiterten Link-Möglichkeiten zum Tragen kommen. Neben *umfangreichen Link-Ankern*, die z. B. über mehrere Absätze gehen, eignet sich das Verfahren zum Hinweis auf *„verteilte" Link-Anker*. Dies sind Anker, die an mehreren Stellen im Dokument auftreten (beispielsweise jeweils zu einem Schlüsselwort), aber zu demselben Link gehören. Generische Links und Link-Sprachen wie XLink führen häufig zu solchen verteilten Ankern (s. Anhang C.3).

Allerdings ist diese Methode vergleichsweise *unpräzise*, da sich nicht einzelne Wörter oder kurze Textabschnitte eindeutig kennzeichnen lassen, sondern nur ganze Bildschirmzeilen. Sie ist daher eher als Ergänzung zu den bereits aufgeführten Konzepten zu sehen.

5.2.7 Unsichtbare Link-Anker

Microcosm/DLS war ein offenes Hypertext-System, das *generische Links* zu beliebigen Dokumenten hinzufügte (s. Anhang B.13). Dies führte dazu, dass ein Großteil der Wörter verknüpft war. Um eine „Überladung" mit Link-Markern zu verhindern, wurde auf eine Hervorhebung der Anker verzichtet. Die Auswahl generischer Links geschah, indem der Anwender ein Wort oder einen Textabschnitt mit der Maus markierte und dann über ein Icon im Titelbalken des Fensters den Link-Dienst aufrief (s. Abb. 190), der ihm eine Liste entsprechender Ziele anzeigte (Carr, DeRoure et al. 1995).

Unsichtbare Link-Anker bieten sich vornehmlich für Informationssysteme mit extrem vielen Link-Ankern bzw. generischen Links an, wie man sie beispielsweise bei der Verknüpfung mit einem Wörterbuch oder Lexikon erhält. Diese Methode ist hingegen bei Dokumenten mit relativ geringer Link-Dichte weniger empfehlenswert. Das zeigte sich bei Studien mit dem *Symbolics Document Examiner*, der Link-Marker nur darstellte, wenn man mit dem Maus-

zeiger über sie fuhr (Walker 1987): Es führte dazu, dass Benutzer regelrecht mit der Maus nach Ankern suchen mussten.

5.2.8 Links-on-Demand

Ein mit dem vorherigen nicht direkt vergleichbares Konzept zum Umgang mit Link-Markern wurde bereits 1968 von Doug Engelbart in seiner legendären Demonstration des Systems *Augment/NLS* präsentiert (s. Anhänge A.2 und B.2). Mithilfe des *Chord Keyset* und der *Maus* war es möglich, zusätzlich in die Dokumente eingebettete Link-Anker einzublenden. Später fand man diese Möglichkeit zum Aktivieren von Link-Markern in Systemen wie *HyperCard* und *Storyspace* (Bernstein, Bolter et al. 1991). Solche *Links-on-Demand* werden nur bei Bedarf sichtbar, wogegen das Dokument für den Rest der Zeit unverändert erscheint (Weinreich, Obendorf et al. 2001).

Links-on-Demand wurden 1987 bei den „Demo Sessions" der ersten Hypertext-Konferenz, bei der alle damals bedeutenden Hypertext-Systeme nebeneinander präsentiert wurden, von den anwesenden Experten als „beste Lösung" zum Umgang mit Link-Markern angesehen (Bernstein 1996).

Das „Verstecken" von Links hat viele Vorteile: Die Dokumente können in ihrer originären Form dargestellt werden, der Text büßt nichts von seiner Lesbarkeit ein, und das Aussehen der Seiten wird nicht von dem Erscheinungsbild der Link-Marker beeinflusst. Erst wenn der Leser weitere Informationen zu einem Thema wünscht, kann er die Link-Marker erscheinen lassen. *Links-on-Demand* bieten sich daher besonders für Seiten mit vielen Link-Ankern an:

> "If there are few anchors, it may be preferable to display them at all times. This alleviates distracting mode switches or ‚hunt and peck' searches for hidden anchors. If anchor density is high, their distinctive appearance may overwhelm the ‚normal' text. A toggle might then be welcome" (Keep, McLaughlin et al. 1993: „Anchor Appearance").

Andererseits bergen *Links-on-Demand* das potenzielle Problem, dass der zusätzliche Moduswechsel auf Anwender störend wirken kann; es sollte daher darauf geachtet werden, dass der Auslöser für die Aktivierung der Link-Marker nahtlos in die Benutzungsschnittstelle des Systems integriert ist (Weinreich, Obendorf et al. 2001).

Bei einer im Kontext dieser Arbeit durchgeführten Studie zeigte sich, dass *Links-on-Demand* zu einem anderen Umgang der Benutzer mit Hypertext führen können. Die Teilnehmer der Studie lasen den Text von verschiedenen Nachrichten-Seiten ausführlicher und fanden gesuchte Informationen signifikant häufiger, wenn sie die im Text eingebetteten Anker erst durch das Drücken einer Taste aktivierten. Waren die Link-Marker immer sichtbar, so verleitete dies die Teilnehmer offensichtlich häufiger zur Auswahl eines Links, ohne den Text zuvor angemessen gelesen zu haben. Die Mehrheit der Teilnehmer gab an, *Links-on-Demand* für Link-Marker im Fließtext gegenüber der normalen Darstellung des Webbrowsers zu bevorzugen, obwohl sie diese Umgangsform mit Webseiten als ungewohnt empfanden. Die ausführlichen Ergebnisse dieser Studie sind in (Obendorf & Weinreich 2003) zu finden.

Isabelle De Ridder kam 2002 in einem weiteren Experiment zu einem ähnlichen Ergebnis: Die Teilnehmer sollten fremdsprachige Dokumente lesen und verstehen. Dabei waren schwierige Vokabeln mit einer Übersetzung in einem Popup-Fenster verknüpft. In einer Versuchsbedingung wurden diese Vokabeln blau und unterstrichen hervorgehoben, in einer zweiten waren sie nicht markiert. Es zeigte sich, dass die Teilnehmer signifikant häufiger auf die Vokabeln klickten, wenn die Links sichtbar waren (Ridder 2002).

5.2.9 Link-Marker für Grafiken

Bei der Konzeption von Link-Markern für Grafiken gelten ähnliche Anforderungen wie für Link-Marker im Fließtext: Die Abbildung soll durch die Hervorhebung möglichst wenig beeinflusst werden und *originalgetreu* erscheinen, gleichzeitig muss der Benutzer aber die Anker *schnell* und *eindeutig* erkennen können. Viele der Methoden zur Hervorhebung von Text-Ankern sind aber für Abbildungen nicht einsetzbar, da Schriftzeichen im Gegensatz zu Grafiken aus einzelnen definierten Elementen bestehen, die über diverse Formart- und Stil-Attribute verfügen und einfarbig sind.

Zur Gestaltung von Link-Markern in Grafiken gibt es in der Literatur nur wenige Hinweise, und die verfügbaren Konzepte wurden bisher nicht systematisch ausgewertet. Vor dem World Wide Web boten nur wenige Informationssysteme überhaupt Link-Anker in Grafiken an, sodass auch im praktischen Bereich vergleichsweise wenig zu finden ist. Eines der ersten Systeme, bei dem auch Grafiken mit Hyperlinks versehen werden konnten, war *Intermedia* (Yankelovich, Haan et al. 1988). Als Link-Marker wurde *oberhalb* der verknüpften Grafik dasselbe kleine Icon „⇨“ eingefügt, das auch bei textlichen Links zum Einsatz kam (s. Abb. 184). Solche Symbole wären ebenfalls geeignet, um als Link-Marker *innerhalb* von Grafiken zu dienen, sofern sie an die Position des Ankers platziert werden; allerdings können die Link-Symbole dabei relevante Elemente des Bildes verdecken.

Bei *HyperCard* bestanden die einzelnen Karten zumeist aus Grafiken mit Text. Das System blendete *Rahmen* für alle Link-Anker auf der aktuellen Karte o*n-Demand* ein, sobald der Benutzer eine bestimmte Taste drückte (Nichols 1987). Der *Harmony Image Viewer* von *Hyper-G* umrahmte Link-Anker in Grafiken mit Polygonen (s. Anhang B.14 und Abb. 191).

Der Webbrowser *Mosaic* unterstützte bereits in der ersten Version eingebettete Grafiken, die gleichzeitig als Link-Anker dienen konnten (s. Anhang B.15). Solche verknüpften Grafiken wurden durch einen zusätzlichen blauen Rahmen *um* die Abbildung herum gekennzeichnet (NCSA 1996). Bis heute verwenden viele Browser wie der *Microsoft Internet Explorer* und *Mozilla Firefox*[127] diese Methode. Da der blaue Rahmen aber zumeist weder mit dem Design der Seite, noch mit der Grafik harmoniert, wird er von nahezu allen Web-Autoren deaktiviert. Stattdessen greifen Web-Designer auf andere, nicht-standardisierte grafische Methoden zurück, um Link-Anker kenntlich zu machen: Beispiele hierfür sind *dreidimensional*

[127] Ein Gegenbeispiel ist der Browser Opera, der auf diese Art der Hervorhebung verzichtet.

anmutende Schaltflächen, *farbliche Hervorhebungen* und „*Rollover-Effekte*“, bei denen sich die Grafik ändert, wenn man mit dem Mauszeiger über sie fährt. Diese Methoden setzen jedoch voraus, dass die Grafiken speziell für den Einsatzzweck als Link-Marker erstellt wurden. Das Hinzufügen von Verknüpfungen zu beliebigen Abbildungen (z. B. durch andere Autoren) ist so nicht realisierbar. Bisher bleibt dem Web-Nutzer daher als einziger *standardisierter* Hinweis auf grafische Link-Anker, dass sich der *Mauszeiger* zu einem Hand-Icon wandelt, wenn er sich über einem Anker befindet.

Eine weitere Methode, die keine explizite Gestaltung der Grafiken voraussetzt und auch für Anker innerhalb von Bildern einsetzbar ist, sind die bereits erwähnten *Overlays* (s. Abschnitt 5.2.5). Mit ihnen können aktive Bereiche farbig oder durch Helligkeitsunterschiede gekennzeichnet werden. Diese Art der Kennzeichnung stellt indes die Herausforderung, dass eine schwache Hervorhebung durch den Overlay bei detailreichen Bildern unter Umständen nicht auffällt, wogegen eine zu starke Hervorhebung die Grafik verfremdet.

5.2.10 Multimediale Link-Marker

Die Literatur zur Benutzungsschnittstelle von Link-Markern für Multimedia-Dateien ist ebenfalls rar. Viele Herausforderungen entsprechen denen bei der Kennzeichnung von Link-Ankern in Grafiken (s. vorheriger Abschnitt); zusätzliche Herausforderungen ergeben sich durch die Zeitkomponente: Beispielsweise können sich verknüpfte Objekte während der Wiedergabe bewegen, sodass die Link-Marker der Bewegung folgen müssen. Zwei Standards, die sich *technisch* mit der Beschreibung zeitlicher und räumlicher Aspekte von Link-Ankern auseinandersetzen, sind *HyTime* (s. Anhang C.2) und *SMIL* (Bulterman, Jansen et al. 2008). Allerdings bieten sie keine Information zur möglichen Realisierung der Benutzungsschnittstelle der Links.

Für Link-Marker in Animationen stehen ähnliche Verfahren zur Verfügung wie für Grafiken. Der *Harmony Film Player* von Hyper-G verwendete *umrahmende Polygone*, um Link-Anker in Filmen hervorzuheben (Andrews 1996). Alternativ lassen sich gefüllte Polygone als *Overlays* verwenden, um den Bereich des Link-Ankers zu kennzeichnen. *Link-Symbole* sind häufig weniger geeignet, sofern sich die Anker in der Animation bewegen, da sie (als relativ kleine bewegliche Schnittstellenelemente) bei einer Eigenbewegung nur schwer mit der Maus auszuwählen sind. Das *Elastic-Charles-Projekt* des MIT Media Labs nutzte als Link-Marker eine miniaturisierte, animierte Darstellung des Films, auf den verwiesen wurde. Diese *Micons* (für „*Moving Icons*“) wurden in den aktuellen Film eingeblendet, verdeckten aber Bereiche in der aktuellen Wiedergabe (Nielsen 1993a: 7; Goldman-Segall, Jonesson et al. 1996).

Neue Herausforderungen stellen *Zielanker* in Multimedia-Objekten, sofern sich das Ziel des Links zeitlich innerhalb der Animation befindet. Hierfür werden Konzepte benötigt, um den zum Zielanker gehörigen Teil kenntlich zu machen. Zudem ist zu definieren, welcher Teil der Animation dargestellt wird: Beginnt man am Anfang mit der Wiedergabe, so muss der

Benutzer möglicherweise lange bis zum Erreichen des eigentlichen Link-Ziels warten; beginnt das Playback zum Zeitpunkt des Ankers, so fehlt der Anfang der Animation und damit der inhaltliche Kontext, sodass Missverständnisse wahrscheinlich sind.[128]

Weitere Herausforderungen stellen Link-Anker in Audio-Dokumenten: Da man einen Ton nicht „anklicken" kann, werden ergänzende Schnittstellenelemente als Link-Anker benötigt. Beispielsweise könnte ein Audio-Player einen zusätzlichen Ausgabebereich anbieten, der (zum geeigneten Zeitpunkt) Informationen zu möglichen Link-Zielen einblendet, die dann mit der Maus auswählbar sind (Nielsen 1993a: 7).

5.2.11 Zusammenfassung

Diese Übersicht und Klassifikation unterschiedlicher Konzepte zur Hervorhebung von Link-Ankern für Texte und andere Objekte hat die Potenziale und Grenzen der einzelnen Methoden gegenübergestellt. Das Web mit seinen meist blauen und unterstrichenen Link-Ankern zeigt, dass auch bei der Gestaltung von solchen – auf den ersten Blick vielleicht trivialen – Aspekten von Benutzungsschnittstellen systematisch vorgegangen werden muss, da ansonsten Konzepte mit offensichtlichen Designmängeln zum *De-facto-Standard* werden können. Sowohl bei der Lesbarkeit des Textes, als auch den Möglichkeiten Preview-Informationen mit anzubieten, weisen andere Ansätze zur Link-Hervorhebung Vorteile auf. In den letzten Jahren gab es in diesem Bereich der Schnittstellengestaltung dennoch nur wenige Fortschritte in Forschung und Praxis, während in anderen Bereichen grafische Schnittstellen signifikant weiterentwickelt wurden. So ist man bei der Darstellung von textlichen Link-Markern bei dem historischen Zufallsprodukt „blauer, unterstrichener Link" als Standard stehen geblieben, und für grafische und multimediale Marker existieren bisher gar keine Normen.

5.3 Link-Destination: Darstellungstechniken für erweiterte Informationen zum Link-Ziel

Benutzer des World Wide Web erhalten oft nur spärliche Informationen zu den angebotenen Links. Dies kann zahlreiche Benutzbarkeitsprobleme zur Folge haben (s. Abschnitt 3.2ff). Zwei grundlegende Konzepte zur Reduzierung dieser Defizite sind *typisierte Links* und *Link-Previews* (s. Abschnitt 4.1ff). Damit diese Techniken hilfreich sind, müssen die entsprechenden Zusatzinformationen für die Benutzer auf geeignete Weise zugänglich sein. Obwohl es selbstverständlich erscheinen mag, war dies nicht einmal bei so wegweisenden Systemen mit typisierten Links wie *TextNet* (s. Anhang B.6), *Intermedia* (s. Anhang B.10) und *Hyper-G* (s. Anhang B.14) gegeben: Sie verwendeten die Typen lediglich zum Verarbeiten und Filtern

[128] Dies lässt sich exemplarisch beim Video-Dienst YouTube beobachten. Hier werden „Deep Links" zu einer bestimmten Stelle in einem Video durch eine Zeitangabe im Fragment-Teil der URI ermöglicht, wie „`#t=1m13s`" für einen Start des Videos bei 1min 13s (siehe: http://youtubetime.com/). Es fehlt dabei aber oft der Kontext, der durch den vorhergehenden Teil des Filmes oder der Audiodatei gegeben wird. Ein Beispiel ist der folgende Beitrag der BBC: http://www.youtube.com/watch?v=OzLeVGUDkvQ#t=0m22s.

des Hypertextes oder zeigten sie nur auf Anfrage. Andere Informationssysteme wie *NoteCards* oder *Sepia* stellten die Typ-Informationen zwar dar, verwendeten aber jeweils individuelle Visualisierungsmethoden, deren Gebrauchstauglichkeit kaum oder gar nicht evaluiert wurde. Darüber hinaus existieren bis heute kein Standard und keine systematisch erarbeiteten Erkenntnisse dazu, welche Methode zur Anzeige von Typ-Informationen für Links in welchem Einsatzbereich am geeignetsten ist.

Hypertext-Experten fordern seit Längerem, dass jeder Link-Typ einen eindeutigen und standardisierten Darstellungsstil haben sollte (u. a. Evenson, Rheinfrank et al. 1989; Hardman 1998). Dieser Abschnitt gibt einen Überblick über die wichtigsten Konzepte zur Anzeige zusätzlicher Informationen zu Hyperlinks und geht auf ihre jeweiligen Potenziale und Grenzen ein. Dazu werden die Konzepte früherer Hypertext-Systeme und Studien im Umfeld des Webs berücksichtigt und in Bezug zueinander gesetzt.

5.3.1 Ergänzender Link-Text

Das wohl nächstliegende Verfahren, um Links mit zusätzlichen Informationen zu versehen, ist das Einfügen eines erläuternden Textes beim Link-Anker. Der Text muss dabei lediglich für den Leser als zusätzliche Link-Information identifizierbar sein. Die Kennzeichnung kann beispielsweise durch spezielle Sonderzeichen oder grafische Symbole geschehen.

Bereits das erste Hypertext-System *Augment/NLS* verwendete diese Methode, um dem Anwender über den Link und das Zielobjekt zu informieren. Als Link-Anker diente ein Textcode in spitzen Klammern, z. B. „<augment, 13d, :xym>“. Er enthielt einen Bezeichner für den Dokumentnamen, den Abschnitt des Link-Ziels und eine *ViewSpec* genannte Zeichenfolge, die angab, wie das Zielobjekt erscheinen sollte (s. Anhang B.2 und Abb. 171).

NoteCards nutzte eine für Anfänger deutlich eingängigere Methode, um zusätzliche Informationen zum Link anzuzeigen. Umrahmt von einem Kästchen erschien der Link-Titel in spitzen Klammern gefolgt vom Titel des Zielobjektes (s. Abb. 48 unten).

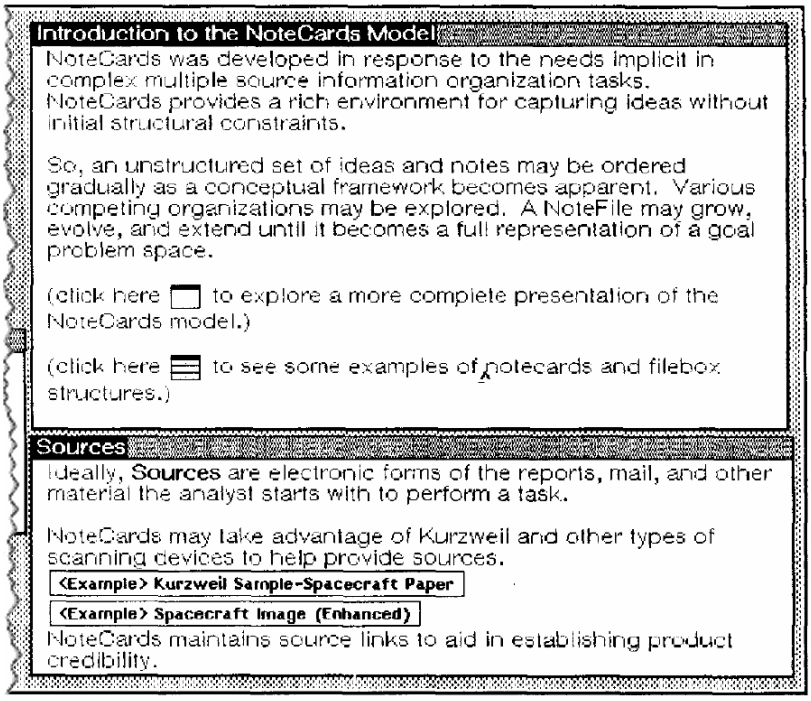


Abb. 48: Hyperlinks in *NoteCards* (nach Trigg, Suchman et al. 1986).

Da gegenwärtige grafische Webbrowser als Link-Marker lediglich einen Teil des Dokuments hervorheben und außer dem URI im Statusbereich keine weiteren Informationen zum Link zur Verfügung stellen, wird empfohlen, potenziell missverständliche Links durch zusätzliche Informationen hinter dem Anker-Text zu ergänzen (Spool, Scanlon et al. 1998; Nielsen 1999a). In Benutzbarkeitsstudien haben sich solche ergänzenden Texte, beispielsweise für die strukturellen Links auf Übersichtsseiten von Websites, als hilfreich erwiesen (s. Abb. 49 links): Die Teilnehmer konnten gesuchte Informationen im Durchschnitt am schnellsten finden, wenn Links mit „7 bis 12 Wörtern assoziiertem Text“[129] erläutert wurden (Spool, Scanlon et al. 1998). Solche ergänzenden Informationen sind zudem angebracht, wenn sich mehrere Links auf einer Seite sehr *ähneln*, die Verknüpfung zu *umfangreichen* Informationen führt oder wenn sie auf Objekte mit *ungewöhnlichen Eigenschaften* verweisen, z. B. einer großen Datei oder einem unerwarteten Dateityp (Nielsen 1999a; BITV2 2011; ISO9241-151 2008).

Ergänzende Link-Texte sind einsetzbar, wenn sie den Lesefluss und die Lesbarkeit des Dokuments nicht beeinträchtigen, z. B. für Auswahlmenüs und Listen von Links. Eine Variante dieser Methode wird bei den Ergebnislisten von Suchmaschinen eingesetzt (s. Abb. 49 rechts): Sie zeigen zu den Titeln der gefundenen Objekte – die als Link-Anker dienen – zusätzlich Ausschnitte aus dem Textkörper der Dokumente als „*Words in Context*“ an (Kaczmirek 2003).

[129] Hierunter verstehen Spool et al. Wörter, die aufgrund ihrer Position und der Seitengestaltung offensichtlich zum eigentlichen Anker-Text gehören (Spool, Scanlon et al. 1998).



Abb. 49: Zwei Beispiele für ergänzende Link-Texte in Webseiten.

Diese Methode ist weniger geeignet für Links, die im Fließtext eingebettet sind, da der assoziierte Text den Lesefluss stören und vom Inhalt ablenken kann. Zudem muss eindeutig vermittelt werden, dass sich der zusätzliche Text auf den Link bezieht und nicht Teil des angezeigten Dokuments ist. Darüber hinaus können mit dieser Methode *grafische* Link-Anker kaum mit zusätzlichen Informationen versehen werden, da eine Zuordnung zu der Grafik schwierig ist und häufig entsprechende Einfügungen bei den Abbildungen unmöglich sind.

5.3.2 Link-Hinweise durch Symbole und Icons

Einige frühere Hypertext-Systeme verwendeten spezielle Symbole oder Icons als Link-Marker (s. Abschnitt 5.2.3). Verschiedene Typen von Links können durch den Einsatz unterschiedlicher Symbole gekennzeichnet werden. Wegweisend für diese Technik war wiederum *NoteCards* (s. Anhang B.9), das bei bestimmten Link-Zielen den Typ der Zielkarte mittels eines entsprechenden Icons darstellte (s. Abb. 48 oben).

Dieses Konzept fand bereits in den 1980er Jahren zahlreiche Fürsprecher. Evenson et al. sprachen sich für die Entwicklung einer entsprechenden *Link-Sprache* mit standardisierten Icons oder Symbolen aus (Abb. 50), die im Text hinter den Link-Ankern erscheinen (Evenson, Rheinfrank et al. 1989). Grundlegende Link-Typen wie „Fußnote“, „Erweiterung“ oder „Definition“ sollten durch entsprechende Grafiken repräsentiert werden. Eine ähnliche Auffassung vertrat Lynda Hardman, die einen standardisierten Darstellungsstil für jede Art von Link als notwendig ansah (Hardman 1988). Als Beispiel führte sie an, dass eine Verknüpfung zum Anfang des Dokuments immer als Haus-Icon und ein sequenzieller Link zur nächsten Seite durch einen Pfeil nach rechts repräsentiert werden könnte.

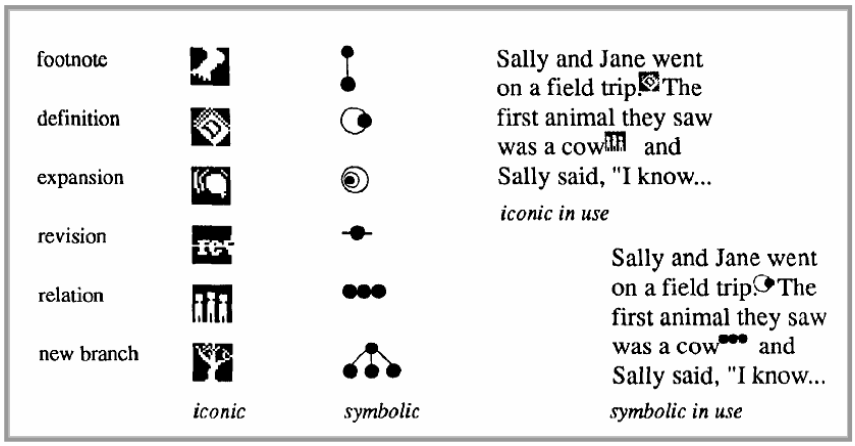


Abb. 50: Link-Icons und Link-Symbole nach Evenson (aus Evenson, Rheinfrank et al. 1989: 90).

Für das World Wide Web wurde dieses Konzept fast 10 Jahre später von einigen Experten wieder aufgegriffen (Levine 1996; Berners-Lee 1998a). Bruce Tognazzini schlägt beispielsweise zwei entsprechende Icons vor: Links zu einer anderen Position im selben Dokument sollten durch ein kenntlich gemacht werden, wogegen externe Links zu einer anderen Website mit einem zu versehen seien (Tognazzini 1998).

Link-Symbole weisen einige Vorteile auf: Sie benötigen wenig Platz, können bereits beim Überfliegen einer Seite wahrgenommen werden und die Informationen lassen sich direkt beim Link-Anker einfügen (Wollenweber 2004: 179).

Problematisch ist bei dieser Methode, dass die Icons den Lesefluss stören und die Verweise noch stärker betonen, als dies bereits durch die normale typografische Hervorhebung geschieht. In der Praxis ist dies auf der Website *Telepolis*[130] zu beobachten, die Site-interne und -externe Links unterscheidet; trotz der dezenten Gestaltung treten die Icons dominant im Text hervor.[131] Jakob Nielsen spricht sich daher gegen diese Methode für das Web aus und favorisiert Tooltips (s. Abschnitt 5.3.7; Nielsen 1999a: 67ff).

Eine weitere Herausforderung ist das Fehlen einer einheitlichen Icon-Sprache für die unterschiedlichen Link-Eigenschaften.[132] Benutzer müssen sich bei jeder Site mit neuen Symbolen auseinandersetzen, sofern sie überhaupt Link-Icons anbietet. Wenn alternativ das Informationssystem die Symbole automatisch ergänzen würde, ließe sich das Problem der Inkonsistenz vermeiden. Auf der anderen Seite können zusätzlich eingefügte Elemente das Erscheinungsbild des Originaldokuments negativ beeinflussen. Dies zeigte sich beispielsweise

[130] Telepolis, das „Magazin der Netzkultur“, wird vom Heise-Verlag online herausgegeben. Es ist unter der Adresse http://www.telepolis.de/ zugreifbar.

[131] Die Auffälligkeit der Icons lässt sich leicht beim Überfliegen dieser gedruckten Seite nachvollziehen.

[132] Es gibt einzelne Bestrebungen, einen solchen Standard für das Web zu schaffen. So schlug Michael Herrick von Matterform Media bereits 1994 über 30 verschiedene Icons als sogenannte „QBullets“ vor, um beispielsweise einen E-Mail-Link auszuzeichnen oder auf das Öffnen eines neuen Fensters hinzuweisen. Siehe: http://www.matterform.com/qbullets/.

bei der Browser-Erweiterung *Traffic-Lights*, die schmale, farbige Symbole[133] bei allen Link-Ankern einfügte (s. Abb. 51; Campbell & Maglio 1999). Viele Webseiten verloren durch diese Ergänzungen ihr ursprüngliches Layout und wurden unübersichtlich.

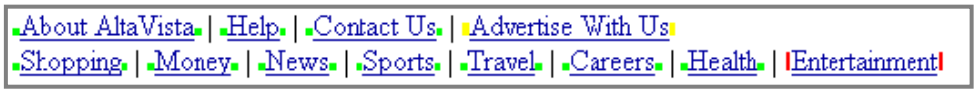


Abb. 51: Ein Beispiel der Darstellung von Link-Ankern mit der Browser-Erweiterung „*Traffic Lights*".

5.3.3 Farbig codierte Link-Marker

Farben eignen sich nicht nur zur Hervorhebung von Link-Ankern, bei Verwendung mehrerer Farben, Helligkeiten oder Muster lassen sich darüber hinaus unterschiedliche Link-Eigenschaften kennzeichnen. Bereits die erste veröffentlichte Version von Mosaic verwendete dieses Konzept und setzte zwei verschiedene Farben für Link-Marker ein: Links zu kürzlich besuchten Elementen wurden lila hervorgehoben, ansonsten waren sie blau (s. Abschnitt 5.1.3). Diese Methode wird bis heute in allen namhaften grafischen Browsern verwendet und als entscheidende Navigationshilfe angesehen (Spool, Scanlon et al. 1998; Siegel 1999; Nielsen 2004a). Bereits 1996 brandmarkte Jakob Nielsen eine von diesem Standard abweichende Link-Darstellung als einen der „*Top Ten Mistakes in Web Design*" (Nielsen 1996). Sechs Jahre später bestätigten Studien, dass die Unterdrückung dieser Methode eine Hauptursache für schlecht benutzbare Intranet-Sites ist (Nielsen 2002). Halverson et al. belegten in einer methodischen Untersuchung, dass zweifarbig unterschiedliche Link-Anker ein effizientes Unterscheidungsmerkmal bei der Suche nach *noch nicht* besuchten Dokumenten sind (Halverson & Hornof 2004).

Leider ist die Anzahl der schnell und eindeutig unterscheidbaren Farben begrenzt: So sieht (Byrd 1999) aufgrund seiner Studien *fünf Textfarben* als das Maximum dessen an, was Benutzer noch sicher differenzieren und zuordnen können (s. auch Kapitel 6.4.1). Dies schränkt die Anzahl der darstellbaren Link-Typen ein. Zudem ist eine intuitive Zuordnung von Farben zu bestimmten Link-Typen schwierig.

Aus Autorensicht können durch das System vorgegebene farbliche Kennzeichnungen von Link-Ankern hinderlich sein, da festgelegte Text- oder Hintergrundfarben nicht mit jedem gewünschten Design harmonieren und sie somit die Gestaltungsmöglichkeiten einschränken. Eine Anpassung der Link-Farben an das Design verbietet sich aber bei Verwendung mehrerer Link-Typen, da Website-abhängige Farbenzuordnungen das Verständnis der Methode bereits bei zwei Farben wesentlich erschweren (Nielsen 2004b; Nielsen 2004a).

---

[133] Abhängig von der voraussichtlichen Antwortgeschwindigkeit des Servers, der das Zielobjekt zur Verfügung stellte, fügte „Traffic Lights" kleine grüne, gelbe (langsam) oder rote (sehr langsam) Balken vor und hinter den Link-Markern ein.

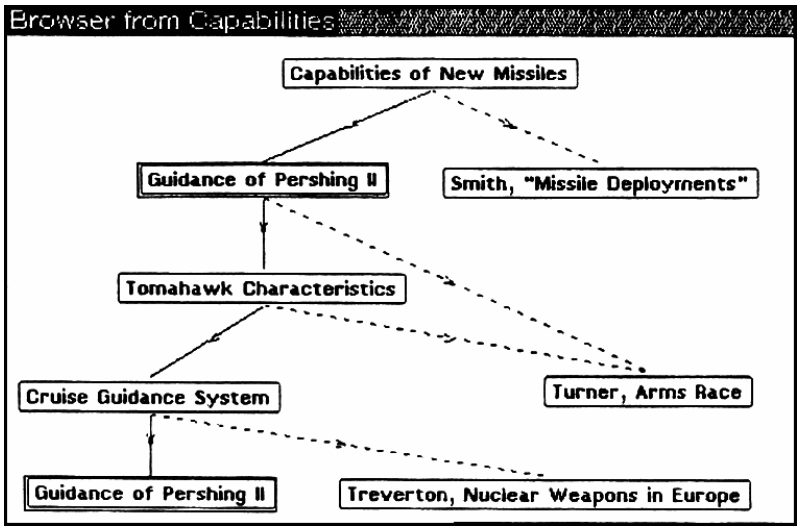


Abb. 52: Die Browser Card von *NoteCards* unterschied zwischen strukturellen Links und Querverweisen (aus Halasz 1988).

5.3.4 Link-Hinweise in Übersichtskarten

Eine Reihe früherer Hypertext-Systeme zeigte Link-Typen und Link-Previews in einer lokalen Karte (s. Abschnitt 3.1.4) an. Grafische Elemente oder Beschriftungen in unmittelbarer Nähe der Kanten (welche die Links symbolisierten) repräsentierten den Link-Typ.

Das erste System, das diese Technik verwendete, war *NoteCards* (s. Anhang B.9). Strukturelle und referenzielle Links unterschied das System durch zwei Linienstile (s. Abb. 52).

gIBIS nutzte Symbole auf der Karte für unterschiedliche Link- und Knoten-Typen (Conklin & Begeman 1987). Das Beispiel in Abb. 53 zeigt sieben als Quadrate repräsentierte Knoten. Der Knoten rechts unten wurde gerade ausgewählt, und sein Inhalt ist im rechten Drittel der Abbildung zu sehen. Die Pfeile stellen die Beziehungen in einem Netz von Argumenten und Standpunkten dar, und die kleinen Icons auf den Pfeilen stehen für den Typ der Relation[134] (vergleiche Abschnitt 2.2).

Die Systeme *SEPIA* (Thüring, Hannemann et al. 1995) und *MacWeb* (Nanard & Nanard 1993) verwendeten *Beschriftungen* bei den Pfeilen, um den Link-Typ zu kennzeichnen (Abb. 54). Dies sollte die Einlernzeit reduzieren und die Unterscheidbarkeit zahlreicher Link-Typen vereinfachen.

[134] Spätere Versionen von gIBIS verwendeten zusätzlich verschiedene Farben für unterschiedliche Knoten und Beziehungstypen. Dies erwies sich in Studien als ausgesprochen beliebt (Conklin & Begeman 1987).

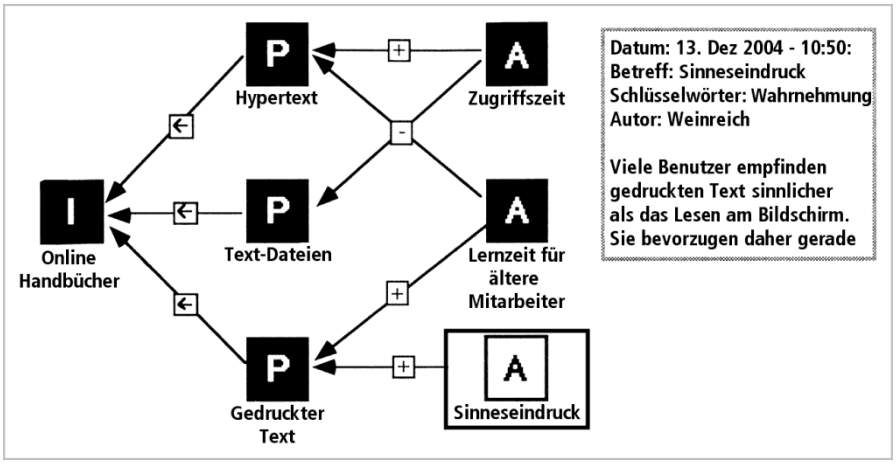


Abb. 53: Illustration eines *Issue Networks* in gIBIS[135] (nach Nielsen 1993a: 49).

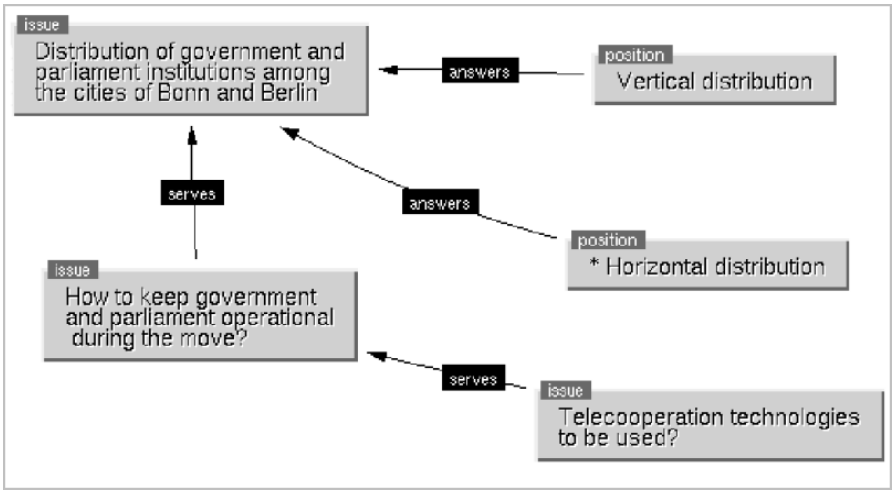


Abb. 54: Eine Karte in SEPIA, die als „Planungsraum" dient (nach Streitz, Haake et al. 1992).

Intermedia bot eine weitere Variante des Konzepts an: Anstatt die Link-Typen in einer lokalen Karte anzuzeigen, gab der grafische *Web View* eine Vorschau auf alle Zieldokumente der aktuell angezeigten Seite (s. Abschnitt 4.1.1). Als Link-Preview wurden für jedes Link-Ziel der *Titel* als Text sowie der *Objekttyp* als Icon dargestellt (s. Abb. 24; Catlin, Bush et al. 1989).

Die Vorteile der Übersichtskarte zur Darstellung zusätzlicher Link-Informationen ist die gleichzeitige Verfügbarkeit von Navigations- und Orientierungshilfen. Benutzer bekommen einen Überblick darüber, woher sie kommen, wo sie gerade sind und wohin sie navigieren können.

Allerdings hat die Anzeige von Link-Informationen in einer Übersichtskarte den Nachteil, dass Benutzer zugleich auf die Karte *und* das aktuelle Dokument mit den Link-Ankern achten müssen, um eine Beziehung zwischen den beiden Bereichen herzustellen. Zudem belegt die Karte zusätzlichen Platz auf dem Bildschirm, der nicht immer ausreichend zur Verfügung steht. Ein weiteres Problem kann sich bei der Verwendung mehrerer Fenster oder

[135] Es geht in dem Beispiel um die Erstellung eines Handbuches: **P** kennzeichnet die verschiedenen Standpunkte, **A** repräsentiert die Argumente, wobei das rechte untere aktuell ausgewählt ist. Die Icons bei den Pfeilen geben die Bedeutung des Links an: Stützt ein Argument einen Standpunkt oder spricht es dagegen.

Tabs ergeben, wenn die Zuordnung der Dokumente zu den Objekten auf der Karte nicht eindeutig gelöst ist. Für das World Wide Web sind momentan darüber hinaus Karten aufgrund der fehlenden strukturellen Informationen nur mit hohem Aufwand zu erstellen (s. Abschnitt 3.1.4).

5.3.5 Der reservierte Anzeigebereich

Der *reservierte Anzeigebereich* bezeichnet eine festgelegte Region der Benutzungsschnittstelle, die speziell für die Ausgabe zusätzlicher Link-Informationen vorgesehen ist. Dies können beispielsweise ein Teil des Fensterbereichs der Applikation, ein Areal auf dem Desktop oder ein zusätzliches Fenster sein.

HyperTies setzte als eines der ersten Hypertext-Systeme diese Methode ein: Im unteren Bereich des zeichenorientierten Bildschirms waren zwei Zeilen für eine Zusammenfassung des Zielobjektes reserviert (s. Anhang B.8, Abb. 180 und Abb. 181). Sobald der Anwender den Cursor zu einem Link-Anker bewegte, wurden diese Informationen angezeigt, und er erfuhr mehr über das Link-Ziel (Marchionini & Shneiderman 1988; Shneiderman & Kearsley 1989).

Bei Webbrowsern wird seit den ersten öffentlichen Beta-Versionen von Mosaic den URI des Link-Ziels unten im Statusbereich des Fensters eingeblendet, sobald sich der Mauszeiger über einem Link-Marker befindet (s. Abb. 55). Diese Methode wird bis heute in den meisten Webbrowsern eingesetzt. Der URI des Link-Ziels ist im Web dadurch die einzige *zusätzliche* Information, die der Benutzer grundsätzlich über das Zielobjekt erhält.

Soll ein Hypertext-System mehr Informationen zu dem angewählten Link darstellen als in eine oder zwei Zeilen passen, kann ebenfalls ein zusätzliches Frame oder Fenster eingesetzt werden. Beispielsweise bot eine Hypertext-Version der Ausgabe 7/95 des Magazins „Communications of the ACM“ zusätzliche Meta-Information zum ausgewählten Link in einem Frame der Seite an (Ramanathan, Bieber et al. 1997).

Der reservierte Bereich bietet den Vorteil, dass der Benutzer die zusätzlichen Informationen immer an derselben vertrauten Stelle findet. Er kann sie beachten, wenn er sie benötigt, ansonsten kann er sie ignorieren. Sie stören nicht die Lesbarkeit des Dokuments, und es lassen sich auch detailliertere Daten zum Link anbieten.

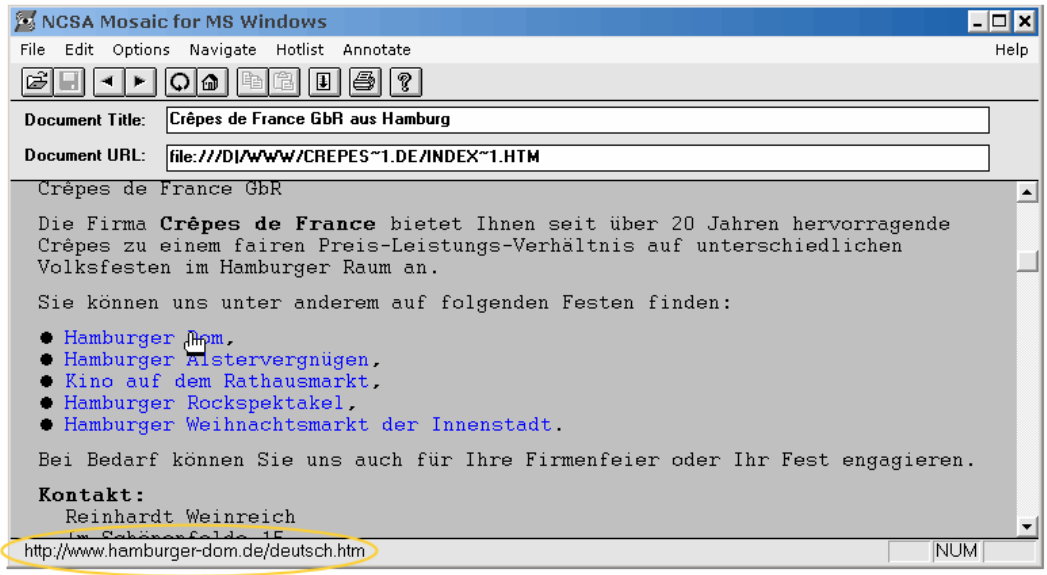


Abb. 55: Der Browser Mosaic für Windows zeigte den URI des Links im Statusbereich an. Die Abbildung zeigt die erste öffentliche Beta-Version 0.6 vom September 1993.

Diesen Vorzügen stehen einige Defizite gegenüber: So werden die zusätzlichen Informationen erst angezeigt, wenn der Benutzer einen Link anwählt, er also z. B. den Mauszeiger über den Link-Marker bewegt. Zudem erscheint die Information unter Umständen relativ weit vom Fokus der Aufmerksamkeit des Benutzers (dem Link-Anker) entfernt. Er muss dann größere „visuelle Sprünge" (*„visual saccades"*) vollziehen, um den Link-Text mit den ergänzenden Informationen in Beziehung zu setzen (Zellweger, Regli et al. 2000). Dies ist beim Lesen des Dokuments hinderlich. Zudem besteht das Risiko, dass der Anwender die Zusatzinformationen ungewollt übersieht (ibid.).

5.3.6 Der adaptive Mauszeiger

Eine weitere Möglichkeit zur Anzeige zusätzlicher Link-Informationen ist der *adaptive Mauszeiger*. Dieses Konzept ist bei grafischen Benutzungsschnittstellen recht verbreitet: Der Mauszeiger symbolisiert beispielsweise den gegenwärtigen Systemstatus und teilt dem Benutzer mit, ob der Computer gerade arbeitet ⌛, eine Anwendung auf eine Benutzereingabe wartet I oder ein Objekt ausgewählt werden kann ☝.

Das erste Hypertext-System, das den Mauszeiger zur Unterscheidung von Link-Typen einsetzte, war *Guide* (Nielsen 1995b: 57). Der Mauszeiger stand dabei für Art des Links und die Link-Aktion (s. Abb. 56 und Anhang B.7).

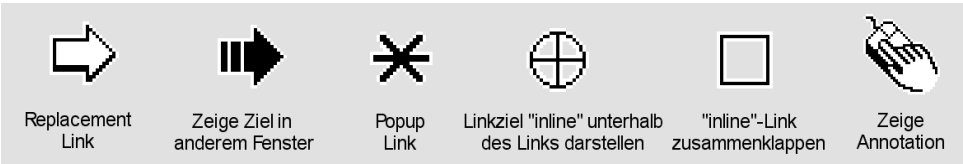


Abb. 56: Die wichtigsten Mauszeiger des Guide-Systems (vergl. Nielsen 1993a: 57).

In Webbrowsern wird diese Methode bisher nicht angewendet, dabei wurde sie bereits 1998 in dem Web-Forschungsprojekt *HyperNavi* vorgeschlagen (Noirhomme-Fraiture & Serpe 1998), in dem der Mauszeiger den Medientyp des Link-Zieles anzeigt. Ebenfalls findet man diese Methode in einem US-Patent von März 2002 (Burg & Schoeffler 2002), das spezielle Mauszeiger für unterschiedliche *Protokolltypen* in der Link-URI definiert: Es wurden unter anderem Symbole für „E-Mail-Links", „Video-Dienste" und „Finanz-Links" geschützt.[136]

Der Mauszeiger weist den Vorteil auf, dass die Information direkt beim Fokus der Aufmerksamkeit des Benutzers erscheint, sofern er gerade mit der Maus eine Aktion ausführt. So kann er beispielsweise immer vor der Auswahl eines Links auf wichtige Link-Eigenschaften hingewiesen werden. Des Weiteren bleibt das dargestellte Dokument unverändert und der Mauszeiger eignet sich auch für Grafiken und Animationen. Dies prädestiniert diese Methode für offene, verteilte Hypertext-Informationssysteme und macht sie zu einer interessanten Option für die Erweiterung von Webbrowsern (Nielsen 1999a).

Das Konzept weist auch einige Einschränkungen auf. Die anzuzeigenden Informationen müssen mittels eines kleinen Symbols beim Mauszeiger *abstrahierbar* sein. Zudem wird der Hinweis immer erst sichtbar, *nachdem* der Benutzer die Maus über den Link-Anker bewegt hat. Sucht ein Anwender aber Links mit bestimmten Eigenschaften – beispielsweise einen Link mit einer E-Mail-Adresse – kann dies zu einer unerfreulichen und zeitaufwendigen Suche mit der Maus im Hypertext-Dokument führen.

5.3.7 Tooltips

Ein *Tooltip* – auch *Popup*,[137] *Mouseover* und *Quickinfo* genannt – ist ein kleines Fenster ohne Kontrollelemente, das beim Mauszeiger erscheint, wenn er für kurze Zeit still über ein aktives Objekt gehalten wird. Dieses Konzept ist Desktopoberflächen seit vielen Jahren für kurze Zusatzinformationen gebräuchlich und wird beispielsweise verwendet, um die Bedeutung von Icons zu erklären (Abb. 57).

Die Idee, diese Technik auch für das Web einzusetzen, ist relativ alt: Bereits 1995 schlug die *Internet Engineering Task Force (IETF)*[138] vor, ergänzende Informationen zu Web-Dokumenten – beispielsweise Fußnoten – mithilfe von Tooltips darzustellen. Sie sollten auch verwendet werden, um ungewöhnliche Link-Typen in einem „*richer style*" wiederzugeben (Maloney & Quin 1995).

---

[136] Die sehr allgemeinen sowie technisch unvollständigen und vielfach fehlerhaften Beschreibungen der verwendeten Techniken in dieser Patentschrift stellen gleichzeitig die Sinnhaftigkeit derartiger Software-Patente in Frage, zumal die konzeptionelle Idee nicht neu ist (siehe das Guide-System und HyperNavi).

[137] Ursprünglich hat der Autor dieser Arbeit den Begriff *Popups* für diese Technik bevorzugt (s. Weinreich & Lamersdorf 2000). Da *Popups* aber zunehmend mit sich automatisch öffnenden und oft störenden Werbefenstern im Web assoziiert werden, wird in dieser Dissertation der Begriff *Tooltips* favorisiert.

[138] Die IETF ist eine internationale Vereinigung, die sich mit der technischen Weiterentwicklung des Internets befasst und technischen Dokumente und Standards definiert. Hierzu gehören die Protokolle TCP, SMTP und HTTP.

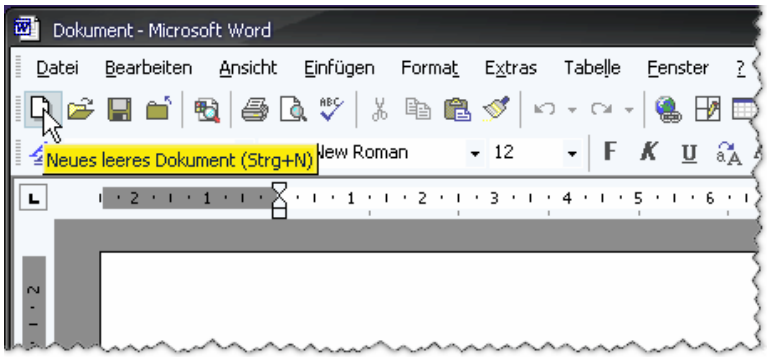


Abb. 57: Ein Tooltip beschreibt die Funktion eines Icons in Microsoft Word XP.

Der erste Browser, der Tooltips zur Anzeige zusätzlicher Informationen auf Webseiten einsetzte, war ab Mitte 1997 der Netscape Navigator 4. Hier fand der Benutzer alternative Texte zu Grafiken. Der Internet Explorer stellte einige Monate später in Version 4 zusätzlich den Inhalt des „`title`"-Attributs von Links in Tooltips dar (s. Abb. 58); dies wird bis heute von allen populären grafischen Browsern so gehandhabt.



Abb. 58: Ein Tooltip mit dem Inhalt des „title"-Attributs im Internet Explorer.

Jakob Nielsen riet zur Verwendung von Link-Titeln als *„one of the first enhancements to the Web that actually help people navigate (as opposed to simply making pages look more fancy)"* (Nielsen 1998a). Sie sollten seines Erachtens immer dann eingesetzt werden, wenn Verweise für den Benutzer missverständlich sein könnten. Zudem sah er es als sinnvoll an, wenn Browser bestimmte Links immer automatisch mit einem Tooltip kennzeichnen. Beispielsweise könnten für externe Links der Titel der Seite, der Name des Servers und der URI in einem Tooltip erscheinen (s Abb. 59). Zudem sollten auf diese Weise grundsätzlich Warnungen zu besonderen Benutzeranforderungen oder potenziellen Problemen erfolgen, beispielsweise als Hinweis „Benutzerregistrierung notwendig" (Nielsen 1999a: 60ff).

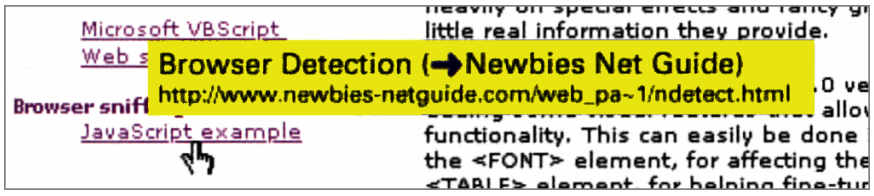


Abb. 59: Eine Illustration zur Kennzeichnung externer Links (aus Nielsen 1999a: 69).

Tooltips weisen eine zahlreiche Vorteile auf: Sie sind für die meisten Anwender ein vertrautes Konzept mit sehr flexiblen Darstellungsmöglichkeiten. In ihnen können sowohl kurze Texte als auch ausführliche mehrzeilige Angaben und Grafiken angezeigt werden. Sie lassen zudem das Dokument unverändert und erscheinen dennoch nahe dem Fokus der Auf-

merksamkeit. Überdies sind sie ebenfalls mit Link-Ankern in Grafiken und Animationen nutzbar.

Es gibt mehrere Variationen von Tooltips, bei denen sie nicht nur zur Anzeige von Informationen, sondern auch als zusätzlicher *Interaktionsbereich* dienen, beispielsweise für eine Auflistung von Auswahlmöglichkeiten. Koved und Shneiderman schlugen bereits vor über 20 Jahren vor, solche „Popup-Windows" mit zusätzlichen Optionen unterhalb von Link-Ankern im Text (s. „*Embedded Menues*" in Abschnitt 4.2.2 und Anhang B.8) einzusetzen, um dem Benutzer ein Auswahlmenü unter dem Link anzubieten (Koved & Shneiderman 1986: 67).

Microsoft integrierte eine ähnliche Technik in eine Beta-Version des Internet Explorers 6: Tooltip-ähnliche „*Smart Tags*" erschienen auf beliebigen Webseiten bei speziellen Schlüsselwörtern. Ein Klick auf den Smart Tag öffnete ein Menü mit zusätzlichen, von Microsoft definierten Zielen,[139] z. B. zu ausgewählten Geschäftspartnern (s. Abb. 60).



Abb. 60: Ein *Smart Tag* in der Beta-Version des *Microsoft Internet Explorer 6*. Beim Anklicken öffnet sich eine Liste von Links.

Trotz ihrer umfangreichen Potenziale haben Tooltips auch einige Nachteile. Sie können wichtige Informationen verdecken, insbesondere wenn sie unbeabsichtigt erscheinen. Zur Reduzierung des Problems erscheinen Tooltips daher in der Regel erst, nachdem der Anwender die Maus über einem aktiven Objekt für eine kurze Zeit stillgehalten hat. Diese Verzögerung führt allerdings zu einer zusätzlichen Wartezeit. Problematisch kann auch sein, dass nicht immer zu erkennen ist, ob ein Tooltip mit zusätzlichen Informationen erscheinen wird oder nicht – so warten Web-Benutzer auf einen Tooltip mit dem Link-Titel häufig vergebens (s. Abschnitt 4.4.2).

5.3.8 Weitere Techniken

Es gibt eine Reihe weiterer Techniken zur Anzeige von zusätzlichen Link-Informationen, die bisher in grafischen Benutzungsoberflächen und Hypertext-Systemen keine nennenswerte Verbreitung gefunden haben und daher eher als experimentell einzustufen sind. Für bestimmte Einsatzgebiete oder spezielle Benutzergruppen sind sie aber durchaus vielversprechend. Zwei Beispiele sollen dies illustrieren:

---

[139] Diese Art des kommerziellen Gebrauchs von *generischen Links* stieß auf große Ablehnung in der Öffentlichkeit, da man einen Missbrauch befürchtete. Microsoft entfernte dieses Feature daher in der Endversion des Browsers.

Bereits 1995 wurde auf der CHI-Konferenz in Palo Alto (Kalifornien) eine Erweiterung für Webbrowser vorgeschlagen, die *akustische Hinweise* einbezog, um Benutzern Daten zum Link-Ziel zu liefern (Albers & Bergman 1995). Die Autoren empfahlen unter anderem zerbrechendes Glas für fehlerhafte Links, das Geräusch eines „Video-Projektors" für Video-Dateien und das Ticken einer Uhr für die voraussichtliche Transferzeit. Die Geräusche sollten alle nacheinander abgespielt werden, wenn der Benutzer mit der rechten Maustaste auf den Link-Marker klickte. Zudem sollten Benutzer Feedback über ihre Aktionen erhalten und ein „Bop-Bop" nach der Auswahl eines Links erklingen.[140] Eine Evaluation des Prototyps erfolgte nicht.

Kritisch ist anzumerken, dass solche Töne die Konzentration des Anwenders stören können und die (zeitintensive) Wiedergabe die Interaktion verzögert. Andererseits sind akustische Hinweise unter speziellen Einsatzbedingungen durchaus sinnvoll, um die Zugänglichkeit und Barrierefreiheit (Accessibility) zu optimieren: Bei Geräten mit beschränkter Bildschirmgröße, für Benutzer mit Sehbehinderungen oder in Situationen, bei denen Sprachsteuerung eingesetzt wird und wenn der Benutzer den Bildschirm nicht beachten kann (z. B. im Automobil), sind solche Konzepte ernsthaft in Erwägung zu ziehen (Chisholm, Vanderheiden et al. 1999; s. auch Abschnitt 9.3.2)

Eine weitere innovative Benutzungsschnittstelle für Hyperlinks heißt *Fluid Links* (Zellweger, Chang et al. 1998). Bei dieser Technik wird *zwischen* den Textzeilen animiert ein Bereich eingefügt, um dort eine Randbemerkung (*Gloss*) mit Informationen zum Link einzublenden, sobald der Benutzer den Mauszeiger über einen Link-Anker bewegt. Dieses Konzept wurde auch für das Web umgesetzt (Zellweger, Bouvin et al. 2001), wobei es bei vielen Webseiten trotz diverser Anpassungen beim Einblenden der Randbemerkung Probleme mit dem Seitenlayout gab. Zudem kann die Bewegung des Textes als irritierend empfunden werden, insbesondere wenn der Einblendvorgang eines *Gloss* ungewollt ausgelöst wurde. Für interaktive und multimediale Anwendungen können solche „fließenden" Interaktionsformen dennoch eine gestalterisch angemessene Option darstellen.

5.3.9 Zusammenfassung

Die vorgestellten Konzepte zur Anzeige zusätzlicher Link-Informationen weisen charakteristische Vor- und Nachteile auf, die sich je nach Anwendungsfeld, Art und Umfang der darzustellenden Informationen unterschiedlich auswirken. Bei der Wahl der optimalen Methode müssen daher eine ganze Reihe von Anforderungen berücksichtigt und gegeneinander abgewogen werden:

- Ein wesentliches Kriterium ist die gewünschte ***Detailliertheit*** der angebotenen Informationen. Soll nur eine geringe *Anzahl* von Eigenschaften unterscheidbar sein, wie dies bei

[140] Diese partiell schon nahezu alberne Realisierung von Audio-Hinweisen wurde übrigens entsprechend von Albers und Bergman im April 2001 als US Patent 6,223,188 geschützt.

einigen streng typisierten Link-Schemata (s. Abschnitt 4.2.3) der Fall ist, und lassen sich diese gut *abstrahieren*, so sind Symbole, Icons und Farben eine geeignete Ausdrucksform. Eine freie Typisierung von Links (s. Abschnitt 4.2.2) oder die Anforderung, umfangreichere Informationen zum Link und Zielobjekt anzubieten, setzen Methoden voraus, die entsprechend detaillierte und textliche Ausgaben ermöglichen, wie dies beim reservierten Bereich, bei Tooltips oder bei Fluid Links gegeben ist.

- Ein weiterer Faktor ist der ***Darstellungszeitpunkt***. Es kann notwendig sein, dass die zusätzlichen Link-Informationen bereits beim Lesen der Dokumente sichtbar sind, damit Anwender sie nach Verknüpfungen eines bestimmten Typs absuchen können und sie sofort sehen, ob ein Link über bestimmte Eigenschaften verfügt oder nicht. In diesen Fällen sind farblich unterschiedliche Link-Marker, eingebettete Texte und Icons die Methoden der Wahl. Werden hingegen die Zusatzinformationen nur auf Anforderung benötigt, so lassen sich der Mauszeiger oder ein reservierter Bereich zur Anzeige verwenden. Tooltips sind in Betracht zu ziehen, wenn eine kleine Verzögerung bis zum Erscheinen der Link-Informationen tolerierbar ist.
- Als drittes Kriterium sollte dedacht werden, ob zur Anzeige der zusätzlichen Link-Informationen eine ***Änderung der Dokumentdarstellung*** möglich ist oder sich dies aus technischen, Layout- oder Copyrightgründen verbietet. Ist letzteres der Fall, so sind ebenfalls Verfahren einzusetzen, die erst auf Anforderung zusätzliche Informationen (über dem Dokument) einblenden. Sie sind ebenfalls mit anderen Medientypen wie *Grafiken* oder *Animationen* nutzbar und bieten den Vorteil, die *Lesbarkeit* des Textes nicht zu beeinflussen.
- Der vierte Faktor ist die geforderte *Nähe* zwischen Link-Anker und dem zusätzlichen Link-Hinweis und damit die unmittelbare Wahrnehmbarkeit. Handelt es sich um besonders wichtige Informationen wie Warnmeldungen, so sollte die Darstellung möglichst nah zum Link-Anker und damit beim Fokus der Aufmerksamkeit des Benutzers vor der Auswahl eines Links geschehen. Nützliche, aber nicht notwendige Hinweise können hingegen auch in einem anderen Bereich des Bildschirms erscheinen, wo sie – wenn benötigt – beachtet werden können, aber ansonsten kaum stören oder ablenken.
- Fünftens müssen das Paradigma der Benutzungsschnittstelle und die verfügbaren Eingabegeräte berücksichtigt werden: Bei *WIMP-Schnittstellen* mit Maussteuerung und Tastatureingabe bieten sich andere Möglichkeiten zum Einblenden von Zusatzinformationen an als bei *NUIs* (*Natural User Interfaces*) mit *Multi-Touch-Steuerung* (s. Kapitel 9.3.2).

Diese Überlegungen waren Grundlage für die Entwicklung der unterschiedlichen Konzepte und Prototypen des *HyperScout*-Projektes, die ab Kapitel 5.5 dargelegt werden.

5.4 Verwandte Forschungsprojekte zur Erweiterung der Link-Schnittstelle im Web

Die Klassifikationen zur Darstellung von Link-Ankern und erweiterten Link-Informationen in den vorherigen Abschnitten 5.2 und 5.3 sowie die Untersuchungen der Link-Schnittstellen früherer Hypertext-Systeme (siehe Anhang B) haben bereits die Konzepte zahlreicher anderer Forschungsprojekte für die Gestaltung der Link-Benutzungsschnittstelle berücksichtigt. Es gibt einige Forschungsarbeiten, die sich explizit mit den Möglichkeiten zur Erweiterung der Link-Schnittstelle im Web auseinandersetzen und deren Ziel es ist, dem Anwender mehr Informationen zum Typ und Ziel des Links darzustellen. Die im Kontext dieser Arbeit relevantesten dieser Projekte werden im Folgenden vorgestellt.

Bereits 1997 kritisierte Ken Magel die Defizite von Hyperlinks im Web und die zur damaligen Zeit vorgenommenen Erweiterungen an der Sprache HTML. Die Entwickler der Browser *NCSA Mosaic* und *Netscape Navigator* führten damals sukzessive neue Elemente wie „`table`“, „`font`“ und „`frame`“ in die Sprache HTML ein (s. Anhang B.15), die sich aufgrund der Dominanz der Browser schnell zum De-facto-Standard im Web entwickelten und später auch in die Empfehlung des W3C übernommen wurden. Magel bemängelt, dass alle diese Erweiterungen die Benutzbarkeit des Webs außer Acht ließen (Magel 1997); um eine konsistente Benutzungsschnittstelle zu erhalten, sollten seiner Ansicht nach Layout-spezifische Elemente aus HTML entfernt und die Darstellung der Seite dem Browser überlassen werden. Im Prinzip plädiert er somit bereits für die Barrierefreiheit im Web. Stattdessen müssten, so argumentiert er, die Benutzbarkeitsprobleme des Webs bei der Weiterentwicklung von HTML im Fokus stehen. Dies ließe sich beispielsweise bewerkstelligen, indem Links mehr semantische Informationen anbieten, um so die Navigation für alle Benutzergruppen zu vereinfachen („*One way of doing so is by adding value to links*“, aus: Magel 1997). Als besonders bedeutsame und oft vermisste Informationen führt er die *Dateigröße* des Link-Ziels, dessen *letzte Aktualisierung*, den *Dateityp* und die *Anzahl der Links* im Zieldokument auf. Seiner Auffassung nach können zusätzliche Link-Attribute entweder auf Serverseite in die Dokumente eingebunden oder durch Prefetching von anderen Servern geholt werden (vergl. Kapitel 8.3.1). Zur Visualisierung der Link-Eigenschaften stellt er die Nutzung unterschiedlicher Textstile, verschiedenartiger Unterstreichungen (dick, dünn, dicker werdend), Schriftfarben und Icons im Text vor. Wenn für einen Link noch keine zusätzlichen Informationen vorliegen, sollte dies erkennbar sein, beispielsweise durch gestrichelte Unterstreichungen der Link-Anker (Magel 1997). Auf die konkrete Realisierung seiner Konzepte geht er nicht weiter ein, und eine Evaluation wurde nicht publiziert.

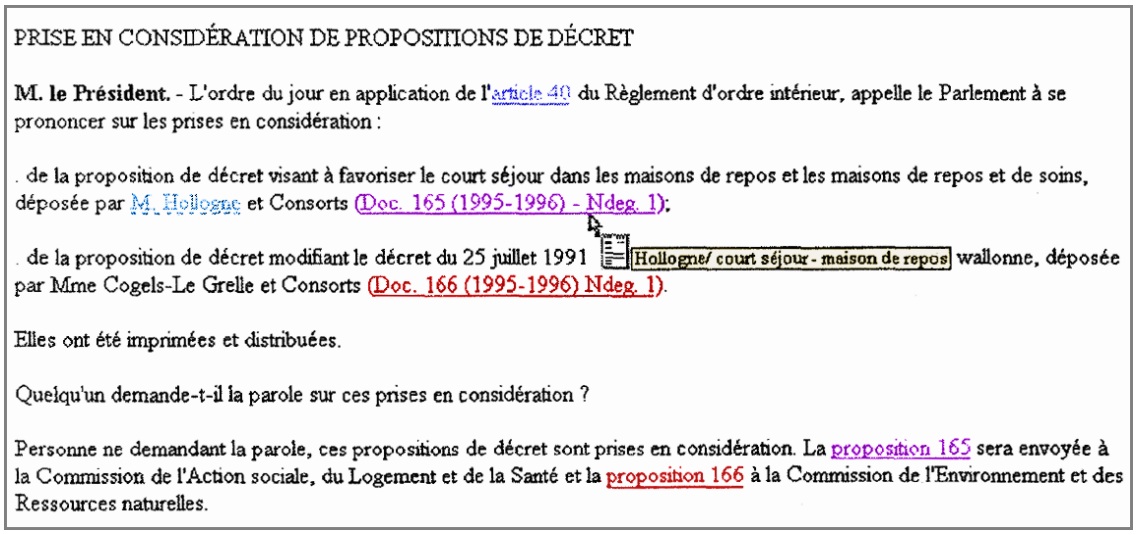


Abb. 61: Vorschläge für die visuelle Darstellung unterschiedlicher Link-Typen (nach Noirhomme-Fraiture & Serpe 1998).

Jahr später stellen (Noirhomme-Fraiture & Serpe 1998) unter dem Namen *HyperNavi* Konzepte zur Darstellung zusätzlicher Informationen zu Links im Web vor. Die Link-Informationen sollen dabei durch die Autoren definiert und vom Webserver bereitgestellt werden. Nach ihren Untersuchungen lassen sich Hypermedia-Links im Web in vier Dimensionen klassifizieren:

- Die erste Dimension „*Nature*“ dient zur Unterscheidung von strukturellen und semantischen Links (s. Abschnitt 4.5.1.1).
- „*Automatic Identification Possibility*“ befasst sich mit der Unterscheidung von manuell gegenüber automatisch generierten Links (s. Anhang B.13).
- Die „*Destination Characteristics*“ beschreiben die Eigenschaften des Link-Zieles. Sie sind in sieben weitere Achsen unterteilt (unter anderem Größe, Alter, Medientyp, Verfügbarkeit und Kosten für den Zugriff).
- „*Physical Characteristics*“ geht auf erweiterte Link-Eigenschaften ein, wie man sie bisher nur aus offenen Hypertext-Systemen (s. Abschnitt 2.2.2.2) kennt. Hierzu gehören einfache gegenüber multiplen Links, die Richtung des Links und die Sichtbarkeit des Link-Ankers (vergl. Kapitel 2.2.1).

Zur Darstellung der Link-Attribute listen die Autoren zahlreiche Methoden auf. Hierzu gehören Text-Stile und -Farben, Icons und Symbole als auch unterschiedliche Mauszeiger, Tooltips und ein Vorschaufenster. Eine prototypische Umsetzung und eine Evaluation wurden in dem Paper angekündigt, aber nicht publiziert. Abb. 61 zeigt einen Entwurf des Systems, bei dem unterschiedliche Unterstreichungen, Link-Farben, ein Icon und ein Tooltip verwendet werden.

An der RWTH Aachen wurde Ende der 1990er eine „Java Toolbox“ entwickelt, die die Navigation im Web vereinfachen und die Probleme der Orientierung reduzieren sollte (Kreutz, Conradi et al. 1998; Kreutz, Euler et al. 1999). Die Anbieter von Webseiten erhalten

durch die Toolbox die Möglichkeit, mithilfe zusätzlicher HTML-Elemente die Struktur einer Website zu definieren und allen Links über ein neues Attribut einen „Typ“ zuzuordnen. Die Struktur lässt sich als Karte der Website (s. Abschnitt 3.1.4) darstellen, in der man die Position des aktuellen Dokuments sieht, für jedes Dokument eine Vorschau erhalten und auch direkt eine Ressource auswählen kann. Die Link-Typen werden im Webbrowser als Icons hinter jedem Link symbolisiert (siehe Abb. 62). Wenn der Benutzer auf das Icon klickt, werden zusätzliche Informationen „on-the-fly“ geladen und in einem kleinen Popup-Fenster dargestellt. Man erhält eine Zusammenfassung des Link-Zieles (s. Abb. 62: „Abstract“) und Angaben über den Autor des Dokuments (s. Abb. 62: „Author“). Für spätere Versionen waren noch weitere Meta-Information geplant. Das System verwendet zur Erweiterung der Link-Schnittstelle Java Applets und es funktioniert daher mit beliebigen grafischen Webbrowsern. Die Autoren haben eine nicht näher spezifizierte Evaluation des Systems mit 60 Studenten durchgeführt, die als Ergebnis erbrachte, dass 80% der Teilnehmer die Erweiterungen wertschätzen würden („80% appreciated the new techniques“, siehe: Kreutz, Euler et al. 1999).

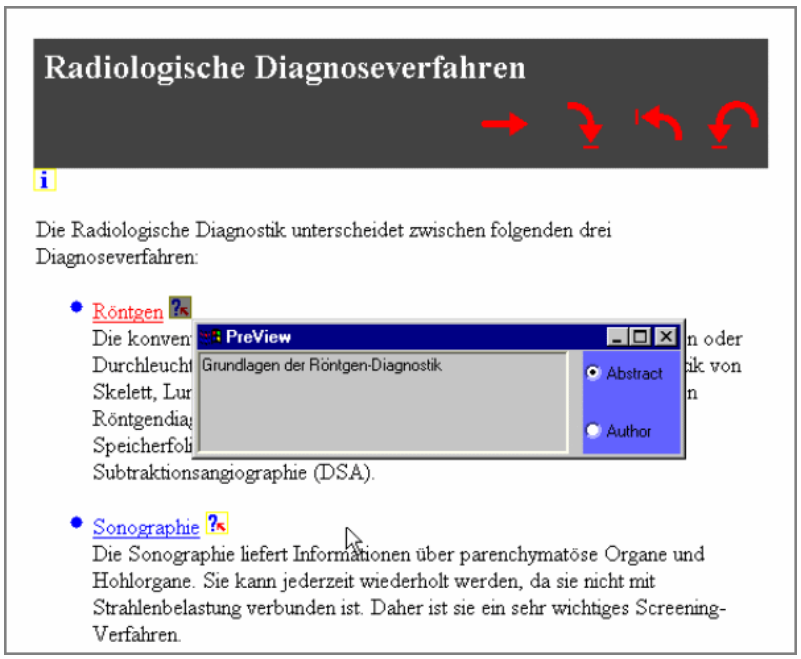


Abb. 62: Die Java-Toolbox zeigt mithilfe von Java Applets zusätzliche Informationen zum Link-Ziel an (nach Kreutz, Conradi et al. 1998).

Kopetzky et al. stellten auf der „8th World Wide Web Conference“ einen Prototyp vor, der Webseiten-Thumbnails zur Darstellung von Link-Previews einsetzt (Kopetzky & Mülhäuser 1999). Die Thumbnails erscheinen als kleines „Popup-Element“ (vergleichbar mit einem Tooltip), wenn man die Maus über einen Link-Anker bewegt. Handelt es sich bei dem Verweisziel nicht um eine Webseite, so wird stattdessen ein „Symbol“ angezeigt, das den „Typ“ des Links repräsentiert. Hierunter verstehen die Autoren Verweise auf E-Mail-Adressen oder FTP-Downloads. Fehlerhafte Links sollen durch ein Popup mit einem Sackgassen-Symbol gekennzeichnet werden. Über die Größe der Thumbnails und Symbole wird keine konkrete

Angabe gemacht.[141] Das vorgestellte System funktioniert mittels eines selbst entwickelten Intermediaries (s. Kapitel 8.4), der JavaScript-Code in die Webseiten einbettet, der zur Darstellung der Tooltips dient. Die Thumbnails werden auf einer zweiten Workstation im Hintergrund generiert, während der Benutzer sich eine Webseite ansieht. Wie die Autoren selbst anmerken, hat diese Lösung den Nachteil, dass sie eine erhebliche Bandbreite für das Herunterladen aller Zieldokumente einer Seite (inklusive Grafiken) erfordert, die Erstellung der Thumbnails sehr lange dauern kann und die Lösung aufgrund der zweiten Workstation recht aufwendig ist. Evaluationsergebnisse zu dem System wurden nicht präsentiert.

Marchionini et al. schlugen im Jahr 2000 ein Konzept und Framework namens *AgileViews* vor, das Benutzern bei der Navigation helfen sollte, indem sie eine bessere Übersicht über den Informationsraum („Information Space“) erhielten (Marchionini, Geisler et al. 2000). Nach Ansicht der Autoren benötigen Menschen mehr Informationen als für sie gegenwärtig im Web verfügbar sind, um sich sicher orientieren zu können und Informationen einfacher zu finden. Das Konzept sieht vier Elemente vor:

- Der *Overview* gibt dem Benutzer eine Übersicht, wohin er navigieren kann.
- Der *Preview* zeigt, wohin ein konkreter Link führt.
- *Review* (bzw. von den Autoren auch *History* genannt) eröffnet dem Anwender, wo er bereits war und welche Suchvorgänge er durchgeführt hat.
- Der *Peripherial View* bietet einen Kontext für die aktuelle Seite. Hierzu gehören Informationen über alle Fenster, die zwar auf dem Bildschirm, nicht aber im Blickbereich des Nutzers sind.[142]

Dem Konzept folgend haben die Autoren zur Optimierung der Link-Navigation im Web das „*Enriched Links Framework*“ entworfen, das mithilfe von interaktiven Tooltips für jeden Link drei Optionen darstellt: *Preview*, *Overview* und *History* (s. Abb. 63). Der Reiter *Preview* zeigt ein Thumbnail des Zieldokuments, den Titel und die Größe in Bytes. *Overview* präsentiert Informationen zu den Links und eingebundenen Medien des Link-Zieles, und *History* gibt eine Übersicht zur Gesamtzahl der Zugriffe auf die Seite, sortiert nach Zeitintervall und Top-Level-Domain der Besucher. Leider führen die Autoren nicht auf, wie das System genau realisiert wurde (oder werden sollte) und ob sie es evaluiert haben.

---

[141] Den Abbildungen nach zu urteilen sind die Symbole mit ca. 60x60 Pixel größer als die meisten Icons in grafischen Benutzungsschnittstellen. Die Thumbnails sind mit ca. 120x120 Pixel zwar zu klein, um Texte in ihnen lesen zu können, aber in den meisten Fällen groß genug, eine Website wiederzuerkennen (siehe Kaasten, Greenberg et al. 2001).

[142] Die Konzepte des Frameworks *AgileViews* von (Marchionini, Geisler et al. 2000) sind somit weitgehend vergleichbar mit den vier Nivergelt'schen Fragen („Wo bin ich? Was kann ich hier tun? Wie kam ich hierher? Wohin kann ich und wie komme ich dorthin?“), die er bereits Anfang der 1980er Jahre als Grundlage für eine funktionierende Mensch-Maschine-Kommunikation definiert hatte (Nievergelt 1983).

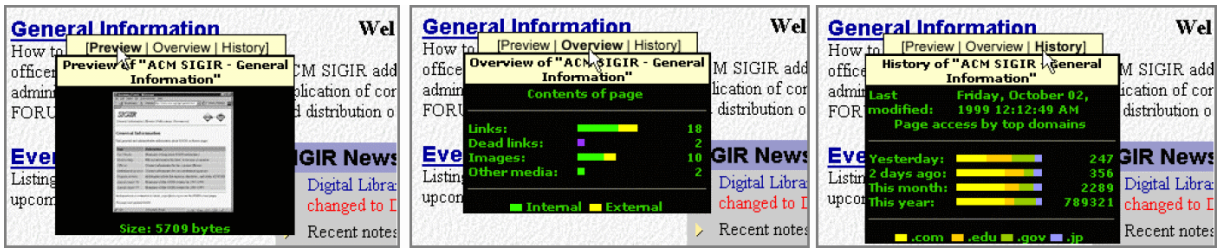


Abb. 63: Die drei Vorschau-Grafiken des *Enriched Link Frameworks* (nach Marchionini, Geisler et al. 2000).

Harper et al. von der Universität Manchester haben wenige Jahre später unterschiedliche Methoden entworfen, mit denen man Links im Web für Benutzer automatisch verständlicher machen kann, um die Navigation zu vereinfachen (Harper, Goble et al. 2004; Harper, Yesilada et al. 2004). Dabei verfolgen sie im Wesentlichen zwei Ansätze: Erstens soll der „Link-Kontext" optimiert werden, sodass der vom Link-Marker hervorgehobene Text immer den gesamten semantisch relevanten Bereich umfasst. Auf diese Weise versuchen die Autoren, Anker-Texte wie „Click here" und solche, die nur einzelne Wörter umfassen, durch verständlichere Beschreibungen zu ersetzen. Die Autoren haben hierzu diverse Heuristiken entwickelt, die automatisch den Bereich des Link-Ankers erweitern (s. Abb. 64). Zweitens werden Preview-Informationen für alle Link-Ziele generiert, die das System automatisch aus dem Titel und dem Inhaltsbereich des Zieldokuments extrahiert und dann als „`title`"-Attribut in die Anker-Elemente des HTML-Codes einfügt. Browser zeigen diese Information als Tooltip an (s. Tooltip in Abb. 64 rechts), wenn ein Benutzer die Maus über den Link bewegt (s. auch Kapitel 5.3.7).

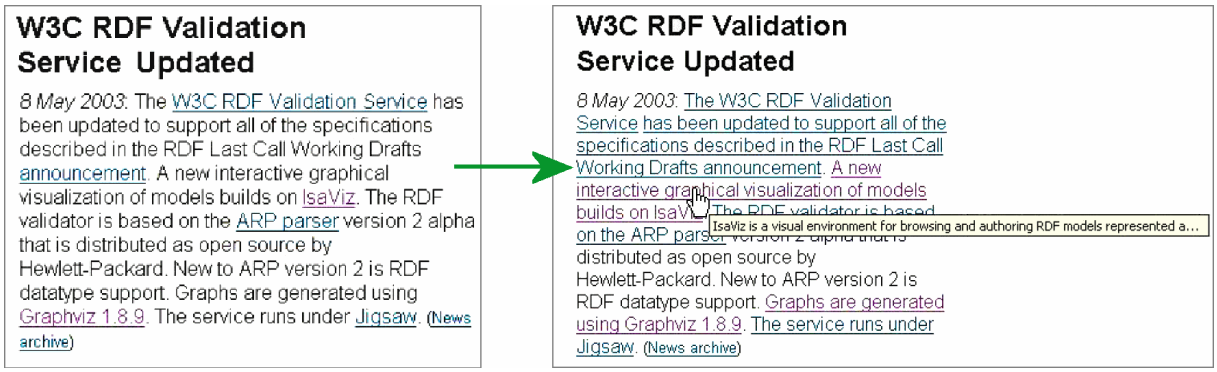


Abb. 64: Erweiterung des Link-Kontextes und Einbindung von Preview-Informationen für die Links in einer Webseite (nach Harper, Goble et al. 2004).

Das System basiert auf einer „Middleware",[143] die als Modul im Webserver integriert ist und die Anpassungen (das „Transcoding", vergl. Abschnitt 8.4.2.4) automatisch während der Übertragung für alle Seiten der Website vornimmt (Harper, Goble et al. 2004).

Die Konzepte wurden mit 42 Teilnehmern in einer kontrollierten Studie evaluiert, bei der die Teilnehmer das Original mit der erweiterten Version unterschiedlicher Webseiten anhand mehrerer Kriterien verglichen. Die Studie hat zahlreiche statistisch signifikante Ergebnisse

[143] Diese Middleware hat damit Ähnlichkeiten mit der in dieser Arbeit vorgestellten Intermediary-Architektur (siehe Kapitel 8), allerdings ist sie nicht so flexibel einsetzbar und auf den Einsatz auf Server-Seite beschränkt.

geliefert, wobei die Bewertung der Teilnehmer von der Art des Links abhing. Bei *assoziativen Links*[144] (sie verknüpfen inhaltlich verwandte Informationen miteinander) fanden die Teilnehmer sowohl die Erweiterung des Kontextes als auch die Tooltips hilfreich und zu bevorzugen, da so die Links deutlich aussagekräftiger seien (Harper, Yesilada et al. 2004). Bei *referenziellen Links* (sie verknüpfen meist Substantive oder Namen mit Dokumenten, die genauere Informationen zu dem Begriff bieten, s. Abschnitt 4.5.1.1) waren die Ergebnisse hingegen nicht eindeutig: Die Teilnehmer haben die nicht erweiterte Version bevorzugt, wenn sich der Link-Anker im Fließtext befand, da sie die Link-Anker bereits eindeutig fanden (Harper, Yesilada et al. 2004); stellte der Anker des referenziellen Links die Überschrift eines Absatzes dar, so wurde der erweiterten Variante der Vorzug gegeben.

Alle fünf hier aufgeführten Forschungsprojekte präsentierten interessante Ansätze zur Optimierung der Link-Schnittstelle im Web. Allerdings ging keines von ihnen genauer auf die Vor- und Nachteile der unterschiedlichen Möglichkeiten zur Erweiterung der Link-Schnittstelle zur Darstellung zusätzlicher Link-Informationen ein, noch wurde untersucht, welche zusätzlichen Arten von Informationen von Anwendern bei der Navigation im Web benötigt werden. In (Noirhomme-Fraiture & Serpe 1998) wurden zwar die unterschiedlichen Möglichkeiten zur Darstellung von Link-Informationen spezifiziert, die Autoren führten aber keine Analyse oder Evaluation der Konzepte durch. Die einzige systematische Benutzbarkeitsevaluation der entwickelten Konzepte wurde von (Harper, Yesilada et al. 2004) vorgenommen, jedoch untersuchten Harper et al. primär den Textumfang von Link-Ankern, nicht aber, welche Methoden für die Darstellung von zusätzlichen Link-Informationen, welche Stärken und Schwächen sie aufweisen oder welche Informationen benötigt werden. Auch spätere Arbeiten zur Benutzbarkeit des Webs und zu den Problemen bei der Navigation und Orientierung im Web gehen nicht genauer auf diese Fragestellung ein. Inzwischen existieren einige Browser-Erweiterungen, die unterschiedliche Konzepte zur Erweiterung der Link-Schnittstelle realisieren (s. Kapitel 9.2.3), aber auch für diese Systeme liegen keine Studienergebnisse vor. Diese Situation war ein weiterer Motivationsfaktor für die systematische Entwicklung und Evaluation von Konzepten zur Erweiterung der Link-Schnittstelle im Web, die in den folgenden Kapiteln vorgestellt werden.

5.5 Konzeption einer erweiterten Hyperlink-Schnittstelle: HyperScout I

Auf Grundlage der Klassifikation impliziter Informationen zu Links und Ressourcen im World Wide Web (siehe Kapitel 4.5) und der Analyse der verfügbaren Methoden zur Hervorhebung von Link-Ankern (Abschnitt 5.2) und zur Visualisierung von Link-Informationen (Abschnitt 5.3) wurden im Rahmen dieser Arbeit mehrere Konzepte zur Erweiterung der Benutzungsschnittstelle von Hyperlinks im Web entwickelt, die dem Benutzer automatisch mehr Informationen für die Navigation als Link-Previews anbieten. Die Konzeption

[144] Die Konzepte *assoziativer Links* (auch „*semantische Links*“ in dieser Arbiet genannt) und *referenzieller Links* werden in Kapitel 4.5.1.1 erläutert. Siehe auch (Haas & Grams 1998a; Miles-Board, Carr et al. 2002).

des ersten Prototyps *HyperScout I* wird im folgenden Abschnitt vorgestellt. Kapitel 6 beschreibt Vorgehen und Ergebnisse der Evaluation von *HyperScout I.* Danach wird die Entwicklung und Evaluation des hierauf aufbauenden Konzepts *HyperScout II* präsentiert, das die erweiterte Link-Schnittstelle des ersten Prototyps optimiert und zusätzliche Techniken zur Darstellung von Link-Informationen einsetzt.

5.5.1 Anforderungen für Link-Previews im Web

Das erste in dieser Arbeit entwickelte Konzept mit dem Namen *HyperScout I* sollte möglichst viele der in Abschnitt 4.5 kategorisierten Daten zu Links und Link-Zielen als automatisch Link-Previews zugänglich machen, sofern sie für den Anwender eine *zusätzliche Information* über eine *möglicherweise unerwartete Eigenschaft* darstellten. Diese Strategie wurde gewählt, da alle unvorhergesehenen Eigenschaften von Links und Link-Zielen für den Benutzer bei der Navigation potenziell relevant sein könnten. Im Rahmen der Evaluation sollte unter anderem ermittelt werden, welche Relevanz die einzelnen Attribute jeweils aufweisen (s. Abschnitt 6.1.1).

Für die Darstellung der zusätzlichen Link-Informationen wurden aufgrund der verhergehenden Analysen (s. Abschnitt 5.3.9) die folgenden Anforderungen festgelegt:

- Die Informationen sollen möglichst nahe beim *Fokus der Aufmerksamkeit* erscheinen, damit sie von den Benutzern nicht übersehen werden.
- Die erweiterte Link-Schnittstelle muss mit den *existierenden Webseiten und Browsern kompatibel* sein. Hieraus folgt, dass die Link-Informationen in einer Ebene über den Dokumenten einzublenden sind, um Layout und Erscheinungsbild der Seiten nicht zu verändern oder zu beeinträchtigen.
- Damit die Link-Informationen keine wichtigen Informationen verdecken, sollten sie auf Anforderung, also „*on-Demand*“, erscheinen.
- Einige der Attribute (z. B. der Titel des Link-Ziels) sind nur *textlich*, nicht aber symbolisch oder farblich zu repräsentieren.
- Es mussten relativ *umfangreiche* und *detaillierte* Informationen zu den Links darstellbar sein.

Bei dieser Dissertation stehen grafische Browser für Desktop-Oberflächen wie der Microsoft Internet Explorer und Mozilla Firefox im Fokus, die von Benutzern ohne besondere situative oder körperliche Einschränkungen eingesetzt werden (s. Kapitel 1), da diese Gruppe bis heute den Hauptteil der Anwender des Webs darstellt (Fennah & Botterill 2010). Zur Einbeziehung weiterer Benutzergruppen, Anwendungsbereiche und Benutzungsoberflächen-Techniken (vergl. Kapitel 9.3.2) wären viele zusätzliche Faktoren zu berücksichtigen gewesen, die im Rahmen dieser Arbeit ausgegrenzt wurden (s. Kapitel 1.1).

Die Anforderungen zur Darstellung umfangreicher, textlicher Link-Informationen, die bei Bedarf beim Fokus der Aufmerksamkeit erscheinen, werden für den aufgeführten Anwendungsbereich insgesamt am besten durch Tooltips erfüllt (s. Abschnitt 5.3.7): Sie stellen ausreichend Platz zur Verfügung, werden erst eingeblendet, nachdem der Mauszeiger für eine gewisse Zeit über dem aktiven Element (also dem Link-Anker) stillgehalten wird und sie erscheinen im Aufmerksamkeitsbereich beim Mauszeiger.

Als Zeitverzögerung für das Erscheinen der Tooltips wurde aufgrund der Ergebnisse von Pretests (s. Kapitel 6.1.4) 0,8 Sekunden gewählt: Eine kürzere Zeit hatte zur Folge, dass die Tooltips häufig ungewollt erschienen, eine längere Verzögerung führte zu störenden Wartezeiten. Der Tooltip verschwindet, sobald der Mauszeiger das aktive Element verlässt. Bewegt der Benutzer den Mauszeiger direkt von einem Link-Anker zum nächsten und vergeht dazwischen eine Zeit von unter einer halben Sekunde, so bleibt das System in einem „Tooltip-Modus“ und der nächste Tooltip erscheint ohne Zeitverzögerung.

5.5.2 Gestaltung strukturierter Tooltips für Link-Previews

Im Gegensatz zu den kleinen, einzeiligen Tooltips mit Hinweisen zu Icons und anderen Elementen der grafischen Oberfläche von Desktop-Anwendungen (s. Abb. 57) sollten die Tooltips von *HyperScout I* wesentlich umfangreichere Informationen als Link-Preview darstellen. Eine weitere Herausforderung ergab sich daraus, dass die verfügbaren Link-Informationen je nach Art des Links und des Zielobjekts variierten. Beispielsweise ist kein „Seitentitel“ für einen Link zu einer Grafik-Datei verfügbar. Zudem sollten *keine trivialen Eigenschaften* in den Tooltips aufgeführt werden, um die Attribute auf die wesentlichen zu reduzieren. Handelt es sich z. B. beim Ziel-Objekt um eine HTML-Seite, so wird dies nicht aufgeführt, da es ein erwartungskonformes Merkmal ist, das keines Hinweises bedarf. Es ist somit von der Art und Verfügbarkeit der Informationen abhängig, *ob* und *an welcher Position* sie in einem Tooltip erscheint.

Um der Menge und Variabilität der in den Tooltips darzustellenden Informationen gerecht zu werden, wurde eine Methode benötigt, die eine Präsentation auf *verständliche* und *schnell erfassbare* Weise gewährleistet. Zum Entwicklungsbeginn der ersten Vorläufer von *HyperScout I* im Jahr 1999 waren mehrzeilige Tooltips noch ungebräuchlich, daher wurde ein neues, geeignetes Schnittstellenkonzept benötigt.

Eines der ersten Beispiele für mehrzeilige Tooltips sind die Datei-Hinweise im Explorer von Windows XP aus dem Jahr 2001 (s. Abb. 65). Sie sind gleichzeitig ein Negativbeispiel und demonstrieren die Unübersichtlichkeit großer Tooltips bei mangelnder Formatierung der Informationen. Windows XP stellt Namen und Werte der Dateiattribute lediglich durch einen Doppelpunkt getrennt und linksbündig dar. Eine klare Trennung von Name und Wert ist daher auf den ersten Blick schwierig. Der Tooltip in Abb. 65 (links) zeigt darüber hinaus in der dritten Zeile einen „Titel“, der bis in die Folgezeile geht, ohne dass dies kenntlich gemacht wird. Die vierte Zeile „Navigation“ erscheint daher wie ein eigenes Attribut. Der

zweite Doppelpunkt in der dritten Zeile reduziert die Verständlichkeit weiter. Die entsprechenden Tooltips wurden in Windows 7 aus dem Jahr 2009 vereinfacht, dafür zeigt der Explorer nun in einem reservierten Bereich unten im Fenster zusätzliche, formatiert dargestellte Informationen zur Datei an (s. Abb. 65 rechts).

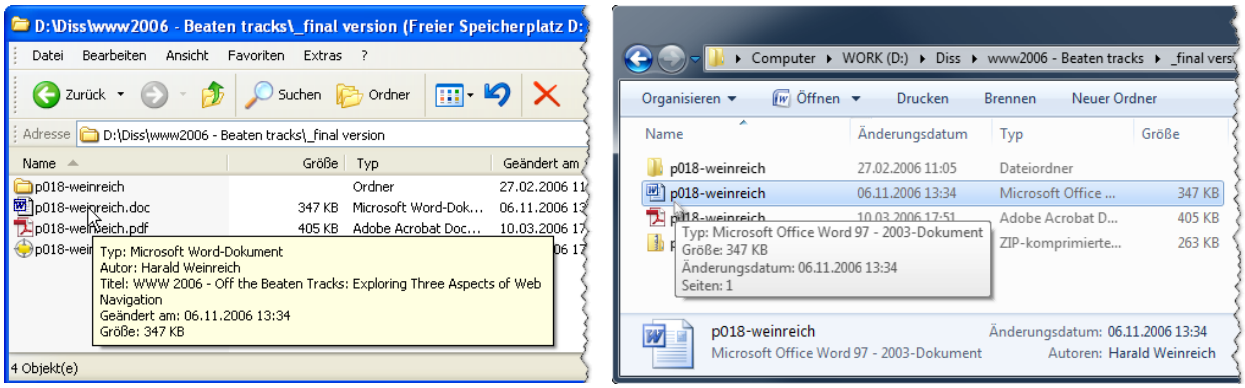


Abb. 65: Mehrzeilige Tooltips mit Dateiinformationen im Explorer von Windows XP (links) und von Windows 7 (rechts).

Damit Benutzer bei den Tooltips von *HyperScout I* eindeutig und schnell erkennen können, *welches* Attribut *wo* zu finden ist und sie in der Lage sind, Art und Inhalt der Daten eindeutig zu identifizieren, wurde bereits für die erste „*Proof-Of-Concept*"-*Implementation* von *HyperScout* gegen Ende 1999 (s. Weinreich & Lamersdorf 2000) ein neues Gestaltungskonzept für mehrzeilige Tooltips entwickelt, die eine *dreispaltige* Ausrichtung der Angaben verwendet (s. Abb. 66):

- Die erste Spalte zeigt ein *Icon*, das den Typ des Attributs symbolisiert und seine Identifikation vereinfachen soll.[145]
- Gefolgt wird das Icon von einem in Fettschrift hervorgehobenen *Bezeichner*, der das Link-Attribut beschreibt.
- In der dritten Spalte befindet sich der *Wert* des Attributes, der (linksbündig eingerückt) auch über mehrere Zeilen gehen kann.

Das Konzept wurde für den im Test eingesetzten Prototyp *HyperScout I* zusätzlich um mehrere Hintergrundfarben erweitert, um die unterschiedlichen Klassen von Link-Attributen zu kennzeichnen und zu gruppieren (s. Abb. 68 ff). Fehler wurden beispielsweise hellrot hinterlegt (Abb. 83 auf S. 178), Hinweise auf frühere Seitenzugriffe in Lila (Abb. 71 auf S. 172).

[145] Die im Prototyp verwendeten Icons wurden mehreren freien Icon-Bibliotheken entnommen und basieren auf früheren Arbeiten und Studien des Autors (s. Weinreich 1997).

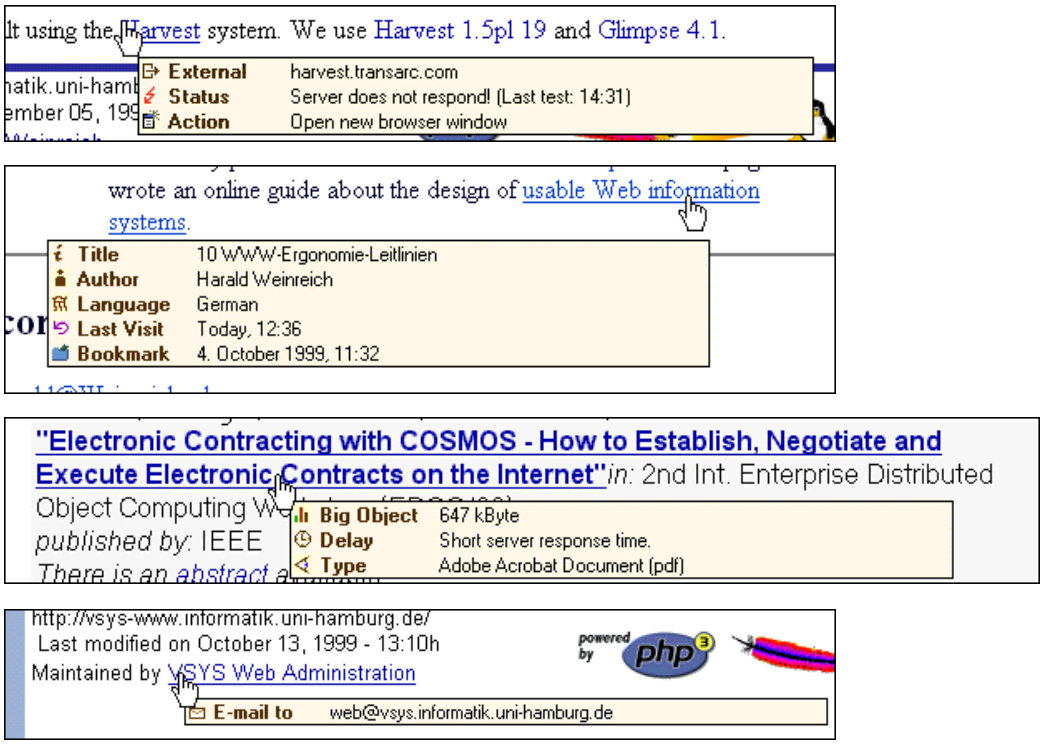


Abb. 66: Screenshots der „Proof-of-Concept"-Implementation von *HyperScout* (aus Weinreich & Lamersdorf 2000).

Diese Art der Präsentation wurde gewählt, da sich Icons als besonders geeignet für das schnelle Wiedererkennen von Informationen erwiesen haben (s. Horton 1994; Braden 1996: 498). Der Bezeichner des Attributs rechts neben dem Icon dient Benutzern, die mit den Icons nicht vertraut sind, und ist eine Gedächtnisstütze für routinierte Anwender. Eine weitere Motivationsbasis für die gemeinsame Verwendung von Icons und Bezeichner sind frühere Studien, bei denen diese Kombination für das schnelle Wiedererkennen von Items in Benutzungsschnittstellen signifikant bessere Ergebnisse lieferte als Icons oder Text alleine (Kacmar & Carey 1991).

5.5.3 Architektur und Funktionsweise von HyperScout I

Für die Evaluation der Konzepte von *HyperScout I* wurde ein Prototyp entwickelt, dessen Funktionsweise und Architektur im Folgenden kurz vorgestellt wird. Die genauen technischen Konzepte von *HyperScout* und Details der softwaretechnischen Entwicklung werden in den Kapiteln 7.5ff und 8 präsentiert.

HyperScout I ist als *„Plug-in"* für das *Scone-Framework* (s. Kapitel 8) implementiert worden. *Scone* ist ein im Rahmen dieser Arbeit entwickeltes Konzept zur prototypischen Realisierung von Erweiterungen für Webbrowser und –server, das als Java-Framework realisiert wurde. Es bietet aufgrund der verwendeten Konzepte unter anderem den Vorteil, weitgehend unabhängig vom eingesetzten Browser und von der Systemplattform zu sein (vergl. Abschnitt 8.3.3).

Die Teilnehmer der Studie sollten auf gewohnte Weise mit ihrem Browser auf das Web zugreifen können (s. Abschnitt 6.1.4). Daher war es notwendig, dass das prototypische

HyperScout-I-System mit allen populären Browsern funktionierte und Tooltips in beliebigen Webseiten darstellen konnte. Das System sollte zudem alle Aktionen des Benutzers aufzeichnen und die Daten für alle Link-Ziele in Echtzeit bereitstellen. Des Weiteren war wichtig, dass der Prototyp zuverlässig läuft, da für jeden Test eine Länge von mindestens einer Stunde vorgesehen war und die Benutzer während dieser Zeit auf einige Dutzend Seiten zugreifen sollten.

Die gewählte Lösung verwendet einen Intermediary (s. Abschnitt 8.4), der zu allen übertragenen Seiten Code hinzufügt. Er sorgt dafür, dass der Browser Tooltips zu allen Link-Ankern und Image-Maps anzeigt, sobald der Benutzer den Mauszeiger über die entsprechenden Bereiche führt. Die Bereitstellung der Tooltip-Daten im Browser und Darstellung der Tooltips[146] selbst wurde mithilfe eines Java Applets realisiert, das ebenfalls der Intermediary in die Seiten einbindet. Dieses *HyperScout-Applet* ist mittels CSS-Code im Dokument „versteckt", sodass es keinen Einfluss auf die sonstige Darstellung der Webseiten hat.

Der Funktionsweise von HyperScout I wird in Abb. 67 illustriert. Folgende Schritte werden für alle Anfragen des Webbrowsers ausgeführt:

1. Der Benutzer greift auf eine neue Webseite zu. Da *Scone* als Web-Proxy für den Browser konfiguriert ist, wird jeder Zugriff vom HyperScout-I-Plug-in verarbeitet.
2. HyperScout I modifiziert die übertragene Webseite: Der *LinkEventAdder* ergänzt JavaScript-Events zu den Link-Ankern und Image-Map-Elementen der Seite. Diese Event-Handler reagieren auf die Mausbewegungen und -klicks des Benutzers. Der „*PopupCode-Adder*" fügt HTML-Code in den „`head`"-Bereich der Seite ein, der zum Laden von zusätzlichem CSS- und JavaScript-Code sowie des HyperScout-Applets führt (siehe Schritt 4).
3. Der *LinkEventAdder* analysiert gleichzeitig die Zieladressen aller Links und übermittelt sie an den persönlichen Web-Crawler von Scone („Robot"). Sofern die Metadaten einer Ressource bisher nicht in der Datenbank des Systems vorhanden oder älter als einen Tag sind, werden sie per „Prefetching" (vergl. Abschnitt 7.2.1) von Robot aus dem Web abgerufen und in der Datenbank gespeichert.
4. Der Browser lädt per HTTP die zusätzlichen CSS- und JavaScript-Dateien sowie das Applet von dem Computer, auf dem das HyperScout-I-System läuft.[147] Der *Generator* des Intermediaries stellt die benötigten Dateien bereit.
5. Das Applet baut gleich nach der Initialisierung eine Socket-Verbindung zum RAS-Server (s. Abschnitt 8.6.5.1) von HyperScout I auf.[148] Über diese Socket-Verbindung werden die Metadaten zu den Links übertragen, sobald ein Tooltip aufgerufen wird.

---

[146] Zur Darstellung der Tooltips wird die Java-Klasse `java.awt.window` verwendet, die ein einfaches Fenster ohne Bedienelemente darstellt, das beliebig auf dem Bildschirm positioniert werden kann.

[147] Bei einer lokalen Installation kann dies beispielsweise auch der Computer des Benutzers sein.

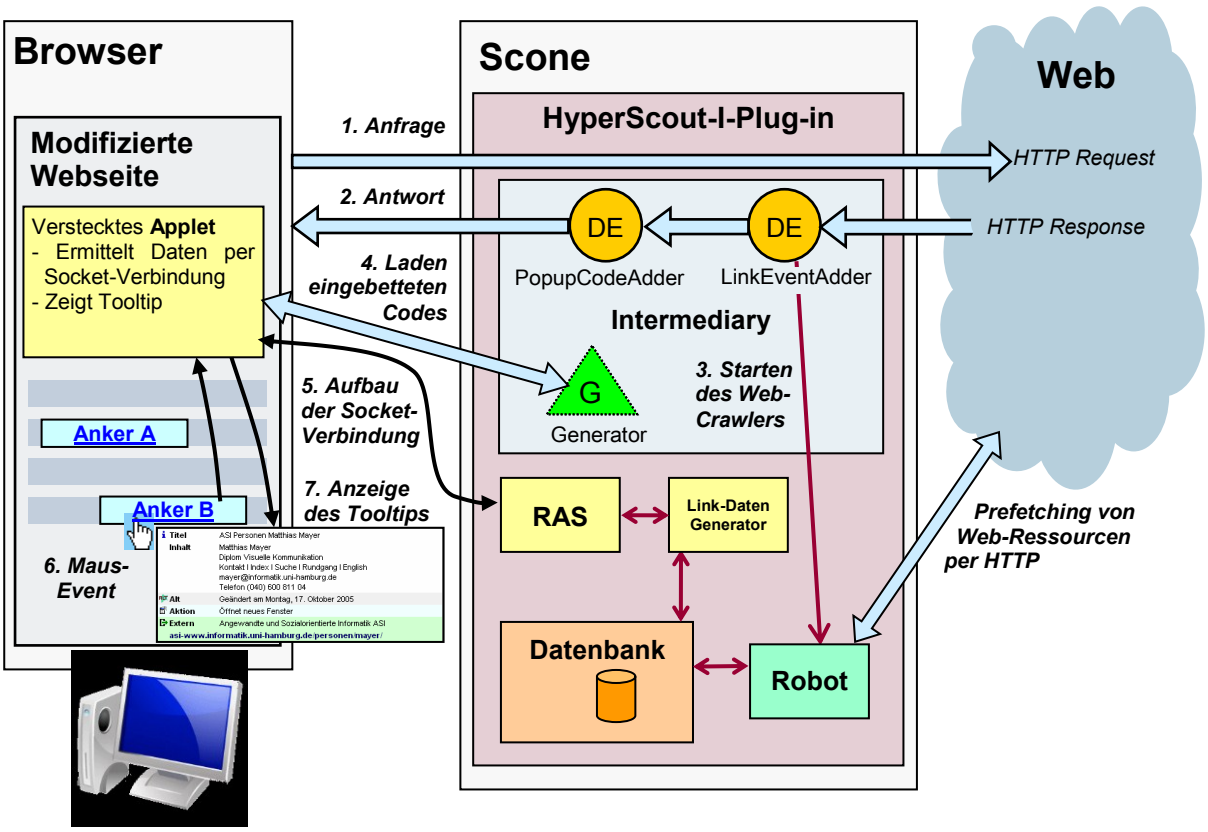


Abb. 67: Schematische Darstellung der Funktionsweise des Software-Prototyps HyperScout I.

6. Die hinzugefügten JavaScript-Events überwachen die Mausaktionen des Benutzers. Sobald er den Mauszeiger für 0,8 Sekunden über einem Link stillhält, wird per *Live-Connect* (Netscape 1996, 1.4) die ID des Links und die Position des Mauszeigers an das Applet übermittelt.

7. Das Applet ruft nun die Tooltip-Daten über die Socket-Verbindung vom HyperScout-I-System ab und stellt sie im Tooltip rechts unterhalb des Mauszeigers dar.[149] Sobald der Mauszeiger den Link-Anker verlässt, wird der Tooltip wieder ausgeblendet.

Der Einsatz eines Java Applets zur Darstellung der Tooltips hat sich in den Studien bewährt, da diese Methode einige Vorteile gegenüber JavaScript[150] bzw. Ajax[151] aufweist. Zum einen sind die Tooltips nicht (wie HTML-Elemente) auf den Dokumentenbereich des Browsers oder einen einzelnen Frame beschränkt, sondern können auch über das Browser-Fenster hinausgehen. Zudem kann der eingebundene JavaScript-Code kurz gehalten werden, wodurch

---

[148] Dies ist ohne Verletzung des Sicherheitsmodelles von Java Applets möglich, da das Applet von demselben Host geladen wurde.

[149] Sofern rechts unterhalb des Mauszeigers nicht genug Platz für den (häufig recht großen) Tooltip ist, wird er automatisch links bzw. oberhalb des Mauszeigers positioniert.

[150] Eine erste Proof-of-Concept-Implementation verwendete nur JavaScript und HTML-Code zur Darstellung der Tooltips. Diese Implementation war aber leider nicht für (längere) Tests geeignet, da die Übertragung der Seite zu sehr verzögert wurde und die Webbrowser aufgrund des umfangreichen Codes Stabilitätsprobleme zeigten. Auf weitere Details der Proof-of-Concept-Implementation wird in (Weinreich & Lamersdorf 2000) eingegangen.

[151] Die heutigen Ajax-Techniken waren zum Entwicklungszeitpunkt von *HyperScout I* noch nicht verfügbar.

sich das Risiko reduziert, dass er mit bereits vorhandenem JavaScript interferiert. Drittens geschieht die Bereitstellung der Tooltips ohne merkliche Verzögerung, da bereits während des Landens der Webseite eine Socket-Verbindung zum HyperScout-I-System zur Übertragung der Tooltip-Daten aufgebaut und dann offen gehalten wird.

HyperScout I sollte für die Studie möglichst viele der in Abschnitt 4.5 klassifizierten Informationen zu allen Links und Link-Zielen zeitnah und vollständig bereitstellen. Dafür war es notwendig, dass für jede aufgerufene Seite auch die Meta-Informationen zu allen Link-Zielen zugänglich waren. Um dies sicherzustellen, wurde erstens für die Studie eine schnelle Internetanbindung des PCs mit dem HyperScout-I-System gewährleistet, sodass sich die notwendigen Informationen zügig per Prefetching abrufen ließen. Zum anderen wurde *vor* jedem Testlauf die HyperScout-Datenbank mithilfe des Web-Crawlers mit aktuellen Daten zu allen in den Testaufgaben voraussichtlich besuchten Seiten gefüllt, sodass für die meisten der besuchten Seiten kein Prefetching notwendig war.

Im folgenden Abschnitt werden die in den Tooltips präsentierten Inhalte beschrieben. Die Evaluation und Weiterentwicklung von HyperScout I wird in Kapitel 6 vorgestellt.

5.5.4 Klassifikation der in den Tooltips dargestellten Inhalte

HyperScout I stellt umfangreiche Daten zum Link und Zielobjekt in den Tooltips dar. Die dargestellten Informationen basieren auf der in Kapitel 4.5 vorgenommenen Analyse und Klassifikation von impliziten Link-Informationen im Web. Die Benutzungsschnittstelle zur Visualisierung der Zusatzinformationen wird im Folgenden gemäß dieser Klassifikation spezifiziert und anhand von Beispielen veranschaulicht.

5.5.4.1 Der semantische Link-Typ

Obwohl HTML mehrere Attribute zur Typisierung von Hyperlinks anbietet, sind die meisten Links im Web nicht explizit typisiert (s. Abschnitt 4.4). Lediglich das „`title`“-Attribut wird in einigen Fällen genutzt.[152] Es stehen daher – abgesehen vom Text im Link-Anker – zumeist keine detaillierten Informationen über die Bedeutung des Links und des Verhältnisses, in das er Quelle und Ziel setzt, zur Verfügung. Für den Anwender ist es teilweise sogar schwierig, die zwei grundlegenden Typen *strukturelle* und *semantische Links* zu unterscheiden (vergl. Abschnitt 4.5.1.1).

Um dem Benutzer genauere Hinweise zur Bedeutung des Links zu geben, analysiert der Prototyp die URIs von Link-Anfang und Ziel. Die Pfadangaben in den URIs geben oft Aufschluss über das Verhältnis der Dokumente zueinander:

[152] Das „`title`“-Attribut wird, wenn vorhanden, im HyperScout-Tooltip ebenfalls dargestellt und hat die Bezeichnung „Link-Titel“.

- Befindet sich das Link-Ziel in einem untergeordneten Pfad zum Ausgangsdokument, so wird der Verweis als hierarchischer Link zu Detailinformationen interpretiert. Dies wird im Tooltip als *Detailpfad*, gefolgt von der Adresse der Ressource, angegeben (s. Abb. 68).
- Im umgekehrten Fall führt der Link zu einer übergeordneten Seite. Dies wird als *Übersicht* attributiert (s. Abb. 80 auf S. 176).
- Befinden sich beide Dokumente im selben Pfad, so werden sie derselben Hierarchieebene zugeordnet. Es wird *Datei* gefolgt vom Dateinamen ausgegeben (s. Abb. 79 auf S. 176).
- Interne Links, für die keine der drei obigen Beziehungen von Quell- und Zielpfad zutrifft, werden als *nicht-strukturelle* Verweise interpretiert und als *Querverweis* bezeichnet (s. Abb. 74 auf S. 173).
- Ein Link, der wieder zur *identischen Seite* führt, wird mit dem Hinweis „*Zeigt zu diesem Dokument*“ gekennzeichnet (s. Abb. 70).
- Verweise zur *Homepage* (bzw. Startseite) der Site werden entsprechend charakterisiert (s. Abb. 69).
- Sofern es sich um einen Sprung innerhalb einer Seite handelt (s. Abschnitt 4.5.1.3), erscheint im Tooltip die Angabe *Referenz* mit Angaben zum Zielanker (vergl. Abschnitt 5.5.4.3, Abb. 72 und Abb. 73 auf S. 172).

Bei den oben verwendeten Verfahren handelt es sich um Heuristiken, die häufig – aber nicht immer – inhaltlich korrekte Ergebnisse liefern, da die Struktur der URIs nicht mit der semantischen Struktur der Site übereinstimmen muss. Ein Beispiel für einen strukturellen Link, den die Heuristik des HyperScout-I-Prototyps irrtümlich als Querverweis klassifiziert, ist in Abb. 78 zu sehen.

5.5.4.2 Die Verteilungscharakteristik des Links

Die Verteilungscharakteristik eines Links bezieht sich darauf, ob sich Quelle und Ziel des Links auf derselben Site befinden (s. Abschnitt 4.5.1.2, *interne* vs. *externe* Links). Von HyperScout I werden Links, die zwei Ressourcen mit unterschiedlichem Domain-Namen verknüpfen, als *externe Links* gekennzeichnet. Interne Links werden nicht explizit hervorgehoben, es wird vielmehr der semantische Link-Typ angegeben (s. vorheriger Abschnitt 5.5.4.1).

Damit der Benutzer einen Eindruck vom Anbieter des Zielobjekts erhält, wird der Domain-Name dargestellt (s. Abb. 71). Wird nicht auf die Homepage verwiesen, so wird zusätzlich eine Beschreibung der Website aufgeführt, da Domain-Namen häufig nicht sehr aussagekräftig sind. Diese Beschreibung (Abb. 83) extrahiert das System aus dem Titel der Homepage der Website, wobei es häufige Zusätze wie „Homepage of“, „Willkommen bei“ entfernt.

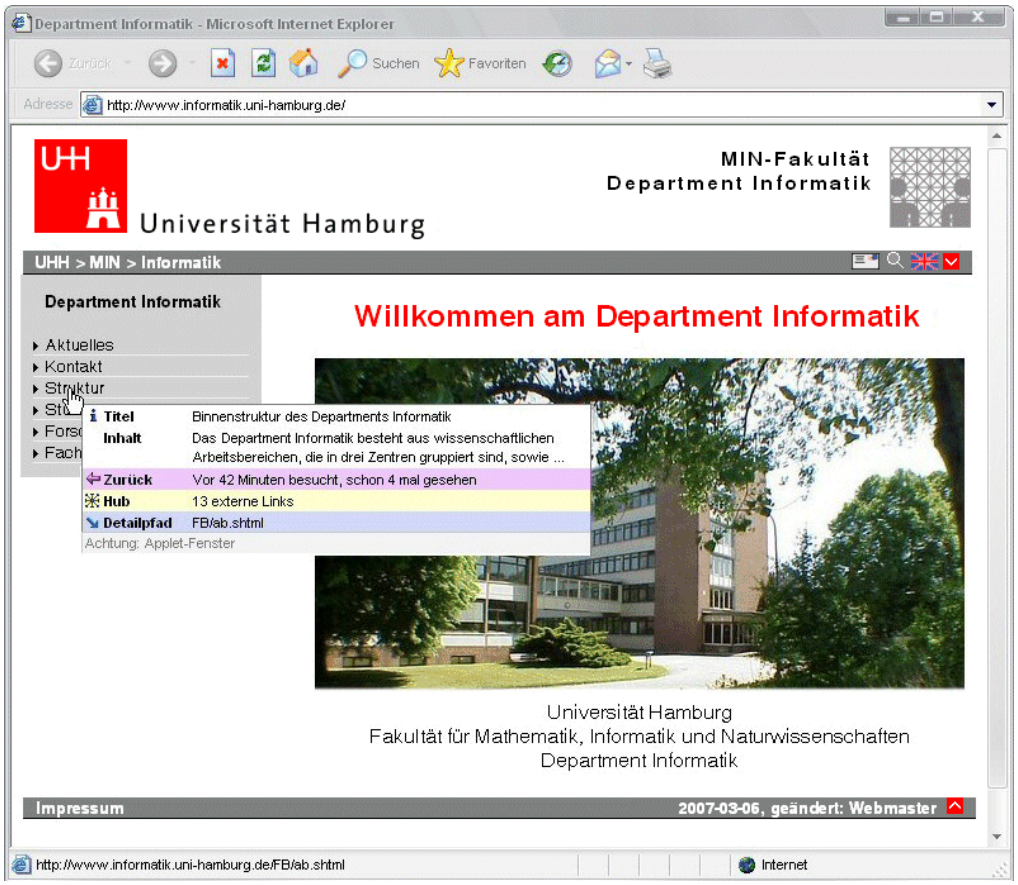


Abb. 68: Tooltip mit *Titel* und *Inhalt* des Link-Ziels. Der Benutzer hat es bereits viermal aufgerufen, und es befindet sich in einem *untergeordneten* Verzeichnis.

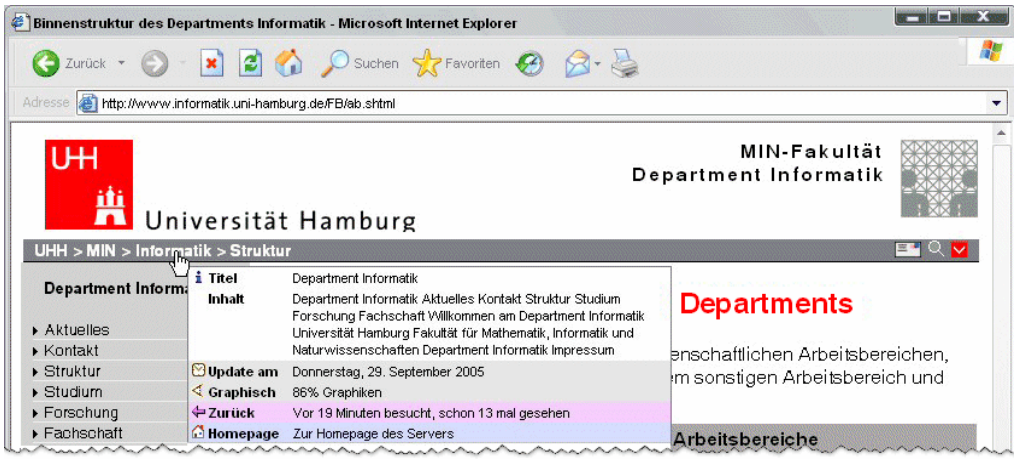


Abb. 69: Tooltip mit Angabe des letzten Updates. Das Zieldokument wird von Bildern und Grafiken dominiert („Grafisch“) und ist die Homepage der Website.

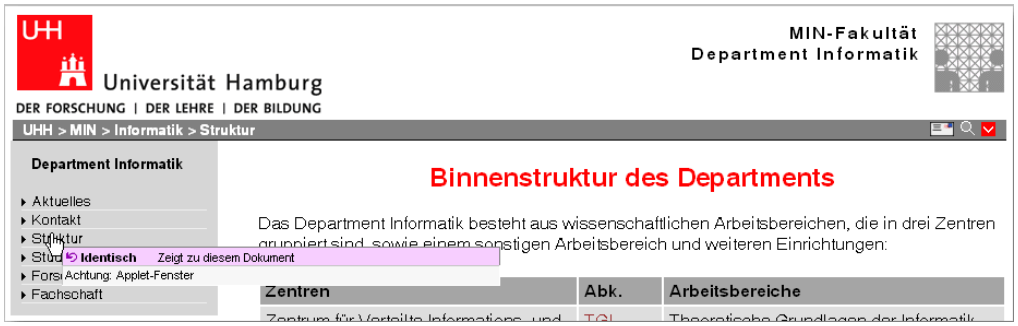


Abb. 70: Ein Link zur aktuell angezeigten Seite.



Abb. 71: Ein Link zur Homepage mit Hinweisen zum letzten Besuch auf der Seite und auf der Website.



Abb. 72: Link zum Fuß der Seite.

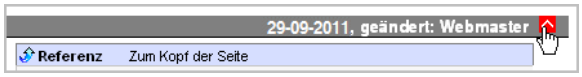


Abb. 73: Ein Sprung zum Seitenkopf.

5.5.4.3 Die Link-Aktion

Die Auswahl eines Links kann bei grafischen Webbrowsern zu vielen unterschiedlichen Link-Aktionen führen. Der Standard sind *Replacement Links* (s. Abschnitt 4.5.1.3), bei denen das Link-Ziel im selben Browser-Fenster dargestellt wird und es das aktuelle Dokument ersetzt. Sie werden daher von HyperScout nicht explizit gekennzeichnet.

Jumps, Verweise innerhalb der Seite, bei denen der Browser zu einer bestimmten Position im Dokument scrollt (auch *„within-page Links“* genannt, siehe: ISO9241-151 2008), werden bei HyperScout I als *„Referenz“* gekennzeichnet. Führt der Link zum Kopf oder Fuß der Seite, so erfolgt eine entsprechende Ausgabe (Abb. 72 und Abb. 73), ansonsten wird der Name des Zielankers angegeben (s. Abb. 91).

Eine weitere Link-Aktion sind *Popup Links* (nach Gloor 1990: 30); vergl. Abschnitt 4.5.1.3), die zur Darstellung des Link-Ziels ein neues Fenster öffnen. Der HyperScout-Prototyp fasst die unterschiedlichen Techniken zum Öffnen eines neuen Fensters (z. B. mittels Link-Attribut oder Javascript) zusammen (Abb. 74 und Abb. 86). Sofern ein Link den Inhalt eines anderen *Fensters* oder *Frames* steuert, wird dies ebenfalls aufgeführt.

Inzwischen werden häufig JavaScript und Ajax eingesetzt, um innerhalb von Webseiten neuartige Interaktionsformen zu bieten (Crane, Pascarello et al. 2005; Garrett 2005). Solche *programmatischen Links* erlauben viele Link-Aktionen, die ursprünglich für das Web nicht vorgesehen waren, wie *Stretchtext* und *Note-Links* (s. Abschnitt 4.5.1.3). Da die Interpretation der Script-Kommandos ein komplexes Vorhaben ist, werden solche Links in HyperScout I einheitlich durch *„Aktion: JavaScript“* gekennzeichnet (s. Abb. 75). Zukünftige Systeme könnten aber detaillierte Hinweise auf die zu erwartende Browser-Aktion liefern (s. Kapitel 9.3.4).

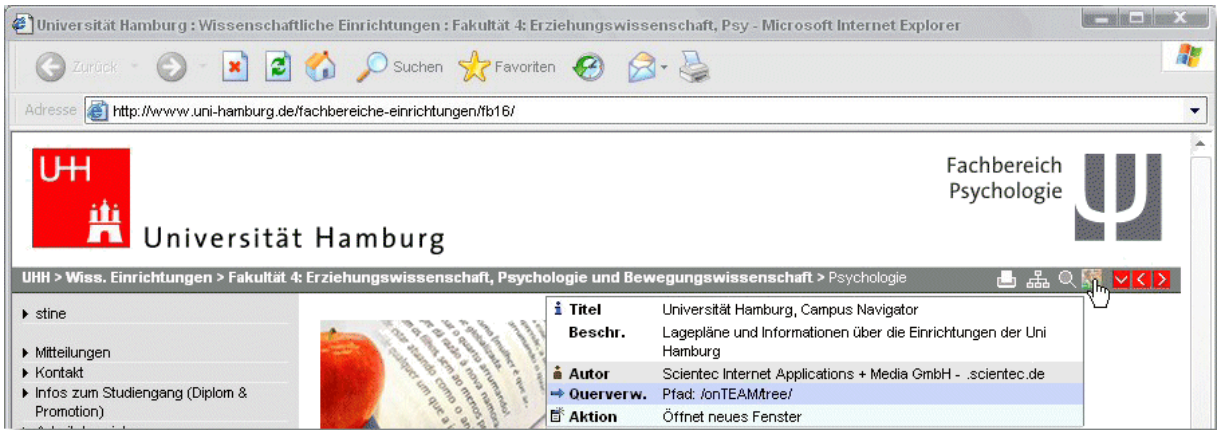


Abb. 74: Das Link-Ziel wird in einem neuen Browser-Fenster dargestellt.

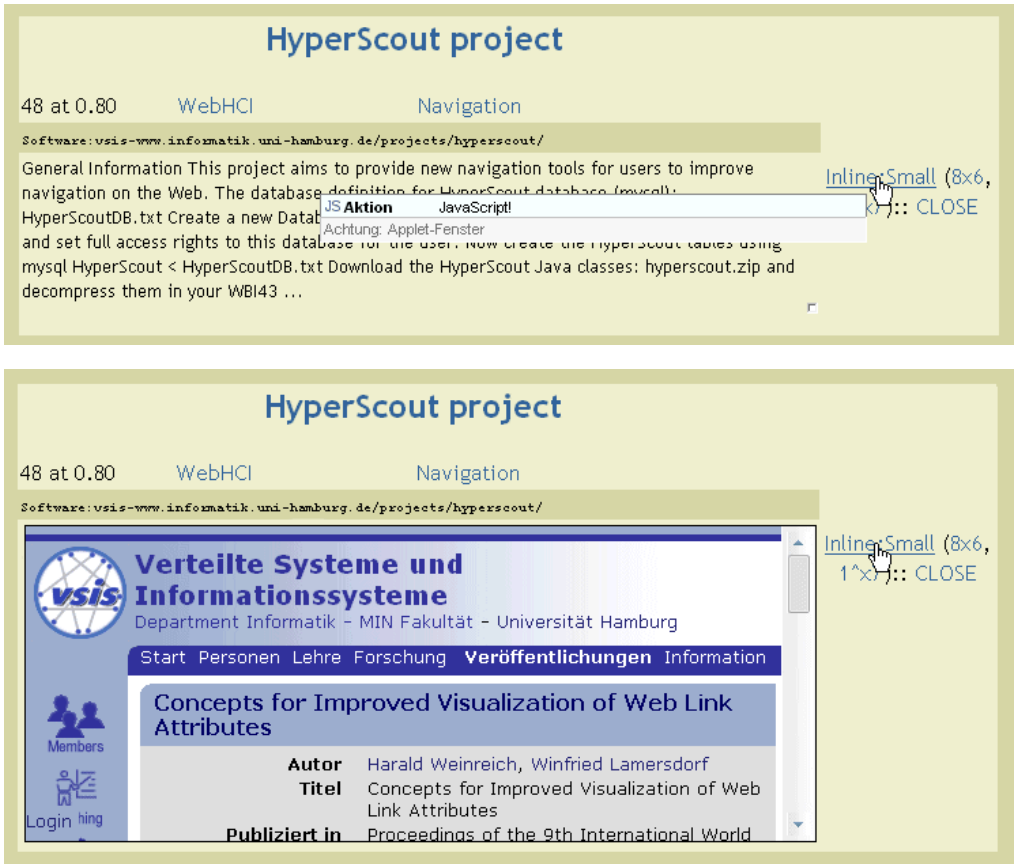


Abb. 75: Ein *Stretchtext-Link* im Webbrowser: Nach dem Anklicken des Ankers *„Inline Small“* (obere Grafik) wird mittels JavaScript ein Bereich eingefügt, in dem die Zielseite erscheint (untere Grafik).

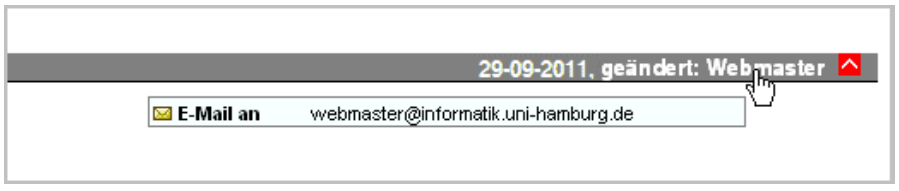


Abb. 76: Eine E-Mail-Adresse als Link-Ziel.

Links, die in der Zieladresse ein anderes Protokoll als „`http:`“ verwenden, werden gesondert behandelt, da sie aus dem Kontext des Webs herausführen und häufig – als Aktion – eine andere Applikation starten. Ein Beispiel sind „`mailto:`“-Links (Abb. 76): Im Tooltip erscheint als Aktion „E-Mail an“ gefolgt von der Empfängeradresse. Auf ähnliche Weise wird mit Protokollen wie „`ftp:`“ und „`telnet:`“ verfahren.

5.5.4.4 Informationen zum Inhalt des Link-Ziels

HyperScout I präsentiert zahlreiche Informationen zum Inhalt des Link-Ziels im Tooltip. Je nach Verfügbarkeit und Relevanz werden der *Titel*, der *Urheber*, das *Datum*, eine *Inhaltsangabe* und die *Sprache* des Dokuments angezeigt.

Der Seitentitel, Inhalt und Beschreibung

In der obersten Zeile der HyperScout-Tooltips findet man zumeist den *Seitentitel* des Link-Ziels (Abb. 68, Abb. 69, Abb. 77 etc.). Er wird dem „`title`“-Element der Seite entnommen (vergleiche auch Harper, Goble et al. 2004; Harper, Yesilada et al. 2004). Es folgt eine Inhaltsangabe. Diese ist entweder eine *Beschreibung*, die aus dem entsprechenden Meta-Element des Zieldokuments stammt (Abb. 74, Abb. 87), oder ein kurzer Auszug des *Inhalts* (z. B. Abb. 77), den das System aus dem Anfang des Content-Bereiches des Ziel-Dokuments generiert.[153]

Autor und Anbieter

Daten zum *Urheber* eines Dokuments sind im Web oft schwer zu ermitteln (s. Abschnitt 4.5.2.2). Bei manchen Dokumenten lässt sich der *Autor* aus dem gleichnamigen Meta-Element entnehmen (s. Abb. 74, Abb. 77). Der *Anbieter* der Informationen bzw. einer Site wird hingegen aus dem Titel der Site-Homepage extrahiert (s. Abb. 83 und Abschnitt 5.5.4.2).

Der Aktualisierungszeitpunkt

Ebenso fehlen im Web zumeist Angaben zum Erstellungszeitpunkt und zur letzten Aktualisierung einer Seite. Dies kann für Benutzer problematisch sein, wenn sie die Aktualität eines Dokuments überprüfen wollen. Studien haben dabei gezeigt, dass das Web immer dynamischer wird und sich die Inhalte von Seiten in immer kürzeren Abständen ändern (Fetterly, Manasse et al. 2003; Adar, Teevan et al. 2009; Teevan, Dumais et al. 2010). *HyperScout I* nutzt das Attribut „`Last-Update`“ aus dem HTTP-Protokoll (Fielding, Gettys et al. 1999), um den Zeitpunkt der letzten *Aktualisierung* einer Ressource anzuzeigen. Im letzten Monat überarbeitete Seiten werden als „Neu“ hervorgehoben (s. Abb. 77), wogegen Seiten, deren Update über ein Jahr her ist, als „Alt“ gekennzeichnet sind.

---

[153] Es wird ein heuristisches Verfahren verwendet, das Navigationsbereiche überspringt und nach der ersten Überschrift und dem ersten längeren Fließtext der Seite sucht. Ein Vergleich solcher Verfahren zur automatischen Generierung von Beschreibungen für Webseiten findet man in (Harper, Yesilada et al. 2004).

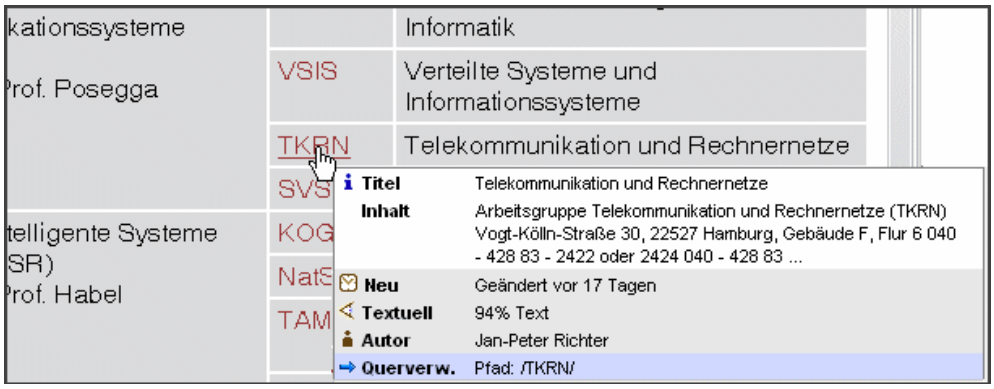


Abb. 77: Ein Link zu einem aktuellen Dokument, das fast nur aus Text besteht.

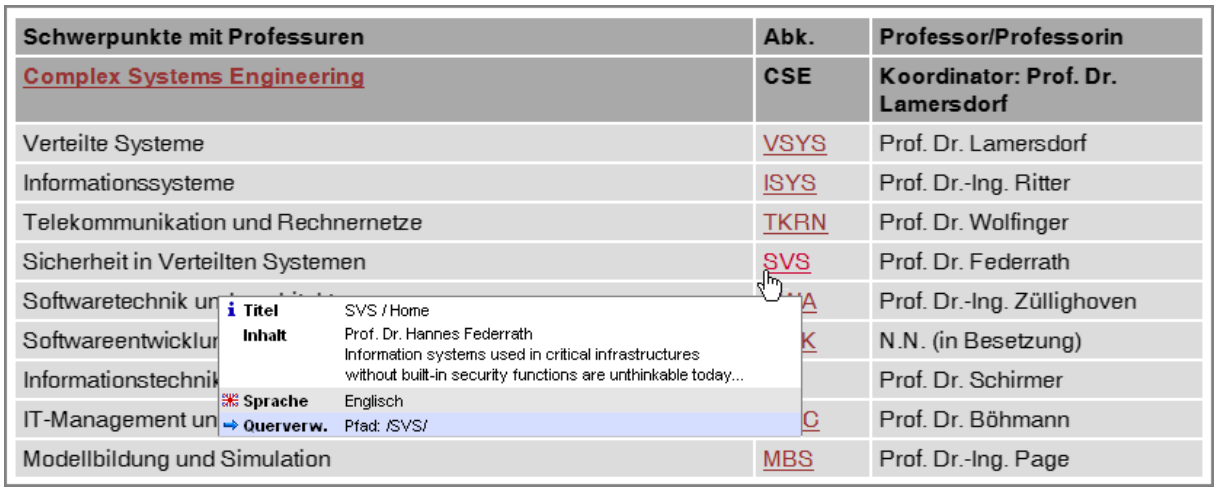


Abb. 78: Ein Querverweis auf ein englischsprachiges Dokument.

Die Sprache

Die *Sprache* des Zieldokuments ermittelt das System mithilfe einer Heuristik. Sie berücksichtigt die jeweils 30 häufigsten Wörter aus dem Deutschen, Englischen, Französischen und Spanischen und zählt ihr Vorkommen innerhalb der Seite. Dominieren die Wörter einer dieser Sprachen, wird angenommen, dass das gesamte Dokument in ihr verfasst ist (Abb. 78). Die Sprache des Link-Ziels wird im Tooltip dargestellt, sofern sie nicht mit der Sprache der aktuellen Seite übereinstimmt.

5.5.4.5 Der technische Typ des Zielobjektes

Der Dateityp des Link-Ziels lässt sich im Web – ähnlich wie bei lokalen Dateien auf dem PC – häufig bereits aus der Endung des Dateinamens erkennen. Der HyperScout-Prototyp berücksichtigt zusätzlich den *MIME Media Type* (Freed & Borenstein 2006) des Zielobjektes und bestimmt so den „technischen Typ“ des Link-Ziels (s. Abschnitt 4.5.3.1). Der getestete Prototyp ist in der Lage, über 40 verschiedene Typen zu unterscheiden und mit charakteristischem Icon versehen im Tooltip darzustellen, beispielsweise „Adobe Acobat Dokument“ (s. Abb. 79), „Microsoft Word Dokument“, „ZIP-komprimierte Datei“ oder „Ausführbares Programm (PC/Windows)“.



Abb. 79: Ein Beispiel für einen Link auf eine PDF-Datei.



Abb. 80: Das Link-Ziel ist eine große Datei mit vielen Site-internen Links. Im Beispiel zeigt die Heuristik zur Bestimmung des Inhaltes Schwächen.

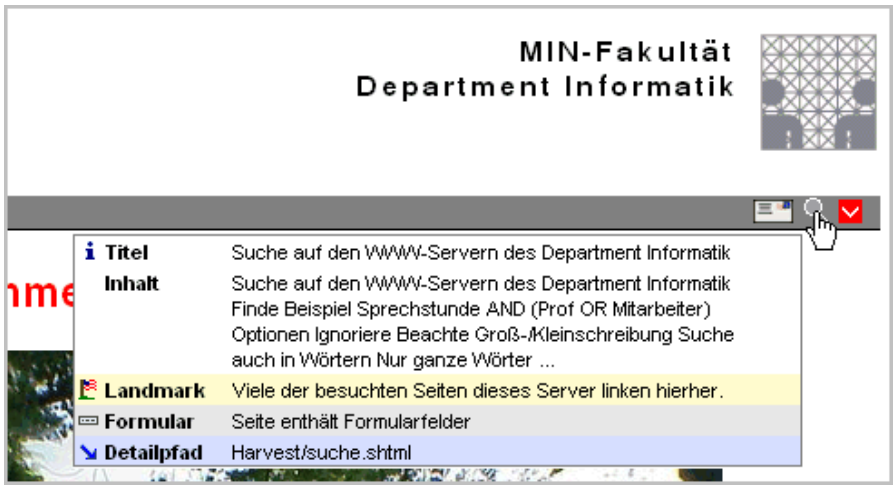


Abb. 81: Ein Verweis auf eine zentrale Landmark-Seite mit Formularfeldern.

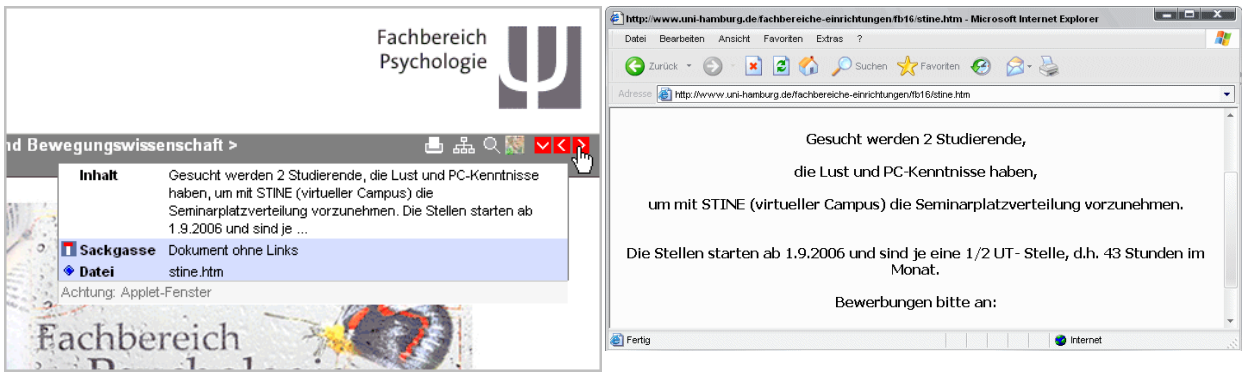


Abb. 82: Der Verweis im linken Bild führt zu der „Sackgassen-Seite“ ohne Titel und Links im rechten Bild.

5.5.4.6 Der Medientyp des Link-Ziels

Der Medientyp des Zielobjektes (u.a. Audio- oder Video-Datei; s. Abschnitt 4.5.3.2) wird von HyperScout I dahin gehend berücksichtigt, dass bei HTML-Seiten die eingebetteten Objekte analysiert und der Umfang des Fließtextes analysiert werden. Das System unterscheidet zwischen den folgenden Typen:

- *Grafische Webseiten*, bei denen Bilder und Grafiken einen Großteil der Seitenfläche ausmachen (s. Abb. 69 auf S. 171);
- Seiten, die Musik oder Sprache im Hintergrund abspielen (*Audio*) und
- Dokumente, die hauptsächlich aus Fließtext bestehen (*Textuell*, Abb. 77).

Andere Dateitypen als HTML werden vom Prototyp nicht analysiert, zumal HTML-Dateien den Großteil der Ressourcen im Web ausmachen.

5.5.4.7 Der topologische Typ des Zielobjektes

Um den topologischen Typ (s. Abschnitt 4.5.3.3) des Link-Ziels zu bestimmen, wird die Link-Struktur ausgewertet: HyperScout I berücksichtigt sowohl die in der aktuellen Seite eingebetteten Links als auch alle bereits in der Datenbank des Prototyps gespeicherten Links, die auf das Link-Ziel verweisen. Auf Basis dieser Daten werden folgende Typen unterschieden:

- *Index-Seiten* verfügen über überdurchschnittlich viele[154] Site-interne Links und wenig Fließtext (s. Abb. 80).
- *Hub-Seiten* bieten mehr als 10 externe Links zu anderen Websites (Abb. 68 auf S. 171).
- *Landmark-Seiten* zeichnen sich dadurch aus, dass fast alle anderen Seiten der Website auf sie verweisen (Abb. 81).
- *Sackgassen* sind Dokumente ohne weiterführende Links (Abb. 82).
- Seiten mit Eingabefeldern werden mit dem Hinweis *Formular* gekennzeichnet (Abb. 81). Diese Eigenschaft wird aufgeführt, da solche Seiten eine andere Interaktionsform als „typische“ Hypertext-Dokumente mit Links erfordern.

[154] Zur Ermittlung wird die Link-Zahl des Ziels mit der aktuellen Seite und der Site-Homepage verglichen. So wird vermieden, dass bei Websites, die über Navigationsbereiche mit vielen strukturellen Links verfügen, jede Seite als *Index-Seite* gekennzeichnet wird. Erst wenn eine Seite mehr als 20 interne Links *mehr* als beide Vergleichsseiten hat, erfolgt eine Klassifikation als *Index-Seite*. Diese heuristischen Verfahren wurden durch zahlreiche Tests ermittelt.

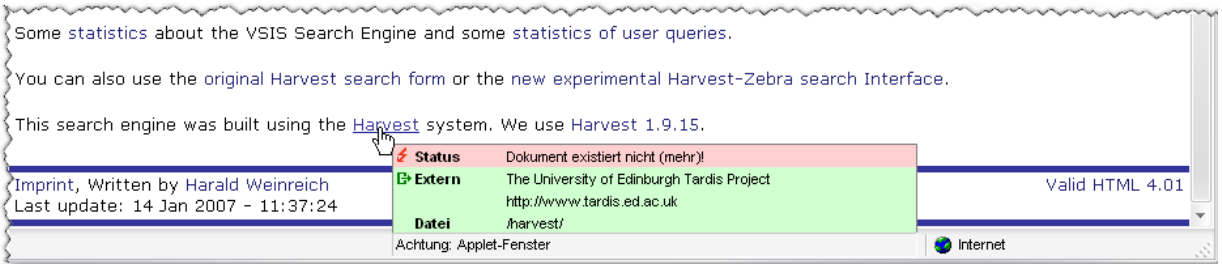


Abb. 83: Das Zieldokument dieses Links wurde entfernt. Das Beispiel zeigt zusätzlich eine Beschreibung der Site, die aus dem Titel der Homepage gewonnen wurde.

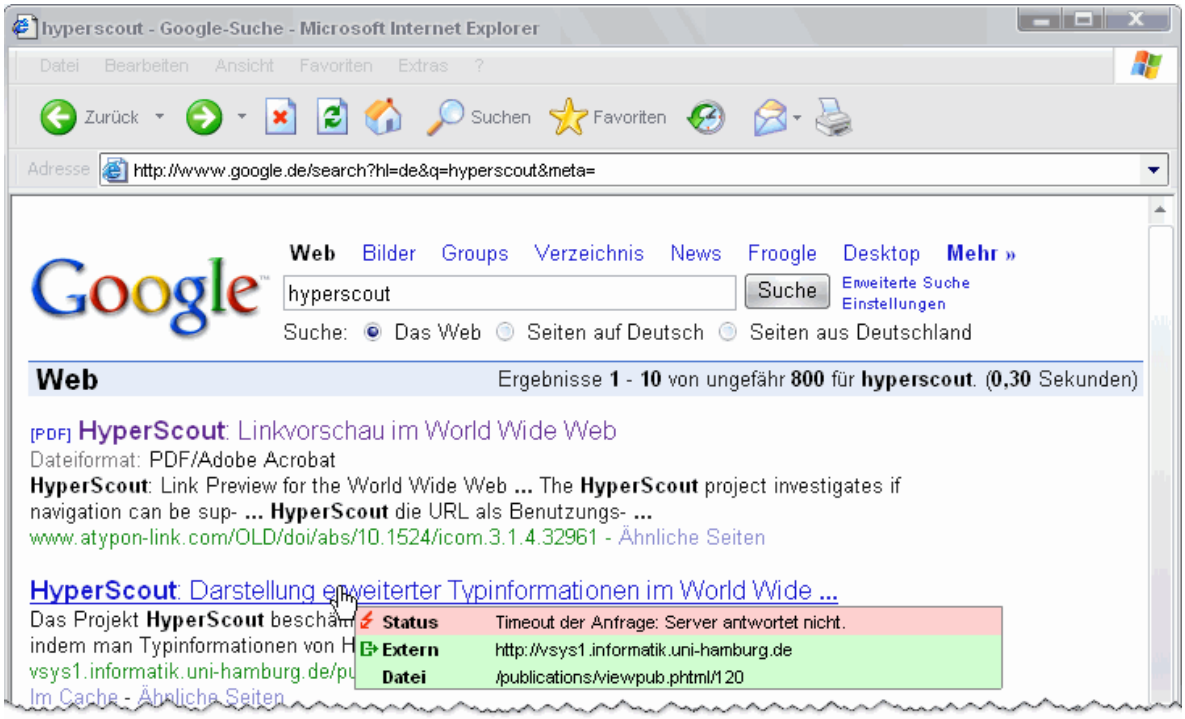


Abb. 84: Die Site http://vsys1.informatik.uni-hamburg.de/ war noch bei Google indexiert, aber nicht mehr online verfügbar (vom August 2006, der Server wurde kurz zuvor ausrangiert).

5.5.4.8 Die Verfügbarkeit des Ziel-Objektes

Eine bedeutende Eigenschaft von Web-Ressourcen ist ihre Verfügbarkeit. Der Zugriff auf Objekte kann beispielsweise durch eine Passwort-Abfrage (s. Abb. 89 auf S. 194) oder für Computer mit bestimmter IP-Adresse eingeschränkt werden. Der HyperScout-Client erstellt aus den Rückgabecodes der Webserver entsprechende Angaben in den Tooltips.

Da im World Wide Web darüber hinaus die Integrität der Hyperlinks nicht sichergestellt werden kann, kommt es häufiger vor, dass eine Seite (Abb. 83) oder sogar eine gesamte Site (Abb. 84) nicht (mehr) zugänglich ist. Beides wird von HyperScout-I-Prototyp entsprechend gekennzeichnet.

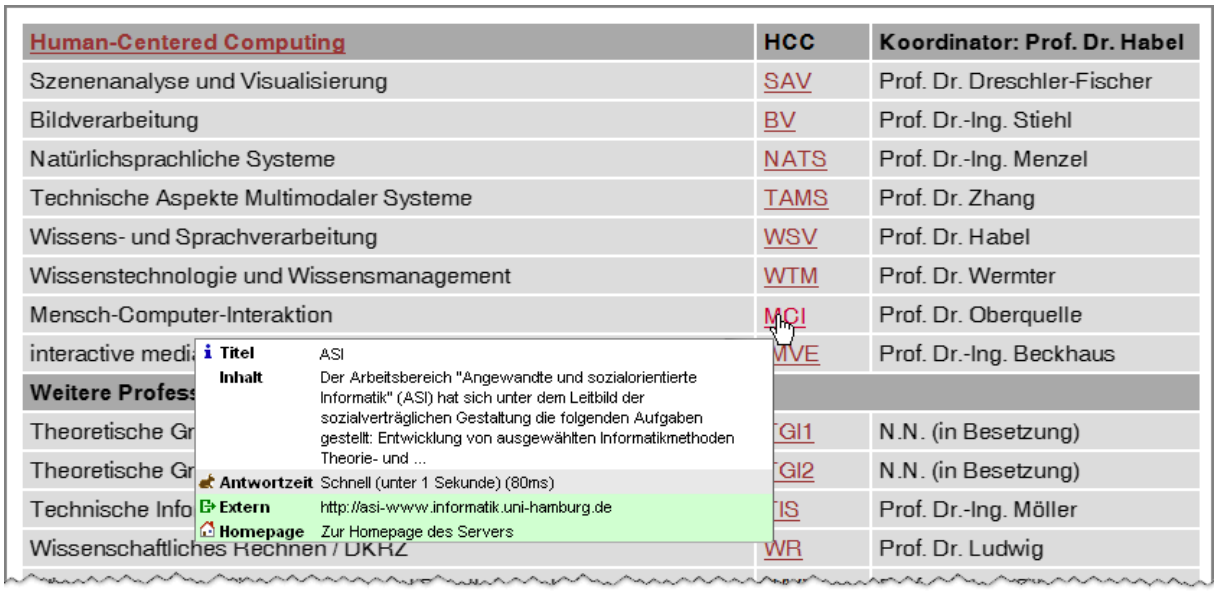


Abb. 85: Ein externer Link zur Homepage eines schnell antwortenden Servers.

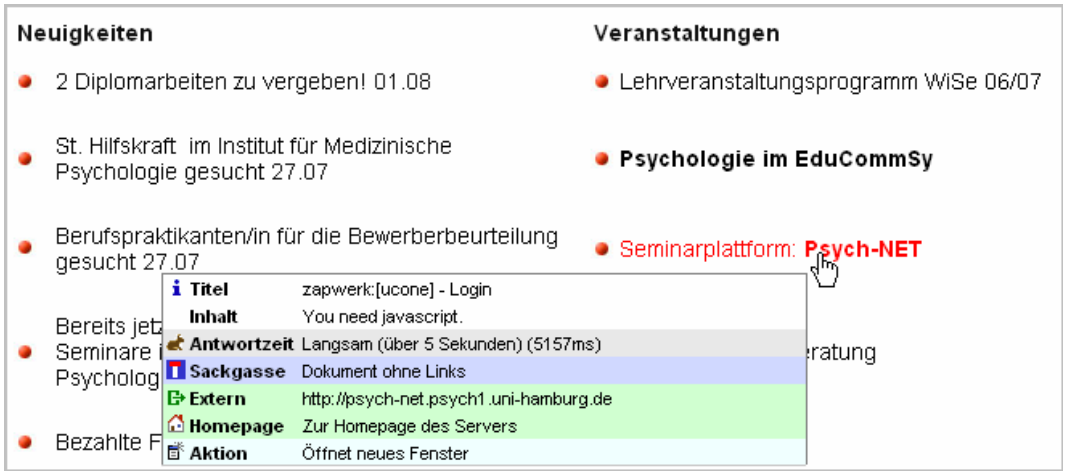


Abb. 86: Ein externer Link zu einem Server mit gegenwärtig langsamer Antwortzeit. Das Link-Ziel wird in einem neuen Fenster dargestellt und benötigt JavaScript.

5.5.4.9 Die Latenzzeit bei der Navigation

Die Zeitverzögerung, bis eine Web-Ressource im Browser des Anwenders erscheint, lässt sich leider nicht genau vorhersagen, da sie von vielen, zum Teil unvorhersehbaren Faktoren abhängt (s. Abschnitte 3.3.1 und 4.5.4.2). Aus diesem Grund werden stattdessen zwei maßgebliche Faktoren für lange Ladezeiten angezeigt: die *Größe des Objektes* und die *Antwortzeit* des Servers.

Die Dateigröße des Link-Ziels wird dem Benutzer ab 40 Kilobytes[155] dargestellt. Zusätzlich wird bei *externen* Links die Antwortzeit des Servers aufgeführt, sofern sie unter einer Sekunde („schnell“) oder über fünf Sekunden („langsam“) liegt (s. Abb. 85 und Abb. 86). Die Ermittlung der Antwortzeit erfolgt über eine zeitnahe HTTP-Head-Anfrage auf die Homepage der Site.

[155] Diese Schranke wurde gewählt, da entsprechende Seiten überdurchschnittlich groß sind (Lyman, Varian et al. 2000; Lyman, Varian et al. 2003; King 2006) und der Download per Modem oder ISDN über 8 Sekunden dauert.

5.5.4.10 Informationen zur Benutzung

Der für den Test eingesetzte HyperScout-I-Prototyp zeigt dem Anwender unterschiedliche Daten zum früheren Zugriff auf die Ressourcen des Webs. Zum einen erscheint der *letzte Zugriff* der Person auf das Ziel eines Links (vergl. Abschnitt 4.5.5.1). Diese Angabe wird je nach verstrichenem Zeitraum in Stunden und Minuten (für am selben Tag angesehene Seiten, s. Abb. 69), in Tagen (bei unter 90 Tagen) oder als Datum formatiert (Abb. 71). Wurde die Seite von dem Anwender schon mehrfach besucht, erscheint zudem die Anzahl der Aufrufe im Tooltip (Abb. 69 und Abb. 71). Bei externen Links wird zusätzlich angegeben, wie viele Seiten der Website der Benutzer bereits angesehen hat (Abb. 71). Dies dient als Indikator dafür, ob der Anwender mit der Website eines Anbieters schon vertraut ist.

Social-Navigation-Hinweise (s. Abschnitt 4.5.5.2) wurden in HyperScout I nicht berücksichtigt, sind aber von (Wollenweber 2004) in dem *CoInternet-Projekt* als Erweiterung von HyperScout I prototypisch implementiert und evaluiert worden (s. Abb. 87 und Abschnitt 8.7).



Abb. 87: Der *CoInternet-Client* nutzt HyperScout I, um Bewertungen anderer Benutzer einzublenden.

5.5.4.11 Der URI des Link-Ziels

HyperScout I stellt den URI des Link-Zieles im Tooltip statt in der Statuszeile des Browsers dar. Die Ausgabe erfolgt dabei in einer oder zwei Zeilen: Die erste Zeile enthält bei externen Links den **Domain-Namen** der Site. Dies soll das Erkennen von externen Links zu gefälschten Phishing-Sites erleichtern (siehe auch Lin, Greenberg et al. 2011). Darüber hinaus werden sowohl bei externen als auch internen Links **Pfad und Dateiname** in einer eigenen Zeile aufgeführt (s. Abb. 83 und Abb. 84). Durch diese Gestaltung sollten die in dem URI enthaltenen Informationen übersichtlicher und einfacher verständlich sein.

6 Evaluation und Optimierung der erweiterten Hyperlink-Benutzungsschnittstelle

Dieses Kapitel beschäftigt sich mit der Evaluation und Weiterentwicklung des im letzten Abschnitt vorgestellten Konzepts zur Erweiterung der Benutzungsschnittstelle von Hyperlinks in verteilten Informationssystemen. Das erste Teilkapitel 6.1 geht auf die empirische Evaluation des ersten Prototyps *HyperScout I* ein. Zuerst werden die Studienziele und die Hypothese der Untersuchung definiert, dann Methodik, Aufbau und Ablauf der Evaluation beschrieben. Es folgen die Ergebnisse der sieben Thinking-Aloud-Tests, die Verbesserungsvorschläge der Teilnehmer und eine Bilanz über die Eignung der vorgestellten Konzepte für die Vereinfachung der Navigation im Web (Abschnitt 6.2).

Um zusätzliche, *quantitativ* aussagekräftige Daten zur Bedeutung der im Web verfügbaren impliziten Link-Attribute für die Web-Navigation zu gewinnen, erfolgte als nächster Schritt eine *Umfrage* zu den Eigenschaften von Links und Link-Zielen im Web (siehe Abschnitt 6.3).

Basierend auf den Ergebnissen der Thinking-Aloud-Tests und der Umfrage wurde ein neues Konzept entwickelt und prototypisch realisiert (siehe Unterkapitel 6.4). Das Konzept *HyperScout II* verfügt neben den Tooltips über zwei innovative Darstellungstechniken für zusätzliche Link-Informationen: Die „*Link-Overlays-on-Demand*“ (s. Abschnitt 6.4.1) bieten einen Überblick über wesentliche Eigenschaften aller Link-Anker einer Webseite auf einmal, und die „*multiplen Maus-Icons*“ visualisieren symbolisch mehrere Link-Attribute neben dem Mauszeiger, sobald er sich über einem Link-Anker befindet (Abschnitt 6.4.2). Da die Icons in den Tooltips des ersten Prototyps in der Studie häufig missverstanden wurden und sie in *HyperScout II* als multiple Maus-Icons eine wesentlich größere Bedeutung erhalten sollten, wurde ein neuer Satz von Icons mithilfe eines partizipativen Verfahrens entwickelt (s. Abschnitt 6.3.3). Abschnitt 6.4.3 geht auf die Gestaltung und Inhalte der überarbeiteten Tooltips von *HyperScout II* ein.

Das zweite HyperScout-Konzept wurde mit derselben Methode wie das erste evaluiert (s. Unterkapitel 6.5). An der zweiten Studie nahmen 13 Personen teil, da noch mehr auf Details bei der Benutzung eingegangen werden sollte und das neue Konzept wesentlich mehr Eigenschaften bot, die in der Studie zu berücksichtigen waren.

6.1 Empirische Evaluation von HyperScout I

Das im Kapitel 5.5ff vorgestellte Konzept *HyperScout I* verwendet *strukturierte, mehrzeilige Tooltips* zur Darstellung zusätzlicher Link-Informationen in Webbrowsern. Dieses Teilkapitel befasst sich mit der Evaluation dieses Konzepts und ist wie folgt aufgebaut: Abschnitt 6.1.1 beschreibt die Ziele und Hypothese der Studie. Danach wird auf die Methodik der Evaluation eingegangen (Abschnitt 6.1.2), und Abschnitt 6.1.3 behandelt die verwendete Testmethode „*Thinking Aloud*“.

Teil 6.1.4 stellt Aufbau und Ablauf der Evaluation von *HyperScout I* vor. Zur weiteren Optimierung des Evaluationsverfahrens wurden zwei Pilot-Studien durchgeführt, die Abschnitt 6.1.5 zusammenfasst. Abschnitt 6.2 präsentiert die Ergebnisse der Studie.

6.1.1 Studienziele und Studienhypothese

Die entscheidende Idee von HyperScout besteht darin, den Anwendern in Tooltips relevante und implizit vorhandene – normalerweise jedoch unzugängliche – Zusatzinformationen zu allen Links anzubieten, um die Bedeutung eines Links und Eigenschaften von Link und Zielobjekt zu vermitteln. Dies soll insbesondere die Navigationsentscheidungen des Benutzers vereinfachen und unnötige Navigationsschritte vermeiden.

Die Konzepte von *HyperScout I* basieren auf Forschungsergebnissen aus dem Bereich (verteilter) Hypertext-Informationssysteme und sind auf den Umgang mit Informationen in solchen Systemen ausgerichtet. Die Studie von *HyperScout I* konzentriert sich dementsprechend auf die Informationsrecherche. Primäre Zielgruppe von *HyperScout I* sind Personen, die im Web nach Informationen suchen und wissensbezogene Aufgaben bearbeiten. Douglas Engelbart hat für diese Personengruppe in den sechziger Jahren des vorherigen Jahrhunderts den Begriff „*Knowledge Worker*“ geprägt (Bardini 2000: 146ff). Studien haben für diesen Einsatzbereich als wesentliches Benutzbarkeitskriterium die *Effizienz bei der Lösung von Aufgaben* ermittelt (Smith, Newman et al. 1997). Für die zielgerichtete Informationsrecherche sind unter anderem die *benötigte Zeit* und *die „Optimalität“ des Navigationspfads* wichtige quantitative Kennzeichen (Smith, Newman et al. 1997). Übertragen auf HyperScout bedeutet dies, dass die Tooltips den Anwendern helfen sollen, Links *besser zu verstehen* und diese präziser und häufiger korrekt auszuwählen.

Im Rahmen der Studie sollten vor allem *qualitative* Ergebnisse erzielt werden. Als Hypothese wird davon ausgegangen, dass *HyperScout I* geeignet ist, dem Benutzer fehlende Informationen bei der Link-Navigation zu bieten, die dazu beitragen, die in den Kapiteln 3.2 bis 3.4 aufgeführten Benutzbarkeitsdefizite von verteiltem Hypertext zu reduzieren, wie den Cognitive Overhead, die mangelnde Link-Konsistenz oder die oft arbeitshinderliche Performanz.

In Bezug auf das HyperScout-System sollen die Studie folgende Fragen beantworten:

- Welchen Mehrwert bieten die einzelnen Informationen und helfen sie bei der Navigation?
- Gibt es Situationen und Aufgaben, bei denen die Konzepte von HyperScout I eine besonders wichtige Unterstützung für die Navigation im Web darstellen und solche, in denen sie ungeeignet sind? Welche Gründe lassen sich hierfür identifizieren und welche Alternativen bieten sich?
- Sind *Tooltips* für die Darstellung zusätzlicher Link-Informationen geeignet oder treten im Umgang mit Hypertext und dem Web Probleme auf?

- Ist die *Präsentation* der Informationen in den Tooltips von *HyperScout I* angemessen und ermöglicht sie eine schnelle und eindeutige Wahrnehmung der Link-Attribute?
- Wie ist die Gebrauchstauglichkeit der Erweiterung allgemein zu bewerten und wie ist die Zufriedenheit der Teilnehmer im Umgang mit HyperScout I und dem Web?

Des Weiteren sollten Möglichkeiten zur Verbesserung der entwickelten Konzepte ermittelt werden. Dazu gehört, wie sich die Darstellung von Link-Tooltips optimieren lässt und welche Informationen ergänzt oder weggelassen werden sollten.

6.1.2 Methoden zur Evaluation der Benutzbarkeit von Software

Zur Benutzbarkeits-Evaluation von Software-Systemen wurden zahlreiche Methoden entwickelt, die sich für unterschiedliche Personengruppen und Studienziele eignen und somit spezifische Stärken und Schwächen aufweisen (Nielsen & Phillips 1993; Nielsen 1994c; Weinreich 1997: 80 ff; Hegner 2003). Für die Untersuchung von HyperScout I sollte eine Methode verwendet werden, mit der die im vorherigen Kapitel definierten Ziele mit vertretbarem Aufwand zu erreichen sind. Im Folgenden werden wesentliche Kategorien von Evaluationstechniken vorgestellt und ihre jeweiligen Potenziale und Grenzen verglichen.

Heuristische Methoden

Eine Kategorie von Evaluationstechniken sind *heuristische Methoden,* bei denen mehrere Personen einzeln ein System nach potenziellen Benutzbarkeitsproblemen analysieren und anschließend die aufgedeckten Probleme zusammengefasst werden. Eine häufig eingesetzte Variante dieser Technik sind *Experten-Reviews*: Dabei analysieren Ergonomie-Experten ein System anhand ihrer Erfahrung, auf der Basis von Gestaltungsrichtlinien und/oder Kriterien-Katalogen für Software-Ergonomie. Diese Methoden gelten als relativ schnell und preiswert in der Durchführung, setzen aber die Verfügbarkeit und das Budget für entsprechende Experten voraus. Sie bergen das Risiko, dass bei der Untersuchung der Aufgabenkontext ungenügend berücksichtigt wird und sogar kritische Praxisprobleme unerkannt bleiben (Molich & Nielsen 1990; Nielsen & Molich 1990; Nielsen & Phillips 1993; Kieras 1994; Gerhardt-Powals 1996; Nielsen 2005).

Evaluation mit Walkthroughs

Walkthrough-Methoden basieren auf *kognitiven Theorien* über die Benutzer. Zur Betrachtung kommen Lernstrategien, die mentalen Modelle der Benutzer und wie sie aufgrund ihrer Wahrnehmung eines Systems Teilziele generieren. Auf einer solchen theoretischen Basis wird beim *Cognitive Walkthrough* das Verhalten von Benutzern simuliert: Der Evaluator bearbeitet einzelne Aufgaben Schritt für Schritt mit dem System und analysiert dabei jeweils, welche Möglichkeiten das System dem Anwender bietet, wie er sie voraussichtlich wahrnimmt und was er wahrscheinlich als Nächstes tun würde. Diese Simulation wird mit den vorgesehenen Aufgaben und Workflows verglichen, um zu ermitteln, ob der Benutzer in der

Lage ist, alle notwendigen Aktionen auszuführen und welcher (auch mentale) Aufwand mit ihnen verbunden ist. Walkthroughs müssen nicht unbedingt von Experten für Software-Ergonomie durchgeführt werden, sondern sind auch für Software-Entwickler geeignet. Diese Verfahren neigen aber dazu, dass gravierende Probleme übersehen werden, und zeigen Schwächen bei der Bewertung der Praxisrelevanz der Benutzbarkeitsdefizite (Polson, Lewis et al. 1992; Wharton, Rieman et al. 1994; Blackmon, Polson et al. 2002).

Modellbasierte Studienmethoden

Modellbasierte, analytische Methoden ermitteln die Effizienz der Anwender bei der Bearbeitung einzelner Aufgaben mit dem System, indem die Aufgaben als elementare Aktionen modelliert werden. Für jede dieser Aktionen, wie Mausbewegungen, Mausklicks und eine Texteingabe, wird der jeweilige (Zeit-)Aufwand analysiert. Populäre Beispiele dieser Kategorie sind *GOMS* (ein Akronym für „*Goals, Operators, Methods and Selection Rules*“, siehe: Card, Newell et al. 1983) und die *TAG*-Modellierung („*Task-Action-Grammar*“, siehe: Payne & Green 1986). Es gibt einige Erweiterungen dieser Techniken, die unter anderem auch kognitive Prozesse, wie die Zeit zum Erlernen eines Systems, berücksichtigen (z. B. *NGOMSL*, siehe: John & Kieras 1996). Methoden dieser Kategorie eignen sich beispielsweise, um die Leistung von Experten beim Umgang mit Software zu optimieren sowie unterschiedliche Systemvarianten miteinander zu vergleichen. Aufgrund der vereinfachten Modellierung von Benutzungsabläufen sind diese Methoden in ihren Möglichkeiten beschränkt und beispielsweise weniger geeignet, die kognitiven Vorgänge der Benutzer beim Umgang mit einem System zu verstehen, potenzielle Verständnisprobleme zu erkennen und Mängel in der Unterstützung von komplexen Arbeitsabläufen zu ermitteln (Payne & Green 1986; Nielsen & Phillips 1993).

Empirische Verfahren

Von großer praktischer Bedeutung bei der Software-Evaluation sind *empirische Benutzbarkeitstests*, bei denen die Anwender eine zentrale Rolle spielen. Hierzu gehören *Befragungen* der Benutzer in *Interviews* oder mit *Fragebögen* (Root & Draper 1983), *Beobachtungen* von Benutzern sowie *Analysen von Protokollen* bei der Anwendung eines Systems (Eberleh, Oberquelle et al. 1994; Weinreich 1997; Hegner 2003).

Eine häufig eingesetzte beobachtende Technik ist „Thinking Aloud“ (das „Laute Denken“). Dabei bearbeiten typische Nutzer möglichst realistische Aufgaben mit einem System und werden gebeten, ihre Gedanken währenddessen zu verbalisieren (s. Abschnitt 6.1.3).

Zahlreiche Studien haben die Stärken unterschiedlicher Evaluationstechniken miteinander verglichen. Häufig wird die hohe *Qualität* der Ergebnisse von empirischen beobachtenden Testmethoden wie „Thinking Aloud“ hervorgehoben, da man viel über das Vorgehen der Benutzer, ihre kognitiven Probleme beim Erlernen eines Systems und ihre Einstellung gegenüber der Software erfährt (Dix, Finlay et al. 1993: 385ff). Die Tests haben in der Regel eine

hohe Realitätsnähe, da charakteristische Aufgaben durch repräsentative Benutzer bearbeitet werden. Insbesondere schwerwiegende Probleme werden zuverlässiger erkannt als bei anderen Methoden, wogegen kleinere Probleme oft unbemerkt bleiben (Jeffries, Miller et al. 1991; Karat, Campbell et al. 1992). Zudem lässt sich gut die Bedeutung einzelner Probleme bewerten, und man erhält Hinweise auf alternative Lösungen, die zur Optimierung der Benutzungsschnittstelle beitragen können (Nielsen & Phillips 1993).

Im Rahmen dieser Arbeit waren insbesondere *qualitative* Ergebnisse für den Einsatz von HyperScout I bei der täglichen Informationsrecherche im Web von Interesse (s. Abschnitt 6.1.1). Da keine Experten zur Evaluation zur Verfügung standen und bei den Tests auch neue Verbesserungs- und Erweiterungsmöglichkeiten ermittelt werden sollten, wurde eine empirische Evaluationsmethode gewählt: Die hohe Qualität der Ergebnisse gab den Ausschlag für den Einsatz des *Lauten Denkens* (*Thinking Aloud*).

6.1.3 Thinking Aloud Studie: Methodik und Teilnehmerzahl

„*Thinking Aloud*“, das *laute Denken*, ist ein empirisches Verfahren, das in Wissenschaft und Praxis oft zur Software-Evaluation eingesetzt wird. Dabei lässt man repräsentative Benutzer realistische Aufgaben mit einem System bearbeiten und bittet sie, kontinuierlich ihre Gedanken bei der Arbeit laut wiederzugeben. Die Verbalisierung der Gedanken ermöglicht einen Einblick in das mentale Systemmodell der Anwender und hilft bei der Aufdeckung von Missverständnissen und Problemen. In Verbindung mit dem beobachteten Verhalten können die Defizite einer Benutzungsschnittstelle und ihre Ursachen präzise bestimmt werden (Dumais & Redish 1993).

Die Entwicklung der Thinking-Aloud-Technik wird dem Psychologen *Karl Duncker* zugeschrieben, der sich mit den Lösungsprozessen zum Erreichen von Zielen in bestimmten Problemsituationen beschäftigte. In seinem Buch von 1935 schlägt er ein Versuchsverfahren vor, bei dem Menschen zum lauten Denken aufgefordert werden, um ihre mentalen Modelle, Intuitionen und Schlussfolgerungen zu ermitteln (Duncker 1935; Duncker 1945). Für die Evaluation von Software wurde Thinking Alound erstmals in (Lewis 1982) beschrieben. (Jørgensen 1989) geht später genauer auf die Methode ein und berichtet erstmals von Erfahrungen beim erfolgreichen Einsatz von Thinking Aloud.

Inzwischen gibt es mehrere Varianten der Thinking-Aloud-Methode, wie das „*Co-Discovery Learning*“: Benutzer bearbeiten dabei paarweise Aufgaben mit einem System und schildern sich gegenseitig ihre Gedanken (Buur & Bagger 1999). Beim „*Question Asking*“ stellen Benutzer den Testleitern bei der Anwendung aufkommende Fragen zum System und werden dann von ihnen „gecoacht“ (Mack & Robinson 1992). Janni Nielsen et al. geben einen Überblick zu weiteren Varianten von Thinking Aloud (Nielsen, Clemmensen et al. 2002).

Für die Test-Durchführung wird eine Art „Labor“ benötigt. Es erlaubt den Teilnehmern, ungestört mit dem zu untersuchenden System zu arbeiten und den Testleitern, sie dabei zu beobachten und den Testablauf aufzuzeichnen.

Zur Vorbereitung der Studie werden Szenarios mit repräsentativen Aufgaben erstellt. Die Szenarios sollten in Pilotstudien überprüft werden, um sicher zu stellen, dass die gestellten Aufgaben lösbar sind und alle relevanten Aspekte des Systems erfassen (Nielsen 1988a). Aus diesem Grunde wurden auch bei der Evaluation von HyperScout I zwei Pilotstudien durchgeführt, die wesentlich zur Optimierung des Haupttests beitrugen (s. Abschnitt 6.1.5).

Zu Beginn eines Tests werden die Teilnehmer mit ihrer Rolle als *Tester* des Systems vertraut gemacht. Sie erhalten eine kurze Einführung in das Testverfahren und das zu untersuchende System. Die Benutzer sollen die ihnen gestellten Aufgaben mit dem System bearbeiten und dabei alles, was sie gerade in dem Zusammenhang denken, laut wiedergeben. Wenn sie etwas unerwartet, missverständlich oder störend finden – sei es aus ihrer Sicht auch irrelevant oder ein eigener „Bedienungsfehler" –, so sollen sie es äußern.

Die Rolle der Evaluationsleiter ist nicht nur die Beobachtung der Teilnehmer, sondern ebenfalls die als Motivator, das laute Denken nicht zu unterbrechen. Sollte ein Anwender das laute Denken einstellen, so ist er mit Fragen zur Äußerung seiner Gedanken zu animieren. Die Fragen sind in einer Form zu stellen, die den Benutzer im sonstigen Verhalten kaum beeinflusst.[156]

Zur Aufzeichnung des Testablaufes stehen unterschiedliche Möglichkeiten zur Verfügung. Angefangen mit schriftlichen Beobachtungsnotizen des Testleiters, über Tonbandaufnahmen bis hin zu Videoaufzeichnungen und einer technischen Protokollierung aller Benutzerein- und Systemausgaben.

Nach dem Test erfolgt eine Nachbesprechung („Debriefing"), in der die Benutzer sich zusätzlich zum Testverlauf und zum System äußern können. Die Testleiter sollten nun ebenfalls noch offene Fragen zum Benutzerverhalten klären.

Die Auswertung der gewonnenen Daten ist eine nicht zu unterschätzende Herausforderung und entscheidend für die Qualität der Ergebnisse einer jeden empirischen Studie (Bortz & Döring 1995). Dies gilt auch für Thinking-Aloud-Tests. Ericsson und Simon schlagen ein dreistufiges Vorgehen für die Analyse der Studienergebnisse vor (Ericsson & Simon 1993), das auch im Rahmen dieser Arbeit eingesetzt wurde:

- Als erster Schritt werden die Aktionen und Äußerungen der Teilnehmer aus den Aufzeichnungen und dem Protokoll zusammengeführt, aufgelistet und in ein weiter zu verarbeitendes Textformat transkribiert.
- Zweitens werden diese Daten *interpretiert* und *segmentiert*. Die dabei entstehenden Abschnitte sollen jeweils einen (potenziell problematischen) Benutzungsvorgang repräsentieren.

[156] Die Fragen sollten nicht suggestiv sein. Geeignete Fragen sind beispielsweise: „Was denkst du gerade?", „Was sagt dir das?" und „Was siehst du da?"

- Drittens sind die Daten zu *codieren*, das heißt, für jeden Abschnitt werden die potenziellen Probleme identifiziert, gruppiert und bestimmten Programmeigenschaften zugeordnet. Ein entsprechendes Schema zur Gruppierung der Probleme basiert auf einer vorhergehenden Aufgabenanalyse und einer Voruntersuchung des Datenmaterials.

Die laut geäußerten Gedanken der Benutzer sind bei der Interpretation und Codierung der Daten sehr förderlich, da sie Schwierigkeiten bei der Systembenutzung aufdecken und helfen, das mentale Systemmodell der Anwender zu verstehen.

Trotz aller Stärken hat Thinking Aloud zwei potenzielle Nachteile: Zum einen können die Anwesenheit des Studienleiters, die Testsituation und das laute Denken zu einem atypischen Benutzerverhalten führen (Ericsson & Simon 1993). Manche Menschen fühlen sich aufgrund der Situation unwohl, andere haben Probleme bei der Artikulation ihrer Gedanken (Nielsen & Phillips 1993) oder fühlen sich durch das laute Denken kognitiv belastet (Nielsen, Clemmensen et al. 2002). Im Gegensatz dazu sind einige Personen kurzfristig leistungsfähiger als gewöhnlich, wenn sie sich beobachtet wissen („*Hawthorne Effekt*“, siehe: Landsberger 1958). Dies kann dazu führen, dass sie konzentrierter arbeiten und Probleme „herunterspielen“, um einen kompetenteren Eindruck zu machen (Draper 2001). Es muss daher vermieden werden, dass die Teilnehmer fälschlicherweise glauben, sie sollen bestimmten Erwartungen genügen, Leistungsvorgaben erreichen oder gar selbst getestet werden.

Zweitens dauert die Auswertung der Daten relativ lange. Es kann daher gewöhnlich nur eine kleine Anzahl von Tests durchgeführt werden. Als Folge erhält man kaum quantitativ valide Ergebnisse (Nielsen & Phillips 1993). Eine ganze Reihe von Untersuchungen beschäftigen sich damit, wie viele Personen an einem Thinking-Aloud-Test teilnehmen müssen, um alle relevanten Probleme zu finden. Jakob Nielsen hat bereits vor über 10 Jahren den Begriff „Discount Usability Engineering“ geprägt und plädiert für vereinfachte („*simplified*“) Thinking-Aloud-Tests[157] mit kleinen Teilnehmerzahlen (Nielsen 1994b). Nach seinen Analysen zahlreicher Thinking-Aloud-Studien werden bereits mit 3 Teilnehmern in der Regel über die Hälfte der Benutzbarkeitsdefizite gefunden (Nielsen 1988a; Nielsen & Landauer 1993), und er empfiehlt eine Anzahl von 5 Teilnehmern, da so mit großer Wahrscheinlichkeit über drei Viertel der Probleme identifiziert werden. Weitere Benutzer tragen nur verhältnismäßig wenig zum Erkenntnisgewinn bei (Nielsen 1993b: 169), zumal die Menge zusätzlich gefundener Fehler negativ exponentiell mit der Teilnehmerzahl abnimmt (Nielsen 1994a). Seiner Ansicht nach sollten daher besser die bis dato entdeckten Schwächen beseitigt und weitere Testzyklen angeschlossen werden, um zu überprüfen, ob sie tatsächlich behoben wurden und sich nicht neue Probleme eingeschlichen haben.

[157] Im Wesentlichen besteht die Vereinfachung darin, dass die Versuche nicht per Video aufgenommen werden, sondern man direkt das beobachtete Verhalten protokolliert. Dies beschleunigt nach Nielsens Erfahrung die Auswertung erheblich (Nielsen 1994b).

Diese absoluten Teilnehmerzahlen wurden allerdings von anderen Forschern relativiert, die die Ergebnisse von Thinking-Aloud-Studien mit komplexen Systemen analysierten. In Studien von Spool et al. entdeckten sogar bei jeweils 18 Teilnehmern auch die letzten Benutzer noch neue gravierende Benutzbarkeitsmängel. Es wurde allerdings ein komplexes E-Commerce-System untersucht und jeder Anwender bekam nur Teile des Systems zu sehen (Spool & Schroeder 2001). Dieses Ergebnis steht daher nicht unbedingt im Widerspruch zu Nielsens These, dass mehrere Testzyklen mit kleineren Teilnehmerzahlen die Effizienz von Thinking Aloud erhöhten, sondern stellt heraus, dass die Komplexität des Systems bei der Festlegung der Studiengröße zu beachten ist.

Die optimale Studiengröße für einen Thinking-Aloud-Test ist somit von vielen Faktoren abhängig. Dies sind zum einen Art und Umfang des Systems, aber auch die Diversität der Benutzer und die Qualität und Länge der einzelnen Tests (vergl. auch Bevan, Barnum et al. 2003). Bei der Gestaltung der Studie ist insbesondere darauf zu achten, dass alle relevanten Teile des Systems evaluiert werden.

6.1.4 Aufbau und Ablauf der Evaluation von HyperScout I

Die erste Version des HyperScout-Systems wurde im Winter 2001/2002 mit sieben Teilnehmern evaluiert, die an einem Thinking-Aloud-Test teilnahmen. Diese Anzahl von Personen wurde als ausreichend angesehen, da in den Studienszenarios jeder Teilnehmer alle Möglichkeiten (in diesem Falle Kategorien von Link-Informationen) von HyperScout I benutzen sollte. Zudem zeichnete sich bereits nach fünf Tests ab, dass weitere Teilnehmer kaum neue Erkenntnisse erbrachten. Dies bestätigte sich in den letzten beiden Versuchen.

Die Tests dauerten jeweils zwischen 1h 40min und 2h 20min. Alle sieben Teilnehmer schätzten sich selbst als „erfahren“ im Umgang mit dem Web ein und gaben an, sich regelmäßig mit der Recherche von Informationen im Web zu befassen. Sie nutzten es seit mindestens drei Jahren, wenigstens einmal pro Woche und waren mit dem verwendeten Browser (*Microsoft Internet Explorer 6*) vertraut. Das durchschnittliche Alter der Teilnehmer betrug 28,7 Jahre, zwei waren weiblich und fünf männlich. Fünf Teilnehmer kamen aus dem IT-Bereich.

Die hohe Kompetenz und Vertrautheit aller Teilnehmer mit dem Web war beabsichtigt, da die Hoffnung bestand, dass sie, basierend auf ihren Erfahrungen, selbst Anregungen zur Optimierung geben würden. Dieses Vorgehen hat sich bewährt, da alle Teilnehmer tatsächlich Ideen zur Weiterentwicklung der Konzepte beisteuerten. Um die Probleme von Benutzern mit geringerer Web-Erfahrung zu berücksichtigen, wurden bei der Evaluation von HyperScout II (s. Abschnitt 6.5) auch weniger versierte Benutzer einbezogen.

Für die Tests wurde ein zeitgemäßer PC des Fachbereiches Informatik unter Windows 2000 genutzt, der mit dem Deutschen Wissenschaftsnetz X-Win verbunden war und somit über eine sehr gute Netzwerkanbindung verfügte. An das Gerät waren zwei 17-Zoll-Bildschirme mit einer Auflösung von jeweils 1024×768 Pixel angeschlossen.

Vor dem Test wurden den Teilnehmer kurz die Möglichkeiten von HyperScout I vorgestellt. Anhand einer Übungsaufgabe konnten sie erste Eindrücke von der Erweiterung gewinnen. Danach wurden ihnen in 4 Szenarios insgesamt 14 Aufgaben gestellt, bei denen es darum ging, bestimmte Informationen von vorgegebenen Startseiten aus zu finden.

Ein Prototyp des *Scone UserTestTools* (s. Abschnitt 8.6.5.7) diente zur Unterstützung der Studie: Den Teilnehmern wurden mit seiner Hilfe die Aufgaben auf dem linken Bildschirm dargestellt. Nach dem Drücken der „Start"-Taste im *UserTestTool* erschien auf dem rechten Bildschirm der Webbrowser mit der Ausgangsseite für die Bearbeitung der Aufgabe (s. Abb. 88).



Abb. 88: Illustration des Studienaufbaus mit zwei Bildschirmen für die Evaluation der Konzepte von HyperScout I.

Der Testleiter protokollierte das Verhalten und die Äußerungen der Teilnehmer während der Studie. Die Sitzungen wurden akustisch aufgezeichnet. Das Scone-Framework (s. Kapitel 8.5ff) protokollierte automatisch die besuchten Seiten und die dargestellten Tooltips (vergl. Abschnitt 8.6.4).

Nach dem Test folgten eine Nachbesprechung und ein semi-strukturiertes Interview mit den Teilnehmern. Sie wurden zu folgenden Punkten befragt:

- ihre allgemeine Einschätzung des Systems,
- Probleme bei der Interaktion mit den Tooltips,
- die Angemessenheit der Informations-Präsentation,
- die Nützlichkeit der einzelnen Link-Attribute für die Navigation (anhand von Beispielen aus dem Test),
- allgemeines Feedback zu HyperScout I und zur Zufriedenheit mit der Erweiterung.

Die Analyse der Daten erfolgte in drei Schritten gemäß dem in Abschnitt 6.1.3 beschriebenen Verfahren: Als Erstes wurden die Ergebnisse des Protokolls und der Aufzeichnungen in einem Text-Dokument zusammengeführt. Danach erfolgte die Interpretation und Segmentierung in Abschnitte, die einzelne Benutzervorgänge repräsentierten. Diese wurden drittens nach Benutzbarkeitsproblemen und Themenbereichen gegliedert.

6.1.5 Ergebnisse der Pilottests: besondere Herausforderungen bei der Evaluation

Zwei einstündige Pilottests gingen der Hauptstudie von HyperScout I voraus. Der Studienaufbau dieser Pretests entsprach dem der Hauptstudie. Die Vorstudien dienten zur Ermittlung von Problemen bei der Studiendurchführung und zur Optimierung des Ablaufes. Es zeigte sich, dass folgendes für die Hauptstudie zu beachten war:

- Die Teilnehmer der Pretests beschäftigten sich mit einzelnen Aufgaben weitaus länger als vorgesehen, da sie wiederholt relevante Links übersahen, sogar wenn sie diese – nach ihrer eigenen retrospektiven Einschätzung – hätten wahrnehmen müssen. Sie gaben als Ursache an, dass sie nicht alle Informationen auf den Seiten gelesen hätten. Um die Bearbeitungszeit der Aufgaben zu beschränken, erhielten die Teilnehmer der Hauptstudie vom Testleiter unterstützende Hinweise zur Bearbeitung, wenn sie offenbar die Orientierung oder ihre Aufgabe aus den Augen verloren hatten; sie wurden dann beispielsweise zu einer Webseite zurückgeführt, von der aus die Aufgabe lösbar war.
- In der Vorstudie zeigten die Benutzer zumeist ein Verhalten, bei dem die HyperScout-Tooltips *nicht* erschienen und sie somit auch nicht evaluiert wurden: Nach dem Betrachten einer Webseite entschieden sie sich regelmäßig *zuerst* für einen Link und bewegten *anschließend* den Mauszeiger zum Link-Anker, um ihn *sofort* anzuklicken. Da die Tooltips erst mit einer Verzögerung von 0,8 Sekunden erschienen (s. Abschnitt 5.5.2), wurden sie nur in Ausnahmefällen angezeigt, wenn z. B. die Nutzer bei der Navigation zögerten.[158] Die Teilnehmer gaben an, dass dies ein für sie gewohntes Verhalten bei der Web-Navigation sei.

Um die Tooltips sinnvoll und effizient evaluieren zu können, bekamen die Teilnehmer den *expliziten Auftrag*, während der Studie möglichst oft auf die Tooltips zu achten, insbesondere wenn sie sich bezüglich ihres nächsten Navigationsschritts unsicher waren. Zudem wurden sie während des Tests an die Verfügbarkeit der Tooltips erinnert, sofern sie offensichtliche Probleme bei der Link-Navigation hatten und dennoch ihrem gewohnheitsgemäßem Verhalten folgten und die Tooltips nicht berücksichtigten.

Dieses Vorgehen nimmt bewusst in Kauf, dass die Benutzer nicht genauso wie mit ihrem „normalen" Webbrowser interagieren. Das beobachtete Verhalten kann somit nicht direkt mit dem von Anwendern verglichen werden, die ohne die HyperScout-Erweiterung auf das Web zugreifen. Da beobachtende Studien aber sowieso eine gewisse Änderung im Benutzerverhalten zur Folge haben (s. Abschnitt 6.1.3), primär qualitative Ergebnisse erzielt werden sollten und ein Vergleich von Link-Klicks und Bearbeitungszeiten mit einer Kontrollgruppe nicht geplant war, schien ein derartiger Arbeitsauftrag für die Teilnehmer akzeptabel, zumal die gewählte Methode die Effizienz bei der Evaluation von HyperScout I wesentlich

[158] Gleichzeitig wies dies bereits auf das grundsätzliche Problem von Tooltips für Hyperlinks im Web hin, nämlich dass Benutzer sich oft erst entscheiden, bevor sie den Mauszeiger zum Link-Anker bewegen (s. Abschnitt 6.2.1).

verbesserte. Wie geplant konnten bereits mit *sieben Tests* umfangreiche und eindeutige qualitative Evaluationsdaten zum HyperScout-Konzept gewonnen werden, sodass eine Auswertung sinnvoll erschien.

Die folgenden Seiten fassen die wichtigsten Ergebnisse dieser Tests und den dabei erfassten 16 Stunden Audio-Aufzeichnungen und über 70 Seiten handschriftlicher Notizen zusammen.

6.2 Ergebnisse der Evaluation von HyperScout I

In den sieben Thinking-Aloud-Tests konnten zahlreiche Ergebnisse zum Umgang mit den Tooltips von HyperScout I für Links gewonnen werden. Sie wurden drei thematischen Schwerpunkten zugeordnet:

1. Probleme bei der *Interaktion* mit den Tooltips (Abschnitt 6.2.1),
2. Herausforderungen bei der *Präsentation* der Informationen in den Tooltips (Abschnitt 6.2.2) und
3. spezifische Stärken und Schwächen der *Tooltip-Inhalte* (Abschnitt 6.2.3).

Bei der folgenden Zusammenfassung wird nur auf Benutzbarkeitsdefizite eingegangen, die auf *konzeptionelle* Schwächen von HyperScout I zurückzuführen sind, nicht aber auf solche, die auf technischen Unzulänglichkeiten des Prototyps beruhen.

6.2.1 Probleme bei der Interaktion mit Tooltips und Links

Es zeigte sich, dass die Darstellung von Link-Informationen in Tooltips eine grundsätzliche Schwäche aufwies: Trotz der Bitte, möglichst häufig auf die Tooltips zu achten, klickten alle Teilnehmer regelmäßig Links an, *ohne* auf das Erscheinen des Tooltips zu warten. Folgende Gründe gaben sie im Interview für ihr Verhalten an:

- Tooltips für Links waren für sie ungewohnt. Da ihres Erachtens normalerweise keine Tooltips im Webbrowser angeboten werden, hatten sie oft „vergessen“, dass zusätzliche Informationen verfügbar sind. Mehrere Teilnehmer schlugen vor, im Browser einen Hinweis neben dem Link-Anker oder beim Mauszeiger dazustellen, wenn besonders relevante Informationen im Tooltip zu finden sind.
- Die Teilnehmer waren sich bei vielen Links „sicher“, welche Bedeutung sie haben und dass sie zur gewünschten Seite führen. In solchen Fällen wurden Tooltips mit zusätzlichen Hinweisen als unnötig angesehen.
- Der verwendete PC stellte die meisten Webseiten aufgrund der guten Internetanbindung in Sekundenschnelle dar. Die Benutzer erachteten es daher oft als einfacher und schneller, einem Link zu folgen und das Zielobjekt durchzusehen, als auf den Tooltip zu warten,

zumal „bis zum Aufpoppen [des Tooltips] ebenfalls eine gewisse Zeit vergeht“, wie es ein Teilnehmer formulierte.[159]

Fünf der sieben Teilnehmer bewerteten die Zeit bis zum Erscheinen der Tooltips (0,8s, orientiert an den gebräuchlichen Tooltips in Windows XP) als „genau richtig“, zwei empfanden die Verzögerung als zu lang. Allerdings erkannten auch sie, dass eine kürzere Verzögerung zum häufigeren ungewollten Erscheinen von Tooltips führen könnte. Es wurde in diesem Zusammenhang vorgeschlagen, die Tooltips an- und abschaltbar zu machen oder einen zuschaltbaren „HyperScout-Modus“ einzuführen, bei dem Tooltips sofort erscheinen. Dies könnte jeweils über eine Schaltfläche im Browser oder eine spezielle Taste geschehen.

Die Studienteilnehmer haben darüber hinaus oft nicht alle Informationen in den Tooltips berücksichtigt, auch wenn sie sich sie ansahen und ihnen dort entscheidende Hinweise für die Navigation angezeigt wurden. Beispielsweise wurden manchmal externe Links ausgewählt, obwohl sie im Tooltip kenntlich waren und die Aufgabe innerhalb der vorgegebenen Website gelöst werden sollte. Als wesentliche Ursache gaben die Teilnehmer an, dass sie die Tooltips häufig lediglich überflogen hatten oder nur teilweise lasen. Es gab drei Vorschläge zur Reduzierung dieser Problematik:

- Die Informationen sollten auf das Wesentliche beschränkt werden,
- wichtige Daten müssten besser hervorgehoben und
- die Übersichtlichkeit erhöht werden.

Weiterhin wurde mehrfach angemerkt, dass viele der HyperScout-Tooltips zu groß seien und zu viel vom Dokument verdeckten. Dies spricht ebenfalls für eine Optimierung der Umfang und der Präsentation der Inhalte.

Wenn sich ein Hyperlink am unteren Bildschirmrand befand, wurde der Tooltip aus Platzgründen links oder rechts *neben* dem Link-Anker statt unterhalb eingeblendet. Dies stieß auf Teilnehmerkritik, weil der Tooltip dabei Inhalte verdeckte, die sie für das Verständnis des Links als unverzichtbar ansahen. Bei Hypertext-Systemen sollten folglich solche eingeblendeten Objekte die Zeile nicht überdecken, in der sich der gerade beachtete Link-Anker befindet.

Insgesamt scheinen Tooltips für zusätzliche Link-Informationen nur eingeschränkt geeignet zu sein. Sie wurden häufig *nicht* in Betracht gezogen,[160] selbst wenn sie wesentliche Informationen (wie eine Fehlermeldung zum Zielobjekt) enthielten.

---

[159] Zitate der Teilnehmer werden im Folgenden in Anführungszeichen hervorgehoben.

[160] Dies stellt gleichzeitig infrage, ob die momentan übliche Visualisierung des „`title`“-Attributes von Links im Web zweckmäßig ist, da gängige Browser hierfür ebenfalls Tooltips einsetzen (s. Abschnitte 5.1.3 und 5.3.7; Nielsen 1998a).

6.2.2 Herausforderungen bei der Informations-Präsentation in Tooltips

Alle Teilnehmer kritisierten, dass die Tooltips oft zu *viele* Informationen enthalten würden.[161] Daher dauere das Lesen zu lange und sie hätten sich „während der Zeit bereits die Seite ansehen“ können. Die Informationen müssten in *kompakterer* Form dargestellt werden, damit der Umgang mit den Tooltips effizienter wird und zu einer Zeitersparnis führt.

Wichtig war den Benutzern, dass *Redundanz* in den Tooltips vermieden wird. Diese trat beispielsweise auf, wenn Informationen im Tooltip bereits im Link-Anker zu finden waren oder wenn sich mehrere der Angaben im Tooltip überschnitten (z. B. Titel und Beschreibung der Seite).

Mehrere Teilnehmer fanden die variable Größe und das veränderliche Aussehen der Tooltips zu Anfang des Tests „ungewohnt“ und „verwunderlich“; am Ende des Tests wurde es aber von allen als eher vorteilhaft angesehen, da immer *nur die wichtigsten* Daten zu sehen sein sollten und die vom Tooltip *verdeckte Fläche zu minimieren* sei.

Ein weiterer Anwender regte an, die Tooltips durchscheinend zu gestalten, da sie dann nichts verdecken würden. Dies wurde allerdings bereits bei der ersten Proof-Of-Concept-Implementation von HyperScout erprobt (s. Abschnitt 5.5.2). Die Idee hat sich *nicht* bewährt, da die Lesbarkeit der kleinen Schrift in den Tooltips mit abnehmender Opazität sehr leidet: Entweder ist der Tooltip lesbar *oder* der Hintergrund *oder* keines von beiden.

Bezüglich der Detailliertheit einzelner Attribute gab es unterschiedliche Vorstellungen. Statt einfacher Angaben wie *Zugriff nicht gestattet* (s. Abb. 89), wünschten einige Teilnehmer eine genauere Angabe, *warum* der Zugriff nicht möglich sei. Wird beispielsweise ein Passwort benötigt, dann sollte bereits angegeben werden, ob der Benutzer über ein Zugangskonto verfügt. Bisweilen wurden auch mehr *technische Details* gefordert, z. B. der Fehlercode von nicht mehr zugreifbaren Dokumenten; andere Teilnehmer wollten hingegen bei Fehlern statt des Tooltips lediglich ein Fehlersymbol beim Link-Anker, das das Problem unmittelbar vermittelt.

[161] Die Darstellung *aller* verfügbaren und möglicherweise nützlichen Link-Informationen die in Kapitel 4.5 klassifiziert wurden, war eine bewusste Designentscheidung für den ersten HyperScout-Prototyp, da die Auswahl der Daten im Voraus schwierig erschien und alle Kategorien von Link-Hinweisen bei der Evaluation berücksichtigt werden sollten.



Abb. 89: Zugriff ist nur für Mitglieder des W3C gestattet und erfordert ein Passwort.

Insbesondere gab Anlass zur Kritik, dass die inhaltlichen Angaben zum Link-Ziel in den Tooltips „unvollständig“ seien: Da sie lediglich kurze inhaltliche Hinweise auf das Zielobjekt enthielten, könnten sich gerade benötigte Informationen „dennoch irgendwo auf der Zielseite befinden“ oder „von einem Link auf der Zielseite erreichbar sein“. In einigen Fällen folgten Benutzer daher einem Link, obwohl sie sich aufgrund der HyperScout-Hinweise „eigentlich auf dem Holzweg“ wähnten. Einige Teilnehmer sahen es somit als wünschenswert an, wenn ihnen das System auf Anfrage mehr Details zum Link-Ziel bereitstellen könnte oder ihre Aufgabe berücksichtige, damit „genau die benötigten Informationen angezeigt werden“.

Die *Qualität* der Angaben in den Tooltips wurde häufiger bemängelt. Hauptursache hierfür waren unpassende oder unvollständige Angaben in den Zielseiten der Links, beispielsweise im „`title`“-Element oder den Meta-Elementen. Einem der Testteilnehmer war diese Problematik von Suchmaschinen bekannt, da sie für die Ergebnislisten ebenfalls auf solche Daten zurückgreifen. Er wies darauf hin, dass sich in den Meta-Elementen oft „lediglich Buzzwords, sinnlose und manipulative Informationen“ befänden, „um in der Ergebnisliste einer Suchmaschine angeklickt zu werden“. Dieses Defizit hat auch negativen Einfluss auf die Nützlichkeit des HyperScout-Konzepts.

Es zeigte sich, dass insbesondere die Verständlichkeit der *Bezeichnungen der Link-Attribute* größere Herausforderungen stellte als erwartet. Häufig gar nicht verstanden wurden topologische Hinweise (s. Abschnitt 5.5.4.7), beispielsweise auf eine *Hub-*, *Index-* oder *Landmark-Seite* oder dass der Link einen *Querverweis* darstellt; den meisten Teilnehmern waren die entsprechenden Konzepte fremd. Andere Bezeichnungen wurden als missverständlich empfunden, wie *Zurück* für Links zu Seiten, die sie bereits besucht hatten. Sie assoziierten diesen Begriff mit einer Aktion (dem Back-Button) und keiner Eigenschaft. Verbesserungsvorschläge waren hierfür unter anderem „Bereits besucht am“ und „Wiederbesuch“. Ein weiteres Problem stellten Fachbegriffe und englischsprachige Angaben wie „*Zip File*“ dar; hier

wurden gängigere Titel wie „Komprimierte Datei“ oder „Download“ gewünscht. Diese Vorschläge wurden beim nächsten Prototypen HyperScout II berücksichtigt (s. Abschnitt 6.4ff).

Der Einsatz von *Farben* zur Strukturierung der Tooltips wurde allgemein gelobt, da dies die Übersichtlichkeit erhöhe. Die Farbzuordnung im ersten Prototyp beschrieben aber einige Teilnehmer als „inkonsistent“ und „nicht immer intuitiv“. Zum Beispiel seien Autorenangaben und der Hinweis auf „Formularfelder“ unterschiedliche Arten von Metadaten, die nicht mit derselben hellgrauen Hintergrundfarbe versehen werden dürften (vergl. Abschnitt 5.5.2).

Alle Anwender empfanden den Einsatz von *Icons* zur Kennzeichnung der Attribute in den Tooltips grundsätzlich als hilfreich. Allerdings hatten sie bei der Interpretation mehrerer Icons Probleme und bemängelten, dass sich einige „zu ähnlich“ sähen.[162]

Die Auswahl der dargestellten Attribute, ihre Bezeichnungen, Gruppierung und Reihenfolge sowie die Icons und Farben wurden aufgrund des Feedbacks für den nächsten Prototyp überarbeitet (s. Abschnitt 6.4.3).

6.2.3 Bedeutung der angebotenen Informationen für die Navigation im Web

Die Studienteilnehmer der waren aufgefordert, die in den Tooltips angebotenen Informationen zu kommentieren. Zusätzlich wurde im Interview (nach dem Test) auf die einzelnen Attribute eingegangen (s. Abschnitt 6.1.4). Ein Großteil der folgenden Teilnehmer-Einschätzungen stammen aus dieser Nachbesprechung.

Für einige der angebotenen Navigationshinweise war die Teilnehmereinschätzung recht einheitlich, bei vielen war das Bild aber eher heterogen. Die Bewertungen variierten sowohl intra- als auch interindividuell: Nicht nur die persönlichen Vorlieben scheinen bedeutsam zu sein, sondern auch die aktuelle Aufgabe der Person, die Verständlichkeit des Anker-Textes sowie die jeweils den Tooltips zugrundeliegenden Metadaten im Link-Anker und Link-Ziel.

Im Folgenden werden die geäußerten Gedanken der Benutzer und ihre Kommentare bezüglich der verschiedenen Link-Informationen im Web gemäß der Kategorisierung aus Kapitel 4.5 und 5.5.4 zusammengefasst.

6.2.3.1 Der semantische Typ des Links

Der von HyperScout dargestellte semantische Link-Typ (s. Abschnitt 5.5.4.1) führte zu unterschiedlichen Ergebnissen. Einfache Hinweise wie *Link zur Homepage* und *Verweis zur identischen Seite* wurden oft als hilfreich erachtet. Angaben wie *Übersicht* (für Verweise zu übergeordneten Seiten), *Detailpfad* (für Links, die in der Site-Hierarchie hinabführten) und *Querverweis* (assoziative Links) fanden hingegen bei keinem der Teilnehmer Anklang. Diese

[162] Zu einem ähnlichen Ergebnis kam auch Rebecca Witt, die eine frühe Version von HyperScout erweiterte und ebenfalls mit Benutzern evaluierte (Witt & Tyerman 2002).

Informationen wurden – selbst nach der Erläuterung durch den Testleiter – fast durchgängig als „unverständlich“ oder „überflüssig“ bezeichnet.

Einige Teilnehmer gaben an, dass sie stattdessen konkretere Hinweise zur Bedeutung des Zieldokuments bevorzugen würden; beispielsweise sollten bei einem Link-Anker, der als Link-Text den Namen einer Person erhält, Hinweise wie „Zur Kontaktseite der Person“ oder „Publikationsliste der Person“ erscheinen, sofern sich diese Information nicht bereits eindeutig aus dem Link-Kontext ergibt.

6.2.3.2 Die Verteilungscharakteristik des Links

Die Verteilungscharakteristik eines Links (*interne* vs. *externe* Links; s. Abschnitt 5.5.4.2) wurde von allen sieben Teilnehmern als „manchmal wichtig“ oder „oft wichtig“ für die Navigation angesehen, da externe Links – wie es eine Person ausdrückte – „zu Dokumenten mit anderer redaktioneller Zuständigkeit führen“. Die im Tooltip neben dem *Domain-Namen* angebotene *Beschreibung* der externen Site (extrahiert aus dem Seitentitel der Startseite der Site, s. Abb. 90) wurde unterschiedlich bewertet: Die Meinungen reichten von „sensationell gut“ bis zur kritischen Einschätzung, dass diese Information oft nicht „stichhaltig und vertrauenswürdig“ sei oder sie sich „zumeist bereits aus dem Domain-Namen“ ergebe. Diese Divergenz spiegelt gleichzeitig die variierende Qualität des Hinweises im Tooltip wider. Gegebenenfalls sollte daher auch auf weitere Quellen zur Bestimmung des Anbieters zurückgegriffen werden, beispielsweise den Daten, die der DNS-Registrar zum Inhaber der Domain bereitstellt (vergl. Abschnitt 3.3.3).

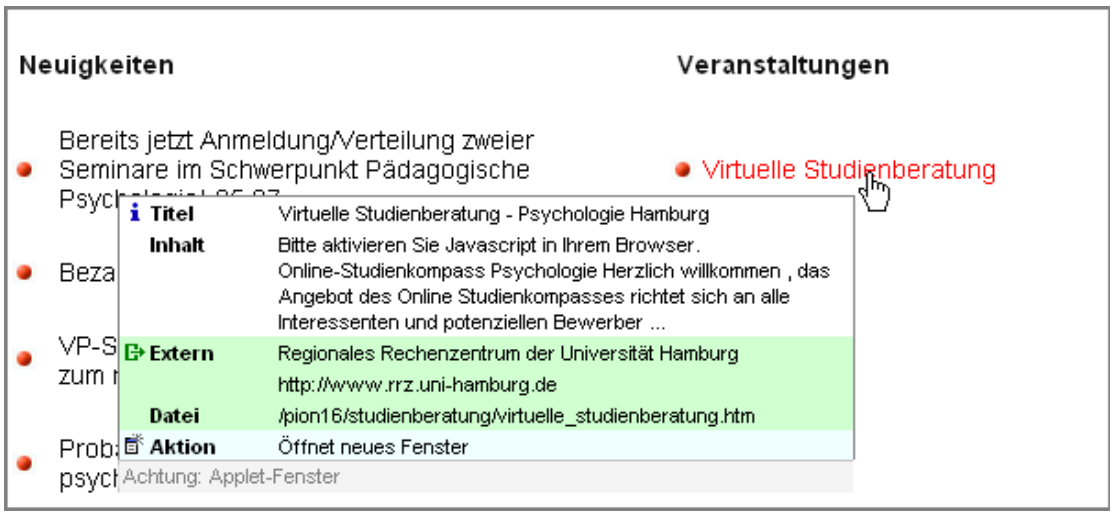


Abb. 90: Bei externen Links wird eine Beschreibung der Site aus dem Titel der Homepage extrahiert (hier: „Regionales Rechenzentrum der Universität Hamburg“).

Bezüglich der Definition externer Links variierten die Ansichten der Benutzer: Beispielsweise stuften zwei Teilnehmer alle Verweise zwischen Sites am Fachbereich Informatik als „intern“ ein, auch wenn sie unterschiedliche Domain-Namen hatten, andere taten dies hingegen

nicht. Ein Vorschlag war, für externe Links nur Unterschiede in der Second-Level Domain[163] zu berücksichtigen. Eine weitere Alternative wäre eine semantische Klassifikation, die sich nicht nur am Domain-Namen orientiert, sondern an der Sicht des Anwenders auf die Organisationsstrukturen des Anbieters. Dies zeigt gleichzeitig, dass bis heute eine eindeutige, für den Benutzer nachvollziehbare Definition *externer Links* fehlt.

6.2.3.3 Die Link-Aktion

Hinweise zur Link-Aktion (s. Abschnitt 5.5.4.3) lösten sehr unterschiedliche Resonanz aus. Die Angabe *Referenz* für „*Reference Links*" – also Verweise innerhalb einer Seite, bei denen der Browser lediglich nach unten scrollt (vergl. Abschnitt 4.5.1.3) – wurde zuerst von keinem der Teilnehmer verstanden. Nach einer Erläuterung des Begriffes wurde dieser Hinweis aber von fast allen als hilfreich angesehen. Der Bezeichner und das ebenfalls missverständliche Icon (s. Abb. 91) waren daher zu überarbeiten.

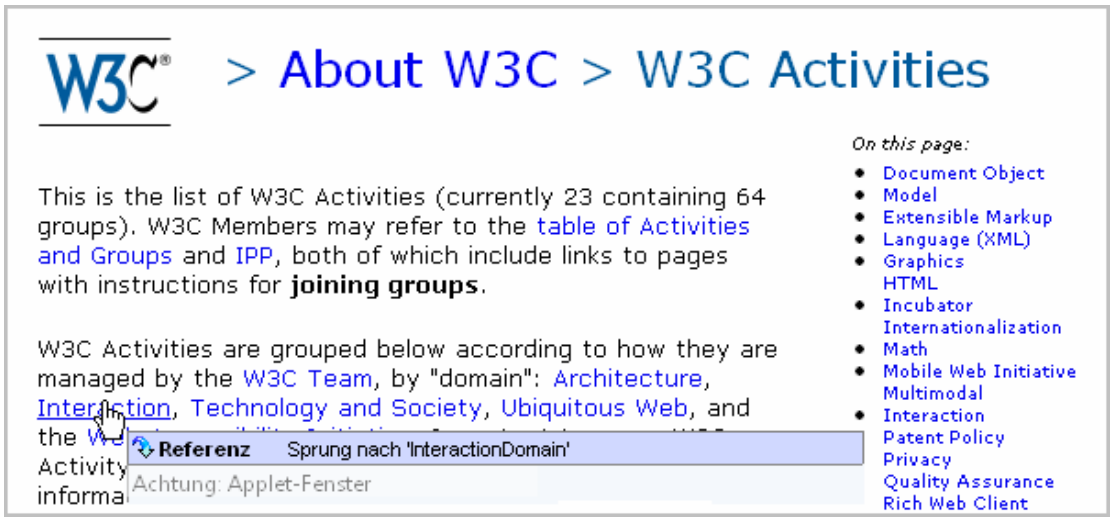


Abb. 91: Ein Link, der einen Bereich weiter unten auf der Seite referenziert.

Bei „mailto:"-Links wurde im Tooltip die E-Mail-Adresse des Empfängers angezeigt. Diese Tooltips wurden von allen Teilnehmern sehr positiv aufgenommen, zumal sie klein waren.

Die Angabe, dass ein Link ein neues Browser-Fenster öffnet, fanden drei Teilnehmer hilfreich, da dies ein unerwartetes Verhalten des Browsers darstellt. Die anderen vier Personen fanden das eher „nebensächlich". Zwei Teilnehmer würden gerne das Öffnen eines neuen Fensters verhindern können, nur dann würde eine solche „Warnung" hilfreich sein.

Hinweise darauf, dass ein Link einen anderen Frame steuert oder ein JavaScript-Programm aufruft, stießen hingegen durchgängig auf Kritik („nur technisch bedeutsam", „belangloser Hinweis"). Diese Informationen seien zudem nicht aussagekräftig genug, um bei der Navigation weiterzuhelfen.

---

[163] Beispielsweise wären Links zwischen „asi-www.informatik.uni-hamburg.de" und „vsis-www.informatik.uni-hamburg.de" nicht site-extern, da sie gemeinsam zur Second-Level Domain „uni-hamburg.de" gehören.

6.2.3.4 Informationen zum Inhalt des Zielobjektes

Die HyperScout-Hinweise zum Inhalt des Zielobjektes (s. Abschnitt 5.5.4.4) bewerteten die Teilnehmer je nach Art der Information sehr unterschiedlich.

Der Seitentitel

Der *Titel* des Zieldokuments wurde häufig als „wichtig" oder „nützlich" eingeschätzt. Allerdings gab es auch des Öfteren Kritik: Viele Titel von Webseiten sind nicht sehr informativ, manchmal sogar irreführend, und entsprechend nutzlos war dann der Text im Tooltip. In anderen Fällen führte der dargestellte Titel nicht über die Information im Link-Anker hinaus; er wurde dann als „überflüssig" erachtet und sollte folglich im Tooltip weggelassen werden.

Die Beschreibung

Die *Beschreibung* der Zielseite entnahm der HyperScout-Prototyp dem entsprechenden Meta-Element (s. Abschnitt 4.5.2.5). Bei einer der Aufgaben[164] verfügten die meisten besuchten Seiten über solche Elemente mit jeweils kurzen, redaktionell erstellten Zusammenfassungen (s. Abb. 74). Die Darstellung dieser Information im Tooltip lobten alle Teilnehmer; es wurde jedoch auch angemerkt, dass einige Links verständlich genug seien und dann weitere Angaben überflüssig wären.

Die Inhaltsangabe

Fehlte das Meta-Element mit der Beschreibung, so generierte der HyperScout-Prototyp aus dem Anfang des Content-Bereiches des Link-Ziels eine kurze *Inhaltsangabe* (s. Abschnitt 4.5.2.4). Dieser Text wurde deutlich seltener als die Beschreibung als nützlich eingeschätzt und häufig als zu „unpräzise", „unstrukturiert" und „zu lang" bemängelt. Die Teilnehmer wünschten „eine eher stichpunktartige Zusammenfassung", die „genau die wichtigsten Informationen der Seite zusammenfasst" und „übersichtlich präsentiert". Folglich werden bessere Algorithmen für die Zusammenfassung von Webseiten benötigt (siehe Harper, Goble et al. 2004) oder ein *standardisiertes* und *obligatorisches* HTML-Element für die Inhaltsbeschreibung.

Alternativ wäre zu überlegen, ob für eine inhaltliche Charakterisierung von Webseiten nicht auch andere Quellen in Betracht kommen. Vielversprechende, neue Ansätze stammen aus dem Social-Navigation-Bereich, bei dem die Anwender selbst Seiten mit Kommentaren oder auch „Tags" versehen und diese anderen Benutzern über ein gemeinsam verwendetes Kollaborationssystem zur Verfügung stellen (Wu, Zhang et al. 2006).

[164] Dabei war nach aktuellen Nachrichten auf dem „Heise Online Newsticker" zu suchen: http://www.heise.de/newsticker/

Autor und Anbieter

Die Information zum Autor wurde ebenfalls aus dem entsprechenden Meta-Element des Link-Zieles entnommen (s. Abschnitt 4.5.2.2). Dieses Element ist im Web relativ rar (vergl. Google 2006), zumal es von keinem Web-Client und keiner globalen Suchmaschine berücksichtigt wird. Erschien eine entsprechende Angabe im Tooltip (z. B.: „*Autor: Jan-Peter Richter*“, s. Abb. 77), so stuften sie die Teilnehmer durchgängig als „belanglos“ und „unbedeutend“ ein. Im Interview zeigte sich, dass dies im Wesentlichen daran lag, dass die während der Studie im Tooltip angezeigten Autoren den Teilnehmern unbekannt waren.

Die Beschreibung der Site mit Hinweisen zum *Anbieter* der Informationen – die aus dem Titel der Homepage der Site genommen wurde – sahen hingegen mehrere Teilnehmer als bedeutsam an (s. Abschnitt 6.2.3.2 und Abb. 83). Die Bewertung der Nützlichkeit solcher Daten scheint entscheidend davon abzuhängen, ob der Anwender aufgrund der Information Rückschlüsse auf *Qualität* und *Verlässlichkeit* einer Ressource ziehen kann.

Der Aktualisierungszeitpunkt

Die Angaben zur letzten Aktualisierung des Ziel-Dokuments wurden recht unterschiedlich bewertet: Nur zwei Teilnehmer fanden Hinweise, dass es „*alt*“ sei (für mehr als ein Jahr alte Objekte) relevant, aber fast alle lobten den Hinweis auf „*neue*“ Seiten (für innerhalb des letzten Monats aktualisierte Ressourcen).

Kritisiert wurde mehrfach, dass die Daten zur letzten Aktualisierung nicht in jedem Tooltip zu finden seien. Teilweise kam es auch vor, dass das Datum im Tooltip nicht mit den Angaben im Dokument übereinstimmte. Dies irritierte die Teilnehmer.

Leider werden die Daten zur letzten Aktualisierung einer Ressource nicht von allen Webservern bereitgestellt, oder sie sind inkonsistent. Zudem fehlen bis heute Standards für die Angabe solcher Metadaten in Webseiten (Daviel 1996; Google 2006). Wenn man bedenkt, welch hohe Bedeutung das Ausgabedatum bei gedruckten Medien hat, wird deutlich, wie sehr solch essenzielle Metadaten gegenwärtig noch im Web vernachlässigt werden.

Die Sprache

Den Hinweis auf die Sprache des Zieldokuments fanden alle Teilnehmer genau dann wichtig, wenn sie diese nicht oder nur ungenügend beherrschten. Benutzer sollten daher die ihnen vertrauten Sprachen auswählen können.

6.2.3.5 Der technische Typ des Zielobjekts

Angaben zum technischen Typ des Zielobjektes (s. Abschnitt 5.5.4.5) waren für fast alle Teilnehmer bedeutsam. Lediglich zwei Teilnehmer schränkten ein, dass der Hinweis für sie nur von Interesse sei, wenn das Zielobjekt nicht direkt im Browser dargestellt werde. Zudem wurde vorgeschlagen, dass aus Konsistenzgründen im Tooltip immer das im Betriebssystem des Benutzers eingesetzte Datei-Icon verwendet werden soll.

6.2.3.6 Der Medientyp des Link-Ziels

Informationen zum Medientyp des Zielobjektes (s. Abschnitt 5.5.4.6, z. B. das Dokument enthält Grafiken, Audio- oder Video-Dateien etc.) stellte der erste HyperScout-Prototyp in Bezug auf die direkt in die HTML-Seiten eingebundenen Objekte dar. Hinweise dieser Kategorie wurden von den Teilnehmern in der Regel *nicht* als hilfreich erachtet. Einige der Teilnehmer gaben aber an, dass solche Informationen (sofern sie „präzise und detailliert genug" seien), bei bestimmten Aufgabestellungen wie der Suche nach einem Audiodokument durchaus wichtig sind (vergl. Nation 1997).

6.2.3.7 Der topologische Typ des Link-Ziels

Informationen zum topologischen Typ des Zieldokuments (s. Abschnitt 5.5.4.7) wurden zumeist nicht verstanden: Hinweise wie *Hub-Seite* (mit hauptsächlich externen Links), *Index-Seite* (mit vielen internen Links und wenig Fließtext) und *Landmark* (für Übersichtsseiten, auf die fast alle anderen Seiten der Site verweisen) erforderten fast immer Erläuterungen durch den Testleiter. Lediglich drei der Teilnehmer bezeichneten die Angaben während der Studie als „irgendwie schon spannend" und „öfter interessant" – dies war aber offensichtlich auch in einem inhaltlichen Interesse an Konzept und Technik von HyperScout begründet.

Das Interview ließ gleichwohl erkennen, dass bei fast allen Teilnehmern ein *grundsätzliches* Interesse daran bestand, welche Navigationsmöglichkeiten ihnen das Link-Ziel bietet. Beispielsweise wurden Seiten mit vielen externen Verweisen (*Hub-Seiten*) als „häufig besonders bedeutsam für die Navigation" angesehen; Dokumente ohne Links (*Sackgassen*) müssten vor dem Folgen des Links erkennbar sein, da es „dort nicht weitergeht". Folglich waren *einige* der topologischen Informationen durchaus von Interesse, aber die Art der Präsentation und die verwendeten Bezeichnungen waren zu überarbeiten.

Verweise auf Seiten mit Formularfeldern wurden im Tooltip ebenfalls gekennzeichnet. Den entsprechenden Hinweis befanden die Anwender nur dann als zweckmäßig, wenn die Zielseite *primär* zur Eingabe von Daten vorgesehen war, nicht aber, wenn eine Seite lediglich ein optionales Eingabefeld (beispielsweise in der rechten oberen Ecke für die lokale Suche) enthielt.

6.2.3.8 Die Zugreifbarkeit des Ziel-Objekts

Von allen Teilnehmern wurden Daten zur *Verfügbarkeit* des Zieldokuments (s. Abschnitt 5.5.4.6) als wichtig angesehen und mit „hilfreich" bis „unverzichtbar" kommentiert. Die Warnung vor fehlerhaften Links wurde mehrfach als so entscheidend charakterisiert, dass sie bereits im Link-Anker zu erkennen sein sollte, damit diese Information „nicht zu übersehen" sei. Dies gilt auch für Hinweise auf nicht zugreifbare Webserver.

Zwei Teilnehmer schlugen vor, das HyperScout bei fehlerhaften Links zusätzliche Hilfen anbieten sollte. Beispielsweise wäre es wissenswert, „ob man zu einem späteren Zeitpunkt das gewünschte Dokument wieder erreichen könne" oder „wo sich eine Kopie des Dokuments

befindet“. Ein direkter Link zu einer Kopie wäre wünschenswert (s. Vorschläge der Teilnehmer im Abschnitt 6.2.5).

Eine Warnung vor Dokumenten, auf die man nur eingeschränkt (z. B. nur mit Passwortabfrage) zugreifen kann, wurde ebenfalls durchgängig als bedeutsam angesehen: Dabei seien Angaben über die Art der Zugriffsbeschränkung wichtig und ob man möglicherweise dennoch Zugang erhalten kann (z. B. ob der Benutzer bereits über ein Konto verfügt oder eine Zahlung leisten muss).

6.2.3.9 Die Latenzzeit bei der Navigation

Da sich die Zeitverzögerung bis zur Darstellung des Zielobjektes nicht prognostizieren lässt, wurde im HyperScout-Prototyp bei größeren Objekten der Umfang des Zielobjektes und bei externen Links die Antwortzeit des Servers dargestellt (s. Abschnitt 5.5.4.8).

Bei kaum einer anderen Information divergierte die Meinungen der Teilnehmer so gravierend wie bei der Größenangabe. Einige fanden sie grundsätzlich „überflüssig“, andere nur bei „wirklich großen Dateien“ sinnvoll, da sie „nichts über die Wichtigkeit und den Inhalt“ aussage. Zwei Teilnehmer wünschten sich hingegen eine Größenangabe bei *allen* Links, und es störte sie, dass sie bei kleinen Dateien fehlte. Unterschiedlich waren auch die Ansichten, ab wann eine Datei als „groß“ einzustufen sei; für einige Nutzer war dies bereits bei „100 Kilobytes“ gegeben, für andere erst ab „mehreren Megabytes“. Die Personalisierbarkeit solcher Parameter scheint daher am sinnvollsten.

Die Antwortzeit des Servers wurde zumeist nur bei externen Links und bei besonders langen Antwortzeiten als relevant angesehen, da man „ansonsten schon über die Geschwindigkeit des Webs im Bilde ist“, wie es ein Teilnehmer formulierte. Auf eine schnelle Antwortzeit müsse nicht explizit hingewiesen werden, da dies „normal“ sei. Die genaue Angabe der Millisekunden bis zur Antwort des Servers fand keiner der Teilnehmer informativ; es wurde stattdessen mehrfach die für das Zielobjekt geschätzte Downloadzeit in Minuten und Sekunden gewünscht.

6.2.3.10 Informationen zur Benutzung

Die Angabe des letzten Zugriffszeitpunkts auf das Link-Ziel (s. Abschnitt 5.5.4.9) wurde von allen Teilnehmern positiv bewertet und unter anderem als „sinnvoll“ und „sehr hilfreich“ charakterisiert. Die ebenfalls dargestellte Anzahl der vorherigen Seitenbesuche fanden sie ebenfalls nützlich. Lediglich die Bezeichnung des Attributs „Zurück“ (s. Abb. 68 und Abb. 71) wurde mehrfach kritisiert (s. Abschnitt 6.2.2). Hinweise zum letzten Besuch einer *Website* bei externen Links sah nur ein Teilnehmer als hilfreich an.

Aufgrund der recht kurzen Testdauer von jeweils unter 2,5 Stunden zeigte der Prototyp den Studienteilnehmern nur spärliche Benutzungsdaten an. Das Potenzial der Integration solch detaillierter „History-Informationen“ in die Navigation kann wohl nur in Langzeitstudien

ermittelt werden, zumal HyperScout im Gegensatz zum Webbrowser sämtliche Seitenabrufe im Web *dauerhaft* speichern soll.

Evaluationsergebnisse zur Einbindung von Social-Navigation-Hinweisen (s. Abschnitt 4.5.5.2) in HyperScout findet man in (Wollenweber 2004) und (Baier, Weinreich et al. 2004).

6.2.3.11 Der URI des Link-Zieles

Alle Teilnehmer gaben an, dass sie für gewöhnlich „häufiger" oder „fast immer" auf den URI im Statusbereich des Browsers achten würden (siehe Abschnitte 5.3.5 und 6.3.3). Der HyperScout-Prototyp stellte stattdessen den URI im Tooltip *zweizeilig* dar (s. Abschnitt 5.5.4.11): Die erste Zeile enthielt den Namen der Domain, die zweite Pfad- und Dateinamen (vergl. Abb. 83 und Abb. 84).

Die zweigeteilte Darstellung wurde von fast allen Teilnehmern kritisiert, da sie diese „sehr ungewohnt" fanden; lediglich ein Benutzer nannte sie „zwar ungewohnt, aber eigentlich besser". Folglich sollte – wenn überhaupt – eine andere Methode zur optischen Trennung von Domain-Name und Ressourcenangabe auf dem Server gewählt werden.

6.2.4 Zusammenfassende Beurteilung der Bedeutung impliziter Link-Informationen für die Navigation

Einige der angebotenen Informationen in den Tooltips wurden fast durchweg als hilfreich für die Navigation im Web eingeschätzt, andere fast immer als unbedeutend, und bei vielen gingen die Meinungen deutlich auseinander. Am wichtigsten wurden Hinweise zur *Verfügbarkeit* des Link-Zieles, zum *Dateityp* und zum *letzten Besuch* bewertet. *Externe Links*, Verweise auf *dieselbe Seite* oder *innerhalb einer Seite* haben die Mehrzahl der Teilnehmer als bedeutend für die Navigation erachtet. In der jetzigen Form fanden die meisten Teilnehmer den *semantischen Link-Typ*, den *topologischen Typ* der Zielseite und Informationen zum *Autor der Seite* als nebensächlich. Unterschiedliche Meinungen gab es unter anderem zu einem Großteil der *inhaltlichen* Informationen, zur *Dateigröße* und zur *Link-Aktion*.

Insgesamt machte die Studie deutlich, dass mehrere Faktoren für die Nützlichkeit von Link-Informationen in Tooltips entscheidend sind:

- Die Informationen müssen für den Benutzer *einfach verständlich* sein,
- es wird erwartet, dass sie *korrekt* und *konsistent* mit dem Link-Ziel sind,
- sie sollten *relevant* im Aufgabenkontext des Anwenders sein und
- sie sind aus gründen der Effizienz auf das *Wesentliche zu reduzieren*.

Mehrfach wurde in dem Zusammenhang angeregt, Text nur für besonders wichtige oder anders nicht darstellbare Attribute zu verwenden und alle anderen Informationen grafisch zu visualisieren, da sie dann schneller wahrnehmbar seien. Hierfür wurden Symbole, Icons und Farbcodierungen vorgeschlagen. So könnte nach Ansicht eines Teilnehmers beispiels-

weise ein fehlerhafter Link mittels einer „roten Ampel“ besser vermittelt werden als durch einen Hinweistext, da so die Information „peripher wahrnehmbar wird“:

> „Nach dem Schema Go oder Don’t Go müsste man sofort erkennen können, ob ein Link OK ist oder nicht [– also auch, ob er] zu dem für die aktuelle Aufgabe benötigten Dokument führt“.

Besonders relevante Informationen (wie fehlerhafte Links) sollten noch *auffälliger hervorgehoben* werden, damit sie nicht übersehen werden. Zwei Vorschläge waren, bei fehlerhaften Links bereits den Link-Anker besonders zu kennzeichnen (z. B. in roter Schrift) oder kleine rote Kästen hinter den Ankern einzufügen.

Eine entscheidende Herausforderung für den nächsten HyperScout-Prototyp bestand daher in der Optimierung der Auswahl und Reihenfolge der dargestellten Link-Informationen. Die Thinking-Aloud-Studie lieferte hierfür aufgrund der kleinen Teilnehmerzahl kein eindeutiges, signifikantes Ergebnis. Es wurde daher zusätzlich eine Online-Umfrage zur Bedeutung einzelner Link- und Objekt-Attribute des Webs durchgeführt (s. Abschnitt 6.3).

6.2.5 Innovative Vorschläge der Teilnehmer zur Vereinfachung der Link-Navigation

Während der Tests machten alle Testteilnehmer auf direkte und indirekte Weise Vorschläge für Erweiterungen des HyperScout-Konzepts. Einige der Anregungen spiegeln die Erfahrungen der Benutzer mit anderen Systemen wider, andere ergaben sich auch aus Missverständnissen („Ich hätte jetzt eigentlich Folgendes erwartet…“).

- Häufig wurde vorgeschlagen, dass die Tooltips bereits Auswahlmöglichkeiten ähnlich einem Popup-Menü bieten sollten, um zusätzliche Optionen zu erhalten.
- Bei site-externen Links wäre ein direkter Verweis zur Homepage der externen Site nützlich.
- Für fehlerhafte Links sollte im Tooltip eine Liste mit Links zu alternativen Dokumenten oder gespiegelten Versionen der Seite angeboten werden (z. B. aus dem *Google Cache* oder dem *Internet Archive*).
- Ein Teilnehmer wünschte sich „interaktive“ Tooltip-Elemente, die er beliebig lange offen halten und in seinem Browser wie Bookmarks sammeln und verwalten könnte. Er nannte sie *Sticky Tooltips*. Sie wären seines Erachtens verständlicher als heutige Bookmarks und erlaubten es ihm, später einen Navigationspfad weiterzuführen.
- Einige Personen äußerten sich positiv darüber, dass ihnen HyperScout „offenbar“ gänzlich neue Informationen zugänglich macht: Beispielsweise würde sie das Aktualisierungsdatum und der Anteil externer Links oft auch für die aktuelle Seite interessieren. Eine entsprechende Erweiterung heutiger Webbrowser sollte in Betracht gezogen werden.

Alle Teilnehmer wollten HyperScout auf die eine oder andere Weise an ihre Anforderungen anpassen: So sollten die Tooltips nicht nur einfach an- und ausschaltbar sein, man müsste auch die Zeit bis zu ihrem Erscheinen angeben und die benötigten Informationen auswählen

können. Weiterhin müssten Parameter wie die vertrauten Sprachen und eine Schranke für die Anzeige der Dateigröße festzulegen sein. Ein Anwender war der Ansicht, dass er für verschiedenartige Aufgaben *unterschiedliche Profile* benötige. Beispielsweise wären beim Erstellen von Webseiten eher technische Informationen (Fehlernummern, genaue Dateigrößen etc.) notwendig als „beim Herumstöbern im Web". Im Optimalfall sollte dem System die aktuelle Aufgabe des Anwenders bekannt sein, damit es automatisch auf seine Bedürfnisse reagieren könne.

6.2.6 Fazit der Evaluation von HyperScout I

Die Ergebnisse der Studie waren motivierend, da deutlich wurde, wo die Stärken des Konzepts sind und welche Möglichkeiten sich zur Reduktion der meisten der gefundenen Probleme bieten.

Grundsätzlich wurde es von allen Teilnehmern als sinnvoll angesehen, in bestimmten Fällen zusätzliche Informationen zu Links verfügbar zu machen, insbesondere wenn es sich um „kritische" Informationen für die Navigation (wie Fehlermeldungen) handelt. Die Ergebnisse zeigen aber, dass gerade Tooltips für die Interaktion mit Links im Web nur eingeschränkt geeignet sind. Die Interaktion erfahrener Benutzer mit dem Browser ist in den meisten Fällen so schnell, dass ihnen bereits eine Wartezeit von 0,8s oft als zu lange erscheint und sogar bei einigen Teilnehmern das Gefühl aufkam, „Zeit zu verlieren". Zwei Teilnehmer wünschten sich folglich eine Art „aktiven HyperScout-Modus, bei dem die Tooltips ohne Zeitverzögerung erscheinen" und der mittels einer Schaltfläche im Browser oder durch das Drücken einer Taste aktiviert werden kann.

Die Tooltips wurden in der Regel häufiger berücksichtigt, wenn die Anwender bei der Navigation zögerten. Dies konnte unter anderem daher rühren, dass sie einen Link missverständlich fanden, die Website noch nicht kannten oder nicht genau wussten, wie sie weiternavigieren sollten. Ebenso wurden die Tooltips beim Zugriff auf *langsame Server* öfter beachtet, da dann die Wartezeit auf das Zielobjekt länger war. Ein potenzieller Einsatzbereich für erweiterte Link-Informationen in Tooltips sind somit Anwender, deren Systeme über eine geringe Bandbreite verfügen, beispielsweise bei ISDN-Zugang oder mobilem Zugriff auf das Internet.

Es erwies sich oft als Nachteil, dass die Teilnehmer den Mauszeiger immer erst zum Link-Anker bewegen mussten, bevor ihnen die Zusatzinformationen dargestellt wurden. Die Maus bewegten sie zumeist aber erst, *nachdem*[165] sie sich für einen Link entschieden hatten,

[165] Die Problematik, dass sich Benutzer erst für einen Link entscheiden, bevor sie den Mauszeiger zu ihm bewegen, wurde bereits von (Ojakaar 2001) in einer Studie mit Mouseover-Menüs in Webseiten festgestellt. In der Studie waren die Benutzer von den (unerwartet) aufklappenden Menüs überrascht, was zu meist schlechteren Leistungen führte. Hierin liegen gleichzeitig die wesentlichen Unterschiede beider Studien, denn bei *HyperScout I* sind die Tooltips nicht unerwartet (da für alle Links verfügbar) und es handelt sich um optionale Zusatzinformationen.

um ihn dann sofort anzuklicken, sodass kein Tooltip erschien. Folglich sollten wichtige Link-Hinweise ohne Verzögerung sichtbar werden und besonders kritische Link-Eigenschaften (wie fehlerhafte Links) schon erkennbar sein, bevor der Mauszeiger zum Link-Anker geführt wird. Möglichkeiten hierzu wurden im zweiten HyperScout-Prototyp evaluiert (s. Abschnitt 6.5).

Alle Teilnehmer gaben an, dass sie die Nutzung und Evaluation des HyperScout-Systems kurzweilig fanden. Drei der sieben Teilnehmer äußerten von sich aus den Wunsch, das System in der getesteten Version bei sich einzusetzen. Die HyperScout-Tooltips würden das Web „zusammenkitten“ und „konsistenter“ machen, da nun bei allen Links Zusatzinformationen zum Link und zum Link-Ziel verfügbar seien.

6.3 Eine Umfrage zur Problematik der Link-Typisierung und -Navigation im Web

Die Thinking-Aloud-Studie von HyperScout I hat detaillierte qualitative Ergebnisse zur Benutzbarkeit der Link-Tooltips geliefert. So wurde unter anderem deutlich, dass die Anzahl der angebotenen Informationen auf das Wesentliche reduziert werden muss und ihre Auswahl und Reihenfolge für die Nützlichkeit der HyperScout-Tooltips entscheidend sind.

Allerdings war der Umfang der im partizipativen Test erhobenen Daten alleine nicht ausreichend um eine Bedeutung der Link-Attribute für die Nutzung in Link-Previews zu bewerten. Aus diesem Grunde wurde zusätzlich eine Umfrage durchgeführt, in die sich wesentlich mehr Personen für eine Beurteilung der Attribute einbeziehen ließen und in deren Rahmen weitere Erkenntnisse zu den Problemen der Link-Navigation gewonnen werden sollten.

6.3.1 Studienziele und Vorgehen

Die Online-Umfrage hatte zum Ziel, eine genauere Einschätzung der Bedeutung der implizit vorhandenen Link- und Objektattribute für die Navigation im Web zu gewinnen. Zudem sollte verifiziert werden, in wieweit Benutzer ihrer Erfahrung nach zusätzliche Navigationshinweise zu Hyperlinks benötigen und ob sie bereits auf den im Statusbereich des Browsers angebotenen URI zurückgreifen.

Die Zielgruppe der Befragung waren regelmäßige Benutzer des Webs, da eine gewisse Erfahrung mit dem Medium eine Voraussetzung für das Verständnis der Problematik und eine fundierte Einschätzung der eigenen Bedürfnisse war.

Die Umfrage wurde als Online-Fragebogen realisiert (s. Abb. 92 auf S. 207). Dies bot die Vorteile, mehr Personen erreichen und die Daten digital erfassen zu können. Der Fragebogen wurde auf dem Webserver der Arbeitsgruppe VSIS installiert und war über einen Zeitraum von 8 Wochen (26. Januar bis 21. März 2002) zugänglich. Von mehreren populären Seiten des Fachbereiches Informatik der Universität Hamburg verwiesen Links zu dem Fragebogen.

Hierzu gehörte die Startseite der „10 WWW-Ergonomie-Leitlinien“,[166] die pro Monat von ca. 2000 Personen besucht wurde, die Startseite der Arbeitsgruppe *VSIS* und die Übersichtsseite der Grundstudium-Vorlesung *Praktische Informatik 3* der Universität Hamburg.

Der Fragebogen bestand aus einem Web-Formular mit sechs offenen und 26 Multiple-Choice-Fragen. Er gliederte sich in *drei Bereiche*: die Erhebung *demografischer Daten*, allgemeine *Fragen zur Navigation mit Links im Web* und die *Bedeutung von Link- und Objektattributen* für die Navigation im Web. Der Fragebogen wurde vor Beginn der Umfrage mit acht Personen erprobt und dabei sukzessive bezüglich der Verständlichkeit optimiert.

Obgleich die Studie bereits vor einiger Zeit stattgefunden hat, sind die Ergebnisse potenziell weiterhin relevant, da seither die Link-Benutzungsschnittstelle von Browsern – und somit die generelle Problematik – unverändert geblieben ist.

6.3.2 Ergebnisse der Umfrage

Insgesamt nahmen 143 Personen an der Umfrage teil, davon beantworteten 138 alle Fragen. Die übrigen haben die Fragen zu den Link-Attributen nicht oder nicht komplett ausgefüllt und blieben daher bei dem Fragenkomplex unberücksichtigt.

Fast alle Teilnehmer (98%) entsprachen der Zielgruppe „erfahrene Benutzer des Webs“, wobei sich 52% selbst sogar als „sehr erfahren“ einstuften und 92% angaben, das Web seit über 2 Jahren zu nutzen. Dies spiegelt sich auch in den Nutzungsgewohnheiten der Teilnehmer wider: 88% benutzten das Web „täglich“, die restlichen entweder mehrfach pro Woche oder Monat.

71% der Teilnehmer ordneten sich selbst fachlich dem EDV-Bereich zu, die restlichen Personen gaben einen anderen beruflichen Schwerpunkt an; hierzu gehörten Medizin, Psychologie, Journalismus und Weinbau. Ein hoher Anteil der Teilnehmergruppe kam aus dem akademischen Bereich: Insgesamt waren 62% Studierende (durchschnittlich im sechsten Fachsemester) und 7% wissenschaftliche Mitarbeiter. Die restlichen 31% waren nicht oder nicht mehr an einer Universität. Das durchschnittliche Alter der Personen war 26,9 Jahre. Bei der Interpretation der Daten ist daher zu berücksichtigen, dass es sich nicht um eine repräsentative Umfrage mit „durchschnittlichen“ Benutzern des Webs handelt, sondern fast alle Teilnehmer Erfahrung im Umgang mit dem Web hatten und ein Großteil sogar mit seinen technischen Aspekten vertraut war.

[166] Die 10 Leitlinien für ergonomische Webseiten wurden vom Autor dieser Dissertation 1997 erstellt und sind unter http://vsis-www.informatik.uni-hamburg.de/ergonomie/ zu finden.

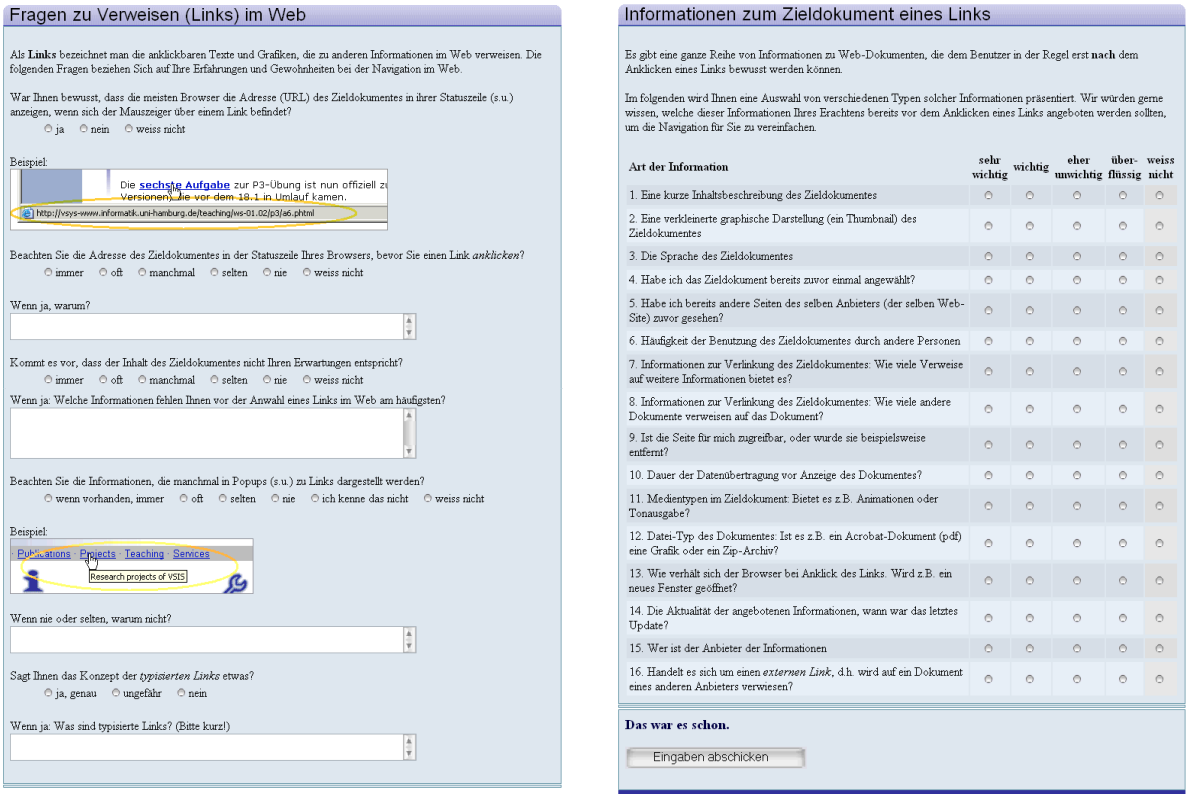


Abb. 92: Ausschnite aus dem Online-Fragebogen.

6.3.3 Probleme der Teilnehmer bei der Navigation mit Hyperlinks

In der Umfrage wurden zuerst Fragen zur Verständlichkeit von Links bei der täglichen Benutzung des Webs gestellt, um die aus der Hypertext-Forschung stammenden Erkenntnisse zu den Herausforderungen bei der Navigation im Hypertext auch für das Web zu bestätigen (s. Abschnitt 3.4). Das Ergebnis belegt recht eindrucksvoll die Probleme der Anwender bei der Navigation im Web: Fast 30% der Teilnehmer waren der Meinung, dass das Ziel eines Links *oft* nicht ihren Erwartungen entspräche, und insgesamt über 85% gaben an, dass zumindest *manchmal* Links zu unerwarteten Seiten führen würden. Weniger als 13% der Teilnehmer sahen dies als ein „seltenes“ Problem an (s. Abb. 93; auf 100% fehlende Werte sind „keine Angabe“).

Eine der wenigen Quellen, die dem Benutzer bei der Navigation im Web (fast) immer als Zusatzinformation zu einem Link zur Verfügung steht, ist der URI im Statusbereich des Browsers (s. Abschnitt 5.3.5 und Abb. 94). Von den 143 Teilnehmern der Studie gaben über 60% an, dass sie diese Adressangabe des Browsers *oft* oder sogar *immer* berücksichtigen würden, bevor sie einen Link anklicken. Nur ca. 11% beachten diese Information *selten* oder *nie* (s. Abb. 94). Der hohe Anteil an Benutzern, die diese recht technische Zusatzinformation nutzen, ist sicherlich auch auf die große Erfahrung der Teilnehmer im Umgang mit dem Web zurückzuführen, dennoch stützt diese Ergebnisse die Annahme, dass für viele Benutzer der Link-Anker als Schnittstelle allein oft nicht aussagekräftig genug ist und der Bedarf nach zusätzlichen Informationen besteht.

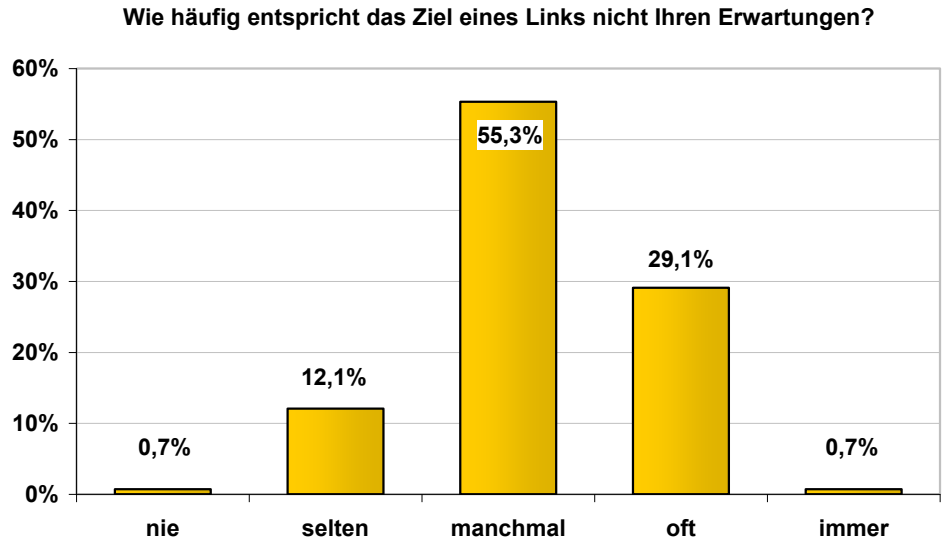


Abb. 93: Erwartungskonformität von Links im Web.

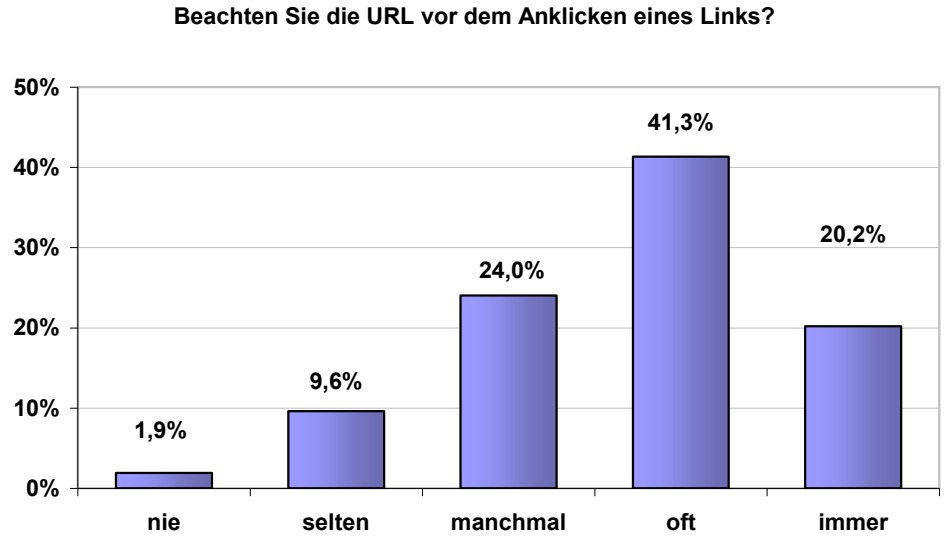


Abb. 94: Die Berücksichtigung des URI im Statusbereich des Browsers.

Auf die offene Frage, *warum* sie die Adresse des Zieldokuments beachten, gaben die Teilnehmer mehrere Gründe an. Zumeist gaben sie eher vage Antworten, beispielsweise dass sie aus „Gewohnheit“ so agieren und „allgemein mehr über das Ziel des Links erfahren“ wollen (54 Nennungen). Konkret wurde am häufigsten das Bedürfnis angeführt, *externe Links* frühzeitig erkennen zu können (28), die *Quelle* und *Authentizität* des Dokuments zu überprüfen (14) sowie Links zu *Werbung* und ungewollten *kommerziellen Angeboten* zu umgehen. Ähnlich häufig bestand das Bedürfnis, den *Dokumententyp* des Zielobjektes zu identifizieren (13). Seltener genannt (unter 10 Nennungen) wurde die Sorge vor spam- und virenverseuchten Seiten, Hinweise auf SSL-gesicherte Verbindungen und ob ihnen das Ziel bereits bekannt sei.

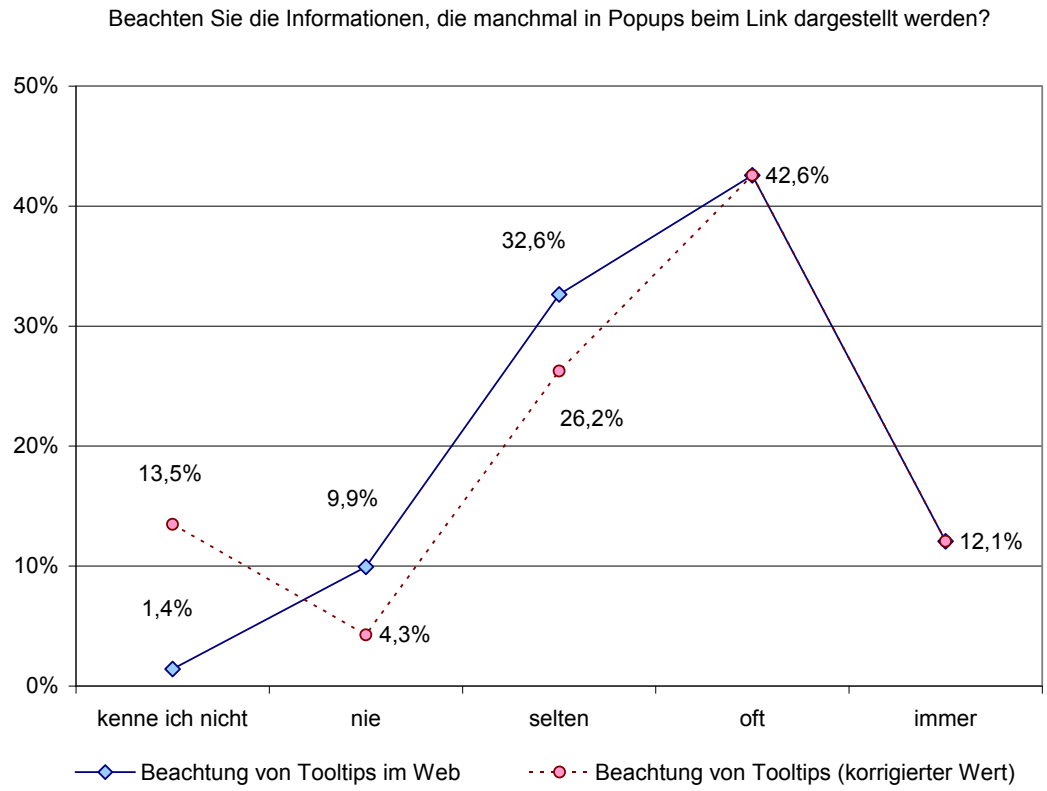


Abb. 95: Berücksichtigung der Tooltips mit Link-Titeln im Web.

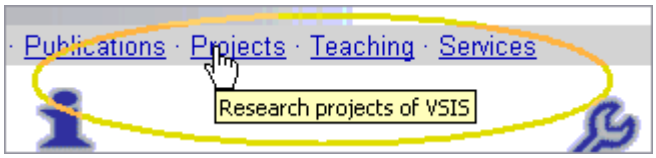


Abb. 96: Die Grafik aus dem Online-Fragebogen zur Illustration von Link-Tooltips.

6.3.4 Zur Bedeutung von Tooltips und typisierten Links im Web

Ein weiterer Fragenblock diente dazu, mehr über den Umgang mit und die Kenntnisse über explizit typisierte Links im Web zu erfahren. Als erstes wurde auf die Möglichkeit eingegangen, Links in HTML mit dem „`title`"-Attribut frei zu typisieren (s. Abschnitte 4.2.2 und 4.4.2). Dieser Link-Titel wird von gängigen Browsern in einem „Tooltip" (bzw. „Popup"[167]) beim Link-Anker dargestellt. Die Teilnehmer sollten angeben, ob für sie diese Tooltips als zusätzliche Informationsquelle von Interesse sind, ob sie sie nutzen und was sie ggf. davon abhält.

Tooltips mit Link-Titeln werden (falls vorhanden) von einem etwas geringeren Anteil der Teilnehmer regelmäßig zur Kenntnis genommen als der URI im Statusbereich des Browsers: Fast 55% gaben an, Link-Tooltips im Web *oft* oder *immer* zu beachten, fast 42% (60 Personen) berücksichtigten sie hingegen nach eigener Aussage *selten* oder *nie* (s. Abb. 95, durchgezogene Linie).

[167] Da den Teilnehmern des Pretests der Begriff Tooltip unbekannt war, wurde im Fragebogen der Begriff Popup verwendet und das Konzept mit einer Grafik illustriert.

Die von Webbrowsern dargestellten Tooltips mit dem Link-Titel sollen Benutzern wertvolle Hinweise für die Navigation bieten. Daher wurden die Teilnehmer gefragt, *warum* sie diese gegebenenfalls nicht oder nur selten beachteten. Häufig wurde aufgeführt, dass die dort verfügbaren Informationen zumeist „nicht aussagekräftig" oder „unbedeutend" seien und „gegenüber dem Link-Text keinen Mehrwert" böten (13 Nennungen); zwei Teilnehmer waren sogar der Ansicht, dass solche Tooltips zu häufig „unzuverlässige und falsche Hinweise" enthielten. Weitere fünf Personen gaben an, dass sie keine zusätzlichen Informationen benötigen würden, da Links zumeist „verständlich" und „selbsterklärend" sind.

Ein zusätzlicher, häufig genannter Nachteil von Tooltips ist, dass sie erst nach einer kleinen Wartezeit erscheinen und das „Zieldokument oft schneller da sei als der Tooltip" (zusammen 12 Nennungen). Dies wurde auch als Defizit der HyperScout-Tooltips in der ersten Thinking-Aloud-Studie identifiziert (s. Abschnitt 6.2.1). Darüber hinaus gaben 9 Personen an, Tooltips zu übergehen, da sie „nur selten zu finden" und „ungewohnt" seien und man daher nicht weiß, wann sie erscheinen würden. Vier Teilnehmer kritisierten, dass Tooltips die Sicht auf andere wichtige Informationen verdecken könnten.

Erstaunlicherweise wurde häufig kritisiert (17 Nennungen), dass Tooltips (bzw. Popups) im Web „zumeist nur Werbung" enthalten würden. Dies ist allerdings eine irrtümliche Angabe: Im Fragebogen wurde aufgrund der Erfahrungen aus den Pretests aus Verständnisgründen der Begriff „Popup" statt „Tooltip" verwendet (der Term „Tooltip" war den Pre-Testern nicht bekannt), und das Konzept wurde anhand einer Grafik (s. Abb. 92 auf S. 207 und Abb. 96 auf S. 209) verdeutlicht. Offensichtlich schienen aber viele Web-Anwender eine so aversive Einstellung zu „aufpoppenden" Elementen zu haben, dass sie dieses Konzept unmittelbar mit Werbe-Popups gleichsetzen. Diese Angabe hat somit implizit aufgezeigt, dass mindestens 17 Teilnehmer die Frage falsch verstanden hatten und mit Tooltips für Link-Titel im Web nicht vertraut waren. Wenn man die Angaben dieser Teilnehmer entsprechend berücksichtigt, erhöht sich der Anteil der Personen, die Link-Tooltips nicht kennen von 1,4% auf 13,4% und der Anteil, die Tooltips ignorierten, reduziert sich auf weniger als ein Drittel der Teilnehmer (s. Abb. 95, gestrichelte Linie).

6.3.5 Zur Bedeutung einzelner Eigenschaften von Links und Zielobjekten

Der letzte Bereich des Fragebogens zielte darauf ab, herauszufinden, welche der impliziten Link-Attribute die wichtigsten seien, die in den Tooltips von HyperScout II dargestellt werden sollten. Die Teilnehmer wurden daher um ihre persönliche Einschätzung zur Bedeutung von 16 verschiedenen Attributen von Hyperlinks und Ressourcen im Web für die Navigation gebeten.

Es wurde über die Hälfte der vorgegebenen Attribute als interessant für die Navigation eingeschätzt (s. Abb. 97). Wichtig bis sehr wichtig fanden die meisten Teilnehmer Hinweise auf *fehlerhafte Links*, Informationen zu Verweisen auf unerwartete *Dateitypen*, Daten zum *Inhalt* des Zieles und Angaben zur *Aktualität* der angebotenen Informationen. Diese Informationen

sollten somit priorisiert dargestellt werden. Die Einschätzung deckt sich zum Großteil mit der Meinung der Teilnehmer in der partizipativen Studie von HyperScout I.

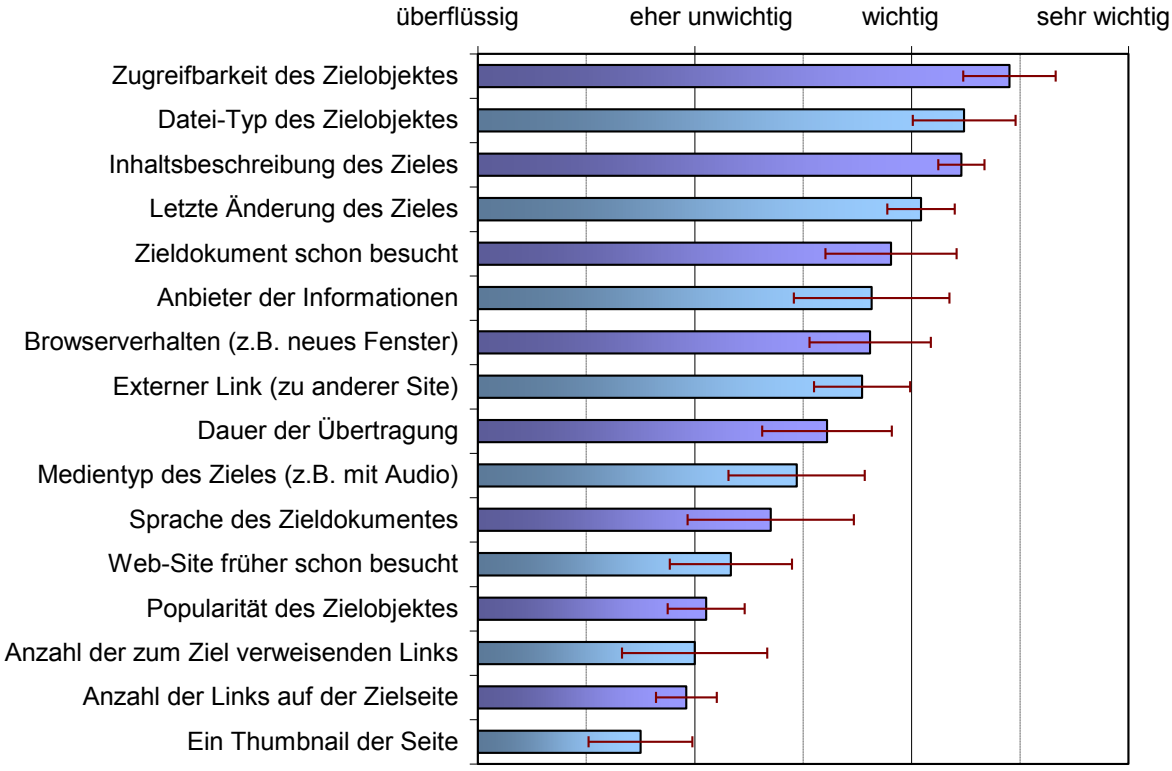


Abb. 97: Bewertung der Wichtigkeit von 16 Link- und Objekteigenschaften für die Navigation im Web.

Etwas weniger bedeutsam aber immer noch als „wichtig“ wurden Informationen zur *eigenen Benutzung* angesehen, also ob das Zieldokument früher *schon besucht* wurde. Gegenwärtig ist dies das einzige von Webbrowsern visualisierte Attribut (die Link-Farbe ändert sich von Blau zu Lila, s. Abschnitt 5.1.3 und Nielsen 2004a). Informationen zum *Anbieter* des Zieldokuments, zu *externen Links*, zum *Verhalten des Browsers* und zur *Dauer der Übertragung* wurden ebenfalls als wichtig eingestuft.

Bemerkenswert ist andererseits, wie unbedeutend als Vorabinformation ein *Thumbnail* der Zielseite, die *topologischen Eigenschaften* des Zieles (die Anzahl der ein- und ausgehenden Links) oder die Anzahl der *Besuche anderer Benutzer* gewertet wurden. Dies steht im Kontrast zu sonst oft in der Literatur vertretenen Annahmen (Cockburn, Greenberg et al. 1999; Wexelblat 1999; Teevan, Cutrell et al. 2009; Won, Jin et al. 2009; Liu & Tajima 2010).

Die Verteilung ist fast unverändert, wenn man die 71% Teilnehmer aus dem IT-Bereich von den restlichen getrennt betrachtet: Die meisten Attribute gelangen auf den gleichen Rang und nur wenige verschieben sich um einen der Plätze. Auffällig ist als einziges, dass die Nicht-ITler Vorabinformationen über die *Dauer der Übertragung* bedeutsamer einschätzen (Platz 7 „wichtig“ statt Platz 9 „eher unwichtig“). Dies ist eventuell auf eine durchschnittlich langsamere Internetanbindung dieser Benutzergruppe zurückzuführen.

Die meisten der am „wichtig“ und „sehr wichtig“ bewerteten Link-Eigenschaften sind aus technischer Sicht vergleichsweise einfach automatisch ermittelbar und darstellbar (Weinreich & Lamersdorf 2000; Weinreich, Obendorf et al. 2004) – gut für eine weitere Optimierung des HyperScout-Konzepts.

6.4 Optimierung der erweiterten Link-Schnittstelle: HyperScout II

Die Ergebnisse der Evaluation von HyperScout I (s. Abschnitt 6.2) und der Umfrage im Web (s. vorherigen Abschnitt) waren Grundlage für die Entwicklung des neuen Prototyps „HyperScout II“. Das System wurde in einem Thinking-Aloud-Test mit 13 Personen unterschiedlicher Fachrichtungen evaluiert (s. Abschnitt 6.5).

Ziele bei der Konzeption und Entwicklung von HyperScout II:

- Da viele Teilnehmer des ersten Tests nicht auf die HyperScout-Tooltips achteten, sollte die Anzeige von Link-Informationen auf „unmittelbarere“ Weise erfolgen. Dies bezieht sich nicht nur auf die Wartezeit bis zum Erscheinen der zusätzlichen Informationen, sondern auch auf die Notwendigkeit, erst den Mauszeiger zum Link-Anker bewegen zu müssen.
- Die visuelle Repräsentation der Informationen sollte optimiert werden, sodass Anwender kritische Eigenschaften schneller und ohne zu lesen erfassen können.
- Um den Benutzer möglichst wenig kognitiv zu belasten, waren die dargestellten Informationen auf die Wichtigsten zu beschränken.
- Die Farben, Icons und Bezeichnungen der Attribute waren zu optimieren.

Diese Ziele ließen sich nicht alleine durch eine Überarbeitung der HyperScout-Tooltips erreichen. Vielmehr war es notwendig, zusätzlich Methoden zur Visualisierung der Link-Informationen einzusetzen. Da die Testteilnehmer Tooltips unter bestimmten Bedingungen – wie missverständlichen Links und Sites mit langsamem Zugriff – als hilfreich und angemessen ansahen, wurden sie als optionale *Details-on-Demand* beibehalten.

Basierend auf der in Kapitel 5.3 durchgeführten Klassifikation grundlegender Konzepte zur Visualisierung von Link-Informationen wurden zwei zusätzliche Techniken zur Darstellung zusätzlicher Link-Hinweise ausgewählt: Erstens eine für das Web angepasste Variation *farbig codierter Link-Marker* (s. Abschnitte 5.3.3 und 6.4.1), zweitens eine erweiterte Variante des *adaptiven Mauszeigers* (s. Abschnitte 5.3.6 und 6.4.2). HyperScout II bietet somit ein *dreistufiges Konzept* zur Anzeige zusätzlicher Link-Informationen:

1. Kritische Attribute kann der Benutzer bereits anhand der Farbe des Link-Markers im Dokument erkennen.
2. Wird der Mauszeiger über einen Anker geführt, so werden sofort genauere Informationen zum Link als Icons beim Mauszeiger sichtbar.

3. Nach kurzer Wartezeit erscheinen Detailinformationen im Tooltip.

Die folgenden Abschnitte beschreiben die Entwicklung und Realisierung dieser drei Konzepte.

6.4.1 Farbig codierte Link-Marker für das Web: Link-Overlays-on-Demand

Farbige Link-Marker sind eine bewährte Methode zur Hervorhebung von Link-Ankern (s. Abschnitt 5.2.2). Sie eignen sich gleichzeitig zur Unterscheidung grundlegender Link-Eigenschaften (s. Abschnitt 5.3.3). Alle gängigen grafischen Webbrowser nutzen diese Methode zur Hervorhebung von Link-Ankern; Verweise zu vor kurzem besuchten Dokumenten werden gekennzeichnet, indem sie violett statt blau erscheinen (s. Abschnitt 5.1.3). Allerdings weist diese Methode für das Web Schwächen auf, da die Standardfarben der Link-Marker häufig von den Autoren geändert werden und bis heute bei grafischen Link-Ankern keine solche Unterscheidung möglich ist.

Für HyperScout II wurde daher das Konzept der *farbigen Link-Overlays* entwickelt. Sie basieren konzeptionell auf der Idee, Hintergrundfarben zur Hervorhebung von Link-Ankern einzusetzen (wie in Harmony, dem Hyper-G-Browser für Unix (s. Abb. 191; Andrews 1996), kombiniert mit den Vorteilen transluzenter (durchscheinender) Benutzungsschnittstellen. Der Einsatz transluzenter Elemente in grafischen Benutzungsoberflächen wurde bereits in den 1990er Jahren erfolgreich getestet (Harrison, Kurtenbach et al. 1995; Harrison & Vicente 1996; Cox, Chugh et al. 1998). Sie gewinnen in jüngerer Zeit zunehmend an Popularität, und „Transparenzeffekte“ gehören inzwischen zur Standardfunktionalität von grafischen Oberflächen wie *Microsoft Windows Vista*, *Cocoa* von *Apple OS X* und *KDE 4*.

Die Link-Overlays von HyperScout II sind transluzente farbige Flächen, die ähnlich wie Textmarker bei gedrucktem Text über die Link-Anker der Seite gelegt werden und sie (zusätzlich) hervorheben (s. Abb. 98 rechts). Die Overlays haben somit keinen Einfluss auf das Layout der Seite, können sowohl für Text als auch Grafiken eingesetzt werden und sind (relativ) unabhängig von den auf der Seite eingesetzten Farben. So werden alle Link-Anker einer Seite eindeutig hervorgehoben und für den Benutzer einfacher erkennbar (s. auch ISO9241-151 2008, 9.4.2). Durch die Verwendung unterschiedlicher Overlay-Farben lassen sich unmittelbar mehrere Link-Eigenschaften für den gesamten sichtbaren Bereich des Dokuments visualisieren, ohne dass der Benutzer den Mauszeiger zu einem Link-Anker bewegen muss (vergl. Abschnitt 5.3.3).

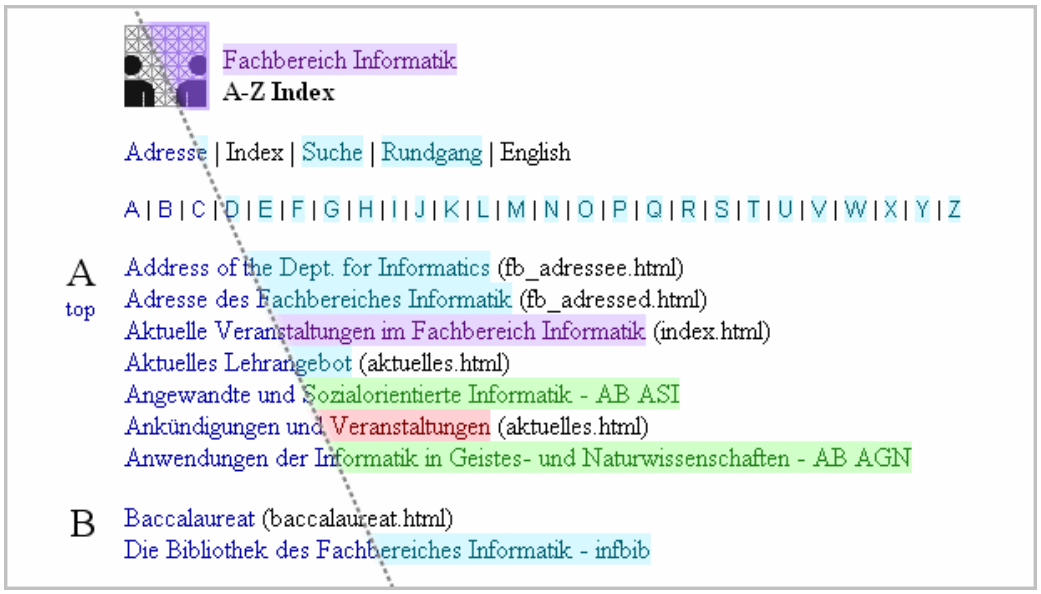


Abb. 98: Rechts im Bild sind die Link-Overlays-on-Demand des zweiten HyperScout-Prototyps zu sehen.

Da zu vermeiden war, dass die Link-Overlays die Seite verfremden oder die Lesbarkeit des Textes einschränken, wurden sie als „*Links-on-Demand*" realisiert (s. Abschnitt 5.2.8): Sie erscheinen erst, nachdem der Benutzer sie aktiviert hat, indem er eine bestimmte Taste drückt. Wird die Taste losgelassen, verschwinden die Link-Overlays wieder. Für *HyperScout II* wurde als Trigger die STRG-Taste (auch CTRL-Taste genannt) gewählt, da sie beim für die Studie primär verwendeten Browser (Internet Explorer 6) bei gleichzeitigem Anklicken eines Links keinen Einfluss auf die Link-Aktion hatte.[168]

Das Konzept, „*Link-Overlays-on-Demand*" für Hyperlinks einzusetzen, ist erstmals in (Weinreich, Obendorf et al. 2001) vorgeschlagen worden, erste Evaluationsergebnisse wurden in (Obendorf & Weinreich 2003) präsentiert. Zwei Ergebnisse dieser Studien waren, dass *Link-Overlays* die Lesbarkeit von Texten weniger reduzieren als Unterstreichungen, und dass bei *Links-on-Demand* die Teilnehmer den Text von Webseiten durchschnittliche länger lasen als bei immer sichtbaren Links. Der hier vorgestellte Prototyp setzt gegenüber dem damaligen Konzept mehrere Farben für die Overlays ein und kann auch grafische Link-Anker hervorheben.

Da Webseiten häufig über farbige Hintergründe und Texte sowie bunte Grafiken verfügen, wurden eindeutig unterscheidbare Farben für die Overlays benötigt. Unterschiedliche Helligkeiten und Sättigungswerte zur Darstellung mehrerer Link-Attribute erwiesen sich in informellen Pretests als ungeeignet, da die Farben vieler Webseiten einen zu starken Einfluss auf die Wirkung der Overlays hatten. Für eine Overlay-Farbpalette mit möglichst hohem Farbkontrast wurden daher die drei Grundfarben der additiven Farbmischung (Rot, Grün, Blau) und ihre drei paarweisen Mischfarben Gelb, Türkis und Violett (Magenta) gewählt.

[168] Sowohl die ALT- als auch die SHIFT-Taste (Umschalttaste) führen jeweils zu einer alternativen Browser-Aktion bei Link-Klicks: Es wird das Link-Ziel heruntergeladen oder in einem neuen Fenster geöffnet.

Leider zeigte sich bei weiteren Vorstudien, dass die Benutzer Probleme bei der Unterscheidung von *türkisen* und *blauen* Overlays hatten, insbesondere wenn sie über Grafiken eingeblendet wurden. Für HyperScout II wurde somit auf die Unterscheidung von Blau und Türkis verzichtet, und neben Rot, Gelb, Grün und Violett wird ein ins Türkise gehendes Hellblau verwendet.[169] Für eine Beschränkung auf fünf Link-Farben sprechen auch frühere Benutzbarkeitsstudien mit grafischen Schnittstellen, bei denen fünf Farben als Obergrenze für eindeutig zuzuordnende und unterscheidbare Textfarben ermittelt wurden (s. auch Byrd 1999 ; ISO9241-151 2008, 9.3.9 und Abschnitt 5.3.3).[170]

Eine weitere Herausforderung stellte die Zuweisung der Farben zu einzelnen Link-Attributen dar. Farblich lassen sich nur elementare Link-Eigenschaften repräsentieren, die weder numerischer noch textlicher Natur sind. Aufgrund dieser Einschränkung und der Ergebnisse der Online-Umfrage (s. Abschnitt 6.3.5) wurden folgende, als besonders relevant bewertete Link-Attribute ausgewählt: fehlerhafte Links, Verweise zu bereits besuchten Seiten, externe Links, Links zu „besonderen" Ressourcen und „normale" Links (s. Tabelle 2). Bei der Zuordnung dieser Attribute zu den fünf Overlay-Farben sind neben den im Web gängigen Farbcodierungen auch westeuropäisch kulturspezifische Stereotypen der Bedeutung von Farben berücksichtigt worden (Heinecke 1992; Russo & Boor 1993).

Tabelle 2: Die Farbzuordnung der Link-Overlays von HyperScout II.[171]

	Rot findet häufig für Warnungen und Fehlermeldungen Verwendung. Darüber hinaus steht die Farbe für „Stopp" (z. B. Ampel) oder Wärme. In *HyperScout II* symbolisiert Rot *„fehlerhafte" Links*: Alle Verweise, die aus irgendeinem Grunde zu keinem Zielobjekt führen, werden entsprechend markiert.
	Gelb wird oft als Hinweis- und Warnfarbe verwendet. In *HyperScout II* weist Gelb auf „besondere" Link-Ziele hin. Hierzu zählen Objekte mit anderem *MIME-Type* als HTML (da zur Anzeige oft eine zusätzliche Anwendung benötigt wird) und Links mit anderen *Protokollen* als HTTP, wie HTTPS, FTP und E-Mail-Links.
	Violett sieht man in Browsern zumeist als Textfarbe für Links zu kürzlich besuchten Seiten. Violett wird daher in HyperScout II ebenfalls für Links zu Seiten verwendet, die der Benutzer früher bereits aufgerufen hat. Gegenüber normalen Webbrowsern kann *HyperScout II* auch grafische Link-Anker kennzeichnen und hebt Links auch bei geänderten Hintergrund-, Link- und Textfarben eindeutig hervor.

[169] Die Overlay-Farben sollten den Helligkeitskontrast und damit die Lesbarkeit des darunterliegenden Textes möglichst wenig einschränken. Ein türkises Hellblau weist gegenüber dunklem Blau den Vorteil auf, dass es aufgrund des Grünanteils ähnlich hell wie die vier anderen Farben erscheint.

[170] Bei der Entwicklung des Prototyps wurde keine spezifische Rücksicht auf Sehschwächen wie die Rot-Grün-Blindheit genommen; dies würde die eindeutig unterscheidbaren Farben vermutlich weiter einschränken.

[171] Die Farben dieser Definition sind nach Relevanz sortiert. Beispielsweise wird ein fehlerhafter Link immer Rot hervorgehoben, auch wenn er zu einer anderen Website verweist.



Grün wird in unserem Kulturkreis oft für positive Meldungen verwendet und steht für Verfügbarkeit, Wachstum und Sicherheit. Man findet Grün aber auch als Hinweis zum „Losfahren" (Ampel) und für Notausgänge (siehe Abbildung rechts). Diese Farbe wurde daher für *externe Links* gewählt, als Metapher für „Ausgang der Site".

Hellblau wird für alle Links verwendet, auf die keine der vorherigen Eigenschaften zutrifft. Die Hervorhebung aller Links stellt sicher, dass dem Web-Nutzer ein konsistentes Link-Interface geboten wird, das die eindeutige Identifikation aller Link-Anker in Webseiten erlaubt (Obendorf & Weinreich 2003).

Die technische Realisierung der Overlays erfolgte ebenfalls auf Basis des Scone-Frameworks, das JavaScript- und CSS-Dateien in alle Webseiten einfügte. Das JavaScript-Programm überwacht den Status der STRG-Taste mittels des Event-Handlers „`window.event.ctrlKey`". Bei Tastendruck werden alle Link-Anker im Dokument gekennzeichnet, indem ihnen eine der CSS-Klassen mit den Overlay-Farben zugewiesen wird. Die entsprechende Farbkategorie für das Overlay liest das Programm aus dem zusätzlichen Anker-Attribut „`markerType`" aus, das der Intermediary während der Übertragung in den Code der Seite einbettet. Darüber hinaus werden diese Attribute über das HyperScout-Applet und den RAS-Dienst von Scone aktualisiert, während sich der Benutzer die Seite ansieht (s. Abschnitt 8.6.5.1).

Im Rahmen der Benutzbarkeitsevaluation von *HyperScout II* sollte überprüft werden, ob den Benutzern diese Art der Link-Auszeichnung zusagt und die Farben und Zuordnungen nachvollziehbar und hilfreich sind. Darüber hinaus wurde der Einsatz der Tastatur zur Aktivierung der Overlays evaluiert (s. Abschnitt 6.5.1).

6.4.2 Multiple Maus-Icons für das Web

Als zweite Detailstufe zur Darstellung von Link-Informationen wurde für *HyperScout II* eine neue Variante adaptiver Mauszeiger (s. Abschnitt 5.3.6) entwickelt, die den Namen „*multiple Maus-Icons*" erhielt. Dabei handelt es sich um eines oder mehrere Icons, die rechts oberhalb des Mauszeigers erscheinen, sobald der Benutzer ihn über einen Link-Anker führt (s. Abb. 99 auf S. 217). Die Icons werden simultan zum Zeiger bewegt, wodurch beides als eine Einheit erscheint.

Die Maus-Icons sollten dem scannenden Verhalten von Web-Nutzern besser gerecht werden als die Tooltips: Sie erscheinen ohne (nennenswerte) Verzögerung beim Mauszeiger, und ihre charakteristische Form sollte Link-Informationen schneller vermitteln als entsprechende Texte. Zudem belegen sie kaum Platz auf dem Bildschirm und verdecken in der Regel keine wesentlichen Teile der Webseite.

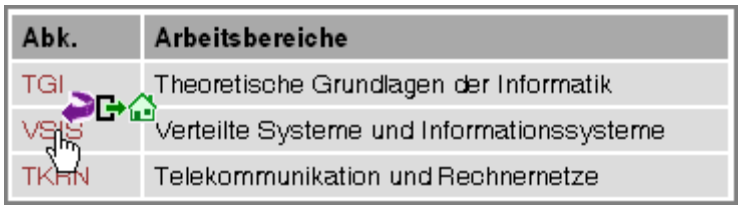


Abb. 99: Beispiel der multiplen Maus-Icons mit drei Icons beim Mauszeiger.

Da die Tooltip-Icons von *HyperScout I* mehrfach als missverständlich und untereinander zu ähnlich kritisiert wurden (s. Abschnitt 6.2.2 und Witt & Tyerman 2002), wurde für die Maus-Icons und die Tooltips von *HyperScout II* mithilfe von partizipativen Methoden ein neuer Satz von Icons entwickelt.

6.4.2.1 Erfahrungen mit partizipativen Verfahren zur Entwicklung von Link-Icons

Als inspirative Quellen für die neuen *HyperScout*-Icons dienten „Das Icon Buch“ (Horton 1994), Icons verschiedener aktueller Programme mit grafischer Oberfläche und Icon-Bibliotheken von Open-Source-Projekten.[172]

Für jede der laut der vorherigen Studien 19 wichtigsten Link-Eigenschaften, die sich symbolisch repräsentieren ließen, wurden 4 bis 15 aus Sicht des Autors möglicherweise geeignete Icons ausgewählt. Diese insgesamt 115 Icons wurden in Größe, Stil und Farbpalette angepasst, um eine Vorauswahl vergleichbarer Icons zu erhalten. Sie dienten als Grundlage für eine partizipative Studie, die sich an eine von Nielsen vorgeschlagene Methode (Nielsen & Sano 1995) und die Testmethode der ISO für grafische Symbole (ISO9186 2001) anlehnt sowie die Erfahrungen des Autors bei einem ähnlichen Projekt berücksichtigt (Weinreich 1997).

Sieben im Umgang mit dem Web erfahrene Personen nahmen nacheinander an der Studie teil. Mit jedem Teilnehmer wurden für jedes der 19 Link-Attribute die folgenden drei Schritte durchgeführt:

- Im ersten Schritt wurde den Teilnehmern die Icon-Vorauswahl für das jeweilige Attribut präsentiert, und sie sollten sagen, welches Link-Attribut die Icons ihres Erachtens jeweils (oder auch gemeinsam) repräsentierten. Auf diese Weise konnte ermittelt werden, ob die Personen einige der Icons bereits mit der gewünschten Bedeutung assoziierten.
- Danach erfuhren die Teilnehmer, welches Konzept eigentlich repräsentiert werden sollte. Sie konnten nun die Icons auswählen, die ihrer Ansicht nach am passendsten waren.
- Drittens bekamen sie die Möglichkeit, eigene Vorschläge für eine grafische Repräsentation des Link-Attributs abzugeben, sofern sie keines der Icons bereits als überzeugend angesehen hatten.[173]

---

[172] Beispiele sind das *KDE-Projekt* (http://kde.org/), das *Tango Desktop Project* (http://tango.freedesktop.org/) und die *QBullet-Website-Icons* (http://www.matterform.com/qbullets/).

[173] Ein ähnliches Vorgehen wurde 2009 von (Wolff & Götzfried 2009) vorgestellt und empfohlen.

Ein Beispiel zur partizipativen Entwicklung der Icons

Zur Illustration werden hier die Ergebnisse der Tests zum Attribut „fehlerhafter Link“ zusammengefasst.



Abb. 100: 15 Vorschläge für ein Icon für fehlerhafte Links im Web.

Zunächst wurden jedem Teilnehmer die in Abb. 100 sehenden 15 Icons vorgelegt. Auf die Frage, für welches Link-Attribut die Icons stehen könnten, gab es zahlreiche Annahmen. Dazu gehörte neben dem vorgesehenen „fehlerhafter Link“ (oder auch „Seite gelöscht“) Attribute wie „defekte Seite“, „Seite unbekannt“, eine „Warnung vor schädlichen Links“ und dass „die Seite gerade überarbeitet werde“.

Im zweiten Schritt sollten die Teilnehmer das oder die ihres Erachtens intuitivsten Icons zur Repräsentation *fehlerhafter Links* heraussuchen. Vier der sieben Teilnehmer bevorzugten von den in Abb. 100 dargestellten Icons das zweite von rechts. Einige Teilnehmer kritisierten, dass das Symbol „zu sehr nach Hochspannung“ aussehe, „künstlerisch nicht ansprechend genug“ oder „zu fett“ sei. Diese Äußerungen wurden für die Gestaltung des endgültigen Icons ebenfalls berücksichtigt.

Da jeder der Teilnehmer für diese Link-Eigenschaft bereits wenigstens eines der Icons als geeignet ansah, gab es hier nur einzelne Vorschläge wie eine „zerbrochene Flasche“ oder eine „Bombe“ als Icon für fehlerhafte Links.

Diese Prozedur wurde danach für die restlichen 18 Attribute wiederholt. Mithilfe dieses Vorgehens ließ sich für fast *zwei Drittel* der 19 Link-Attribute bereits in den ersten zwei Schritten ein mehrheitlich bevorzugtes Icon bestimmen. Für die restlichen Attribute fanden sich keine mehrheitlichen Meinungen, oder es wurde überwiegend keine der Vorgaben als geeignet angesehen. In diesen Fällen kamen die Neuvorschläge der Teilnehmer zum Zug, obgleich die Anregungen für einige der Attribute eher spärlich waren.

Nach Festlegung der Icons sind mit Kollegen und Bekannten mehrere Pretests der multiplen Maus-Icons von *HyperScout II* durchgeführt worden. Dabei zeigte sich, dass Maus-Icons über eine intuitive Symbolik hinaus folgende Eigenschaften aufweisen sollten:

- Sie benötigen einen *starken Kontrast* und intensive Farben, damit sie auch vor bunten und kontrastreichen Webseiten erkennbar sind.
- Sie sollten eine dünne *weiße Umrandung* aufweisen, sodass sie sich auch vor dunklem Hintergrund abheben.
- *Transluzente* (durchscheinende) Maus-Icons sind für viele Webseiten *ungeeignet*, da sie bereits bei geringem Durchscheinen des Hintergrundes (ab ca. 15 % Transparenz) deutlich schwerer zu erkennen waren oder sogar übersehen wurden.

- Im Pretest war eines der Icons animiert: Es wies mit drei nacheinander erscheinenden Punkten auf weitere inhaltliche Informationen im Tooltip hin. Das *animierte Icon wurde* von allen Teilnehmern *als störend empfunden* und daher ersetzt (s. Tabelle 3 unten).
- Alle Teilnehmer sahen die *Nähe der Icons zum Mauszeiger* für die Wahrnehmbarkeit als sehr bedeutend an. Der Abstand wurde aus diesem Grunde reduziert (s. Abb. 101).

Die Ergebnisse der Studie und der Pretests sind die in Tabelle 3 präsentierten 19 Icons. Sie werden in *HyperScout II* sowohl für die multiplen Maus-Icons als auch die Tooltips eingesetzt. Die Farben der Icons stimmen soweit möglich und sinnvoll mit den Overlay-Farben der entsprechenden Kategorie überein (s. Abschnitt 6.5.1).

Tabelle 3: Die partizipatorisch ermittelten Icons von HyperScout II

	Der *rote* Pfeil warnt vor fehlerhaften Hyperlinks. Hierzu gehören Links auf nicht (mehr) vorhandene Ressourcen („Fehler 404“) und zu Servern, die nicht antworten.
	Der *gelbe* Schlüssel weist auf eine per https gesicherte Verbindung hin.
	Der *gelbe* Brief symbolisiert einen E-Mail-Link.
	Die *gelbe* Diskette steht für Hyperlinks, die zu einem Download führen. Dies kann ein anderer Dateityp als HTML oder ein Dateitransfer per ftp sein.
	Der fast zum Kreis geschlossene *violette* Pfeil symbolisiert, dass der Link zum aktuell dargestellten Dokument verweist und somit sozusagen „im Kreise führt“.
	Der rückwärts gerichtete *violette* Pfeil teilt dem Benutzer mit, dass er das Link-Ziel früher bereits besucht hat.
	Referenz-Links: Die *blauen* Pfeile zeigen an, dass der Browser nach Auswahl des Links innerhalb der Seite nach unten bzw. oben scrollt.
	Das *blaue* Haus kennzeichnet einen Link zur Homepage der aktuellen Site.
	Das *grüne* Haus steht für einen *externen Link* zur Homepage einer anderen Site.
	Das *grüne* Ausgangssymbol steht für alle anderen externen Links.
	Für das Link-Ziel wird ein neues Browser-Fenster geöffnet.
	Die Karteikarten stehen für einen Verweis zu einer Navigationsseite mit überdurchschnittlich vielen internen oder externen Links.
	NEU weist auf eine Seite hin, die im letzten Monat überarbeitet wurde.
	ALT steht für Dokumente, die über ein Jahr alt sind.
	Das XY im Oval symbolisiert ein (unbekanntes) Auto-Landeskennzeichen und weist auf ein Dokument in einer fremden Sprache hin.
	Der Elefant steht für große Dokumente von über 100kBytes.
	Die Schnecke warnt vor langsamen Servern, die bei einer Test-Anfrage erst nach über 5 Sekunden geantwortet haben.
	Die drei Punkte zeigen dem Benutzer, dass er im Tooltip weitere inhaltliche Informationen zum Zielobjekt findet.

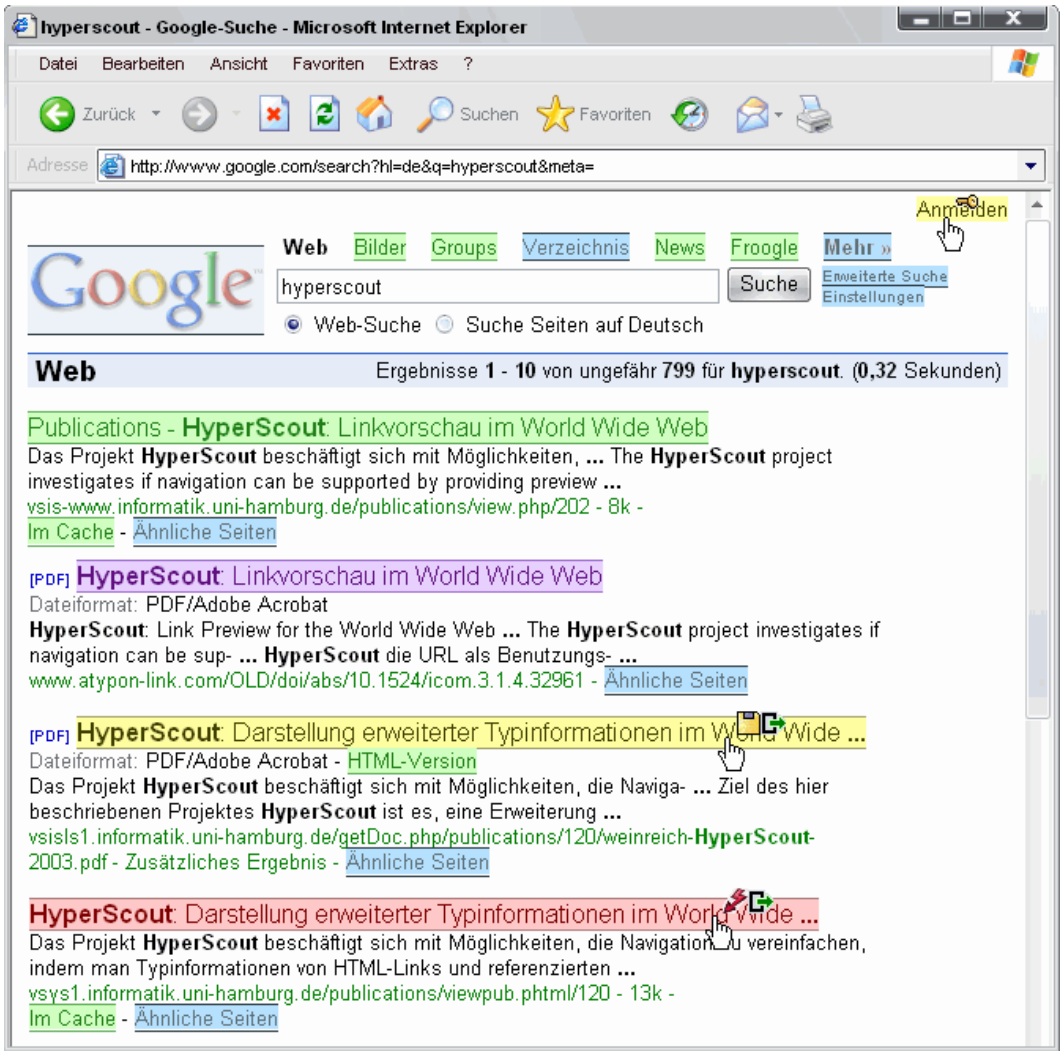


Abb. 101: Drei Beispiele für die Maus-Icons von HyperScout II bei aktivierten Overlays.

Die multiplen Maus-Icons von HyperScout II wurden ebenfalls mit dem Scone-Framework realisiert, das zusätzliche JavaScript- und CSS-Dateien in die Webseiten einbindet. Zu allen Anker-Elementen werden bei der Übertragung Maus-Events hinzugefügt, die dafür sorgen, dass ein „`DIV`"-Element mit den Icons neben dem Mauszeiger erscheint, sobald sich der Zeiger über einem Link-Anker befindet. Beim Bewegen der Maus wird das DIV-Element simultan zum Zeiger bewegt. Innerhalb des DIVs erscheinen entsprechend den Link-Eigenschaften eines bis maximal *drei* der Icons; die Anzahl wurde beschränkt, um die Benutzer nicht zu überfordern. Die darzustellenden Attribute werden zum einen aus dem zusätzlichen, ebenfalls von Scone eingefügten Anker-Attribut „`mouseicons`" ausgelesen. Falls bei der Übertragung der Seite noch keine Link-Daten vorliegen, werden sie dynamisch über das eingebundene HyperScout Applet und Scones RAS-Dienst vom HyperScout-Plug-in abgerufen (vergleichbar mit der Technik zur Bereitstellung der Tooltip-Daten, s. Abschnitt 5.5.3).

In der Evaluation sollten die Nützlichkeit und Akzeptanz dieser Methode sowie die Verständlichkeit der Maus-Icons überprüft werden (s. Abschnitt 6.5.2).

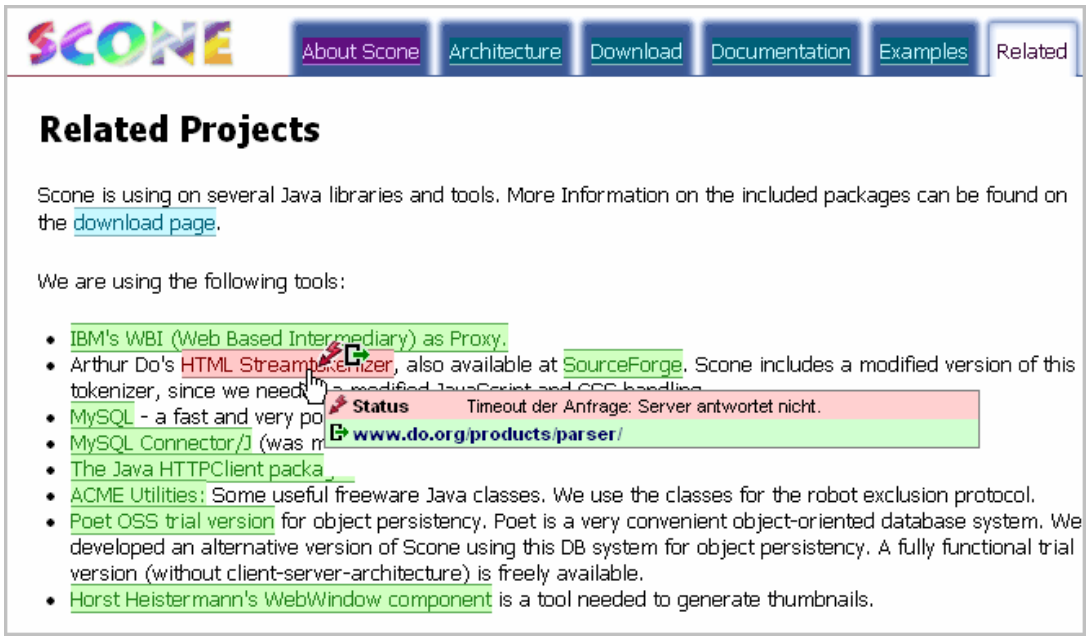


Abb. 102: Details zu fehlerhaften Links erscheinen bei Bedarf im Tooltip.

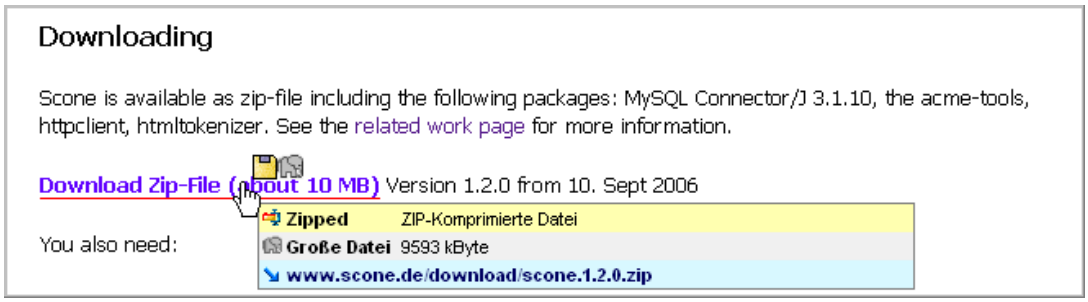


Abb. 103: Link zu einem Download einer größeren ZIP-Datei.

6.4.3 Optimierung der Link-Tooltips für HyperScout II

Die Tooltips von HyperScout II wurden gegenüber der Vorversion in mehreren Aspekten überarbeitet. Um der Kritik an der Konsistenz der Hintergrundfarben zu begegnen, verwendet der neue Prototyp die fünf Farben der Link-Overlays auch in den Tooltips, wobei die Farben derselben Kategorien (rot für Fehler, grün für externe Links etc.) eingesetzt werden. Zusätzlich wird weißer und grauer Hintergrund für inhaltliche Informationen zum Link-Ziel genutzt.

Die ebenfalls kritisierten Tooltip-Icons des ersten Prototyps wurden durch die im vorherigen Abschnitt beschriebenen Icons ersetzt. Aus Platzgründen sind sie im Tooltip etwas kleiner als beim Mauszeiger (12×12 statt 16×16 Pixel). Neben den Maus-Icons gibt es zusätzliche Icons für inhaltliche Informationen im Tooltip, beispielsweise den *Titel* oder die *Beschreibung* einer Seite. Darüber hinaus erscheint im Tooltip bei Links auf andere Dateitypen als HTML das entsprechende System-Icon, und fremde Sprachen werden mittels der passenden Landesflagge repräsentiert. Diese zusätzlichen Symbole sind nicht für die Maus-Icons eingesetzt worden, um dort die Anzahl der zu erlernenden Icons zu reduzieren.

Einzelne Link-Attribute haben gemäß des Benutzer-Feedbacks aus der Evaluation von HyperScout I neue Bezeichnungen erhalten. Die Anzahl der Gruppen wurde reduziert, da die vorherige Kategorisierung den Teilnehmern Probleme bereitete und zahlreiche zuvor als unbedeutend klassifizierte Informationen in HyperScout II nicht mehr berücksichtigt werden. Es gibt als Konsequenz jetzt nur noch *acht* statt *elf* Gruppen von Link-Informationen, von denen zumeist aber – je nach Link-Eigenschaften – nur zwei bis vier relevant sind und im Tooltip erscheinen.

Folgende Informationen findet man in den Tooltips von HyperScout II:

1. Die Verfügbarkeit des Link-Ziels

Zuoberst erscheinen mögliche *Fehlermeldungen* beim Zugriff auf das Link-Ziel. Sie werden durch eine *rote* Hintergrundfarbe hervorgehoben (s. Abb. 102).

2. Dateityp und Protokoll

Hinweise auf *besondere Link-Ziele* folgen und werden gemäß den Overlays *gelb* hinterlegt. Zu dieser Gruppe gehören andere Dateitypen als HTML (Abb. 103) und Links mit anderen Protokollen als HTTP (z. B. FTP-Downloads und E-Mail-Adressen). Gesicherte Verbindungen per HTTPS und durch Passwort geschützte Seiten sind nun auch in dieser Gruppe zugeordnet.

3. Inhaltliche Angaben zum Zieldokument

Als Nächstes folgen *inhaltliche Angaben* zum Link-Ziel vor *weißem* Hintergrund. Sie sind somit für die meisten Links zu „normalen" Webseiten oben im Tooltip zu finden, sofern keine der obigen Fehler- oder Hinweismeldungen zutrifft. Dies wird der hohen Benutzer-Präferenz gerecht, die inhaltliche Informationen in der Umfrage erhielten, und berücksichtigt dem Umstand, dass keine äquivalente Darstellung solcher Daten als Overlay-Farbe oder Maus-Icon möglich ist.

Der Algorithmus zur Erstellung der inhaltlichen Zusammenfassung wurde überarbeitet, beispielsweise geschieht die Ausgabe jetzt formatiert, sodass Auflistungen aus der Webseite jetzt ebenfalls im Tooltip zeilenweise erscheinen. Zudem vergleicht das System den Text des Link-Ankers mit Titel und Beschreibung der Ziel-Seite, um Redundanz zu vermeiden.

4. Weitere inhaltliche Metadaten zum Link-Ziel

Als Viertes folgt eine Gruppe *„weiterer" inhaltlicher Informationen* zum Ziel des Links, die in einem hellgrauen Bereich erscheint. Hierzu gehört das *Datum* der letzten Aktualisierung (s. Abb. 104) und die *Sprache* des Zieldokuments, wobei wie bei HyperScout I eine Landesflagge als Icon verwendet wird. Ebenfalls erhält der Benutzer hier Hinweise auf mögliche Verzögerungen beim Zugriff auf die Ressource, wie die Größe der Datei (s. Abb. 103) und eine

langsame Antwort des Servers. Darüber hinaus werden Seiten ohne Links sowie Navigationsseiten mit vielen internen und externen Links kenntlich gemacht (s. Abb. 106).

5. Die frühere Benutzung des Ziel-Objekts

Es folgen vor *violettem* Hintergrund Daten zu *früheren Zugriffen des Benutzers* auf das Zielobjekt. Vergleichbar mit dem ersten Prototyp wird hier angezeigt, wann und wie oft der Anwender die Seite bereits besucht hat (s. Abb. 104).

6. Die Link-Aktion

Als Sechstes erscheinen vor hellblauem Hintergrund Angaben zur *Link-Aktion* (und dem URI, siehe unten). Zu dieser Gruppe gehören in HyperScout II aufgrund der Studienergebnisse von HyperScout I nur noch zwei Eigenschaften: Das Öffnen des Link-Ziels in einem neuen Fenster (s. Abb. 105) und Referenz-Links, bei denen der Browser auf einer Seite nach unten oder oben scrollt. Ist das Ziel des Referenz-Links innerhalb der gerade dargestellten Seite, so steht diese Information alleine im Tooltip (s. Abb. 107).

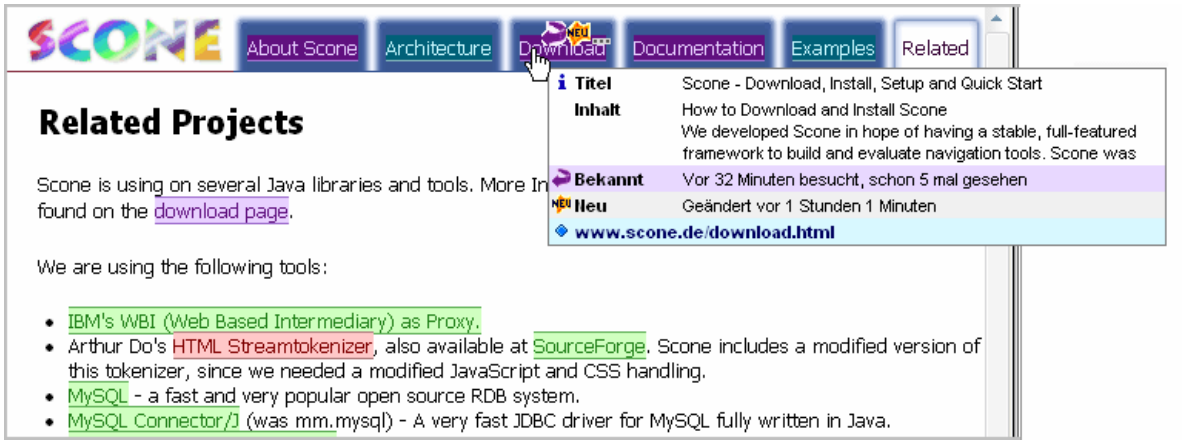


Abb. 104: Der Link führt zu einer bekannten Seite, die kurz zuvor überarbeitet wurde.

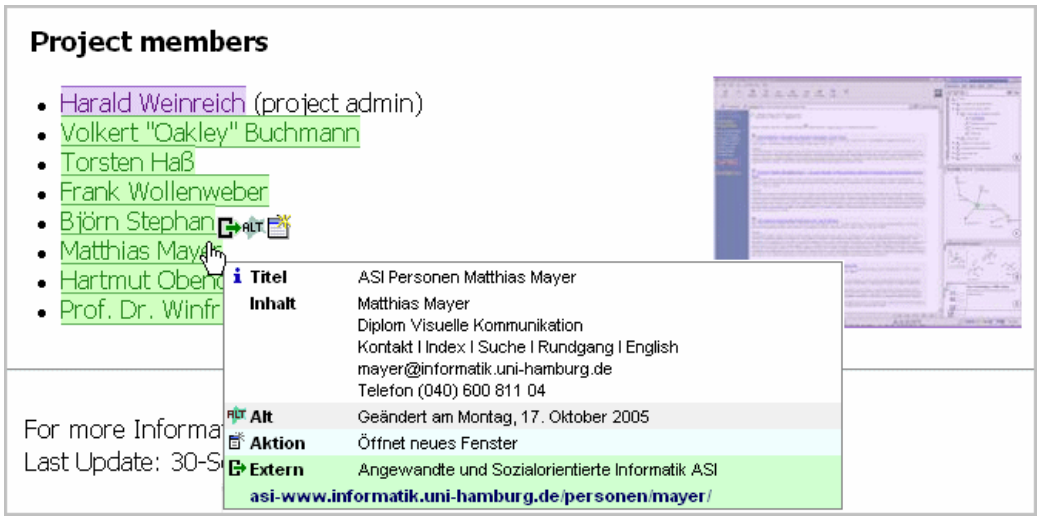


Abb. 105: Ein externer Link zu einem älteren Dokument, das in einem neuen Fenster erscheint.



Abb. 106: Tooltip mit Titel und Beschreibung. Die Zielseite hat sehr viele Links.



Abb. 107: Ein Reference-Link innerhalb der Webseite.

7. Die Verteilungscharakteristik des Links

Informationen zu *externen Links* findet man als siebente Gruppe vor *grünem* Hintergrund. Ein Link wird als „extern“ ausgelegt, wenn sich *First-* und *Second-Level Domain* von Ausgangs- und Zieldokument unterscheiden;[174] weitere Unterschiede im Domain-Namen werden nicht berücksichtigt. Dies trägt der Kritik Rechnung, dass Verweise zwischen den Websites einer Organisation nicht als extern gekennzeichnet werden sollten (s. Abschnitt 6.2.3.2).

Bei der Anzeige wird zudem zwischen Links zur Homepage der externen Site und „Deep Links“ unterschieden, die in die Site hinein verweisen. Bei letzteren erscheint im Tooltip zusätzlich eine aus dem Titel der Homepage generierte *Beschreibung* der Site (s. Abb. 105).

8. Der URI

Zuunterst steht im Tooltip der *URI* des Link-Zieles in einer Zeile. Aus Gründen der Lesbarkeit und um Platz zu sparen, wird der Protokollteil weggelassen, sofern es sich um den Standard „`http://`“ handelt. Als Hintergrundfarbe wird *hellblau* oder *grün* verwendet, je nachdem ob es sich um einen internen oder externen Link handelt.

Bei der Evaluation sollte ermittelt werden, wie die Tooltips zusammen mit den beiden anderen Erweiterungen der Link-Benutzungsschnittstelle genutzt werden und ob die überarbeitete Darstellung die Defizite von HyperScout I verringern (s. Abschnitt 6.5.3 ff).

6.5 Empirische Evaluation von HyperScout II

Die Evaluation von *HyperScout II* erfolgte zwischen November 2004 und Februar 2005 mit derselben Evaluationsmethode wie bei *HyperScout I* (siehe unten). Bei dieser Studie nahmen

[174] Beispielsweise ist hiernach ein Link von einer Seite auf www.uni-hamburg.de zu www.uni-bremen.de extern, ein Link von www.uni-hamburg.de zu www.informatik.uni-hamburg.de aber nicht.

auch weniger technisch versierte Nutzer als beim Test von HyperScout I teil, um ebenfalls Einblick in die Probleme dieser Benutzergruppe zu erhalten. Sie sollten als Vorbedingungen lediglich eine „Interneterfahrung“ von einem Jahr und eine durchschnittliche Mindestnutzung des Webs von einer Sitzung pro Woche aufweisen.

Die 13 Teilnehmer wurden im erweiterten Bekanntenkreis des Autors akquiriert. Elf von ihnen waren männlich, zwei weiblich. Das Alter lag zwischen 28 und 39 Jahren (Durchschnitt 33,2 Jahre, Standardabweichung 3,1 Jahre). Sechs Teilnehmer kamen aus dem IT-Bereich und bezeichneten sich selbst als „sehr erfahren“ mit dem Web und nutzten es nahezu täglich, sieben Teilnehmer hatten kein professionelles IT-Hintergrundwissen, nutzten das Web aber mindestens seit zwei Jahren und einmal bis mehrfach die Woche.

Bei der Evaluation von *HyperScout II* wurde wie bei der ersten Studie eine angepasste Version der Thinking-Aloud-Methode eingesetzt (s. Abschnitt 6.1.4): Die Teilnehmer erhielten anfangs eine Einführung in die Konzepte des Systems und konnten es anhand einer ihnen vertrauten Website erproben. Zudem wurde ihnen die Bedeutung der fünf Overlay-Farben und der wichtigsten Icons erklärt.

Ähnlich wie bei der Evaluation von *HyperScout I* waren sie angehalten, die Browser-Erweiterung aktiv zu nutzen, wobei nun alle drei Methoden berücksichtigt werden sollten, je nachdem welche ihnen gerade angemessen erschien. Während des Tests wurden die Teilnehmer an die Nutzung der Erweiterungen erinnert, sofern sie sie längere Zeit unberücksichtigt ließen. Dieses Vorgehen bewährte sich auch dieses Mal.

Die Tests dauerten zwischen 55 und 95 Minuten. Um eine größere Anzahl von Teilnehmern einbeziehen zu können, wurden einige der Tests bei ihnen zu Hause durchgeführt. Zum Einsatz kamen ein zeitgemäßer Laptop und ein PC. Sie boten einen 15" LCD- bzw. einen 17" CRT-Bildschirm mit einer Mindestauflösung von 1024×768 Pixel. Die Geräte waren jeweils über ADSL mit 2 MBit/s oder schneller an das Internet angebunden. Das *UserTestTool* (s. Abschnitt 8.6.5.7) lief dieses Mal auf demselben Bildschirm wie der Browser und wurde während des Bearbeitens von Aufgaben hinter das Browser-Fenster gelegt. Die Tests sind wieder vom Testleiter schriftlich protokolliert, sowie auf MiniDisc aufgezeichnet und nach dem Test transkribiert worden.

In Folgenden werden die wichtigsten Ergebnisse der Studie präsentiert. Sie sind thematisch nach Art der Erweiterung gruppiert: Zuerst wird auf die *Link-Overlays* eingegangen (s. Abschnitt 6.5.1), gefolgt von den *multiplen Maus-Icons* (s. Abschnitt 6.5.2) und neuen Erkenntnissen im Umgang mit den *Tooltips* von HyperScout II (s. Abschnitt 6.5.3). Abschließend folgen ein Vergleich der drei Methoden und ein Fazit dieser zweiten partizipativen Studie.

Bereits die ersten der dreizehn Tests wiesen recht eindeutig auf zwei kleinere, dennoch störende Defizite des Systems hin. Da eine entsprechende Anpassung des Prototyps einfach und vielversprechend erschien und die Hoffnung bestand, dass durch diese Optimierungen die Effizienz der folgenden Tests gesteigert werden könnte, wurden nach dem *fünften* Test

die Präsentation der Link-Overlays und die Bezeichnung eines Attributs im Tooltip überarbeitet. Diese Änderungen und ihre Auswirkungen sind in den folgenden Studienergebnissen mit aufgeführt.

6.5.1 Ergebnisse der Evaluation zu den Link-Overlays-on-Demand

Das Verhalten und die Kommentare der Teilnehmer in Bezug auf die Link-Overlays wiesen zahlreiche individuelle Unterschiede und eine Gemeinsamkeit auf: Alle Teilnehmer waren am Ende des Tests der Ansicht, dass grundsätzlich eine verlässliche Methode zur eindeutigen Identifikation aller Links in Webseiten oft hilfreich wäre (s. Abschnitt 6.5.1.1).

Die große Mehrheit der Teilnehmer (11 der 13 Personen) verwendete die *Overlays-on-Demand* von sich aus regelmäßig während der Tests und befand sie für „hilfreich" bis „sehr nützlich", um Links eindeutig auf Webseiten zu identifizieren. Zwei Teilnehmer drückten während des Tests sogar auf fast jeder Seite zuerst die STRG-Taste, um die Overlays einzublenden und sich „einen Überblick zu verschaffen", wie sie sagten.

Zwei der dreizehn Benutzer empfanden die Overlays in der jetzigen Form als ungeeignet für ihren Umgang mit dem Web, da sie die *Tastatur* nicht gerne zum Einschalten der Overlays verwenden würden. Sie bevorzugten stattdessen die Maus-Icons und die Tooltips, da sie keine zusätzliche Taste benötigten und zudem mehr Details vermitteln würden. Den Einsatz der Tastatur fand noch ein dritter Teilnehmer störend, der ansonsten mit den Overlays sehr zufrieden war. Als Gründe gaben sie an, dass sie „generell ungern die Tastatur nutzen" oder „die linke Hand frei behalten möchten", beispielsweise „um nebenbei zu rauchen". Die übrigen 10 Teilnehmer sahen die Tastatur als eine geeignete Methode für die Aktivierung der Overlays an, wobei drei von ihnen die gleichzeitige Nutzung von Tastatur und Maus bereits gewohnt waren, unter anderem von Adobe Photoshop, Microsoft Excel und dem Webbrowser Opera.

Zwei Teilnehmer wünschten sich „einrastende" Overlays, um die STRG-Taste nicht für längere Zeit drücken zu müssen. Ein Vorschlag war der Einsatz der CAPS-Lock-Taste (Feststelltaste für Großschrift). Von mehreren Teilnehmern kam die Anregung, alternativ die Overlays (auch) über ein Icon in der Toolbar des Browsers aktivieren zu können und so die Tastatur zu umgehen.

6.5.1.1 Die Bedeutung von Link-Overlays für die Identifikation von Link-Ankern

Eine besondere Stärke von Hyperscout II war nach Ansicht aller Teilnehmer, dass man Links unter Mithilfe der Overlays schnell und eindeutig identifizieren könne. Drei Teilnehmer äußerten noch zu Beginn des Tests Zweifel an der Nützlichkeit einer solchen Erweiterung, da ihres Erachtens auf den meisten Seiten alle Links eindeutig identifizierbar seien. Diese Meinung revidierten sie jedoch im Verlauf des Tests, da sie mehrfach (vor allem grafische) Links mithilfe der Overlays entdeckten, die sie zuvor nicht wahrgenommen hatten. Viele Teilnehmer waren insbesondere überrascht, wenn sie durch die Overlays auf ihnen ver-

trauten Seiten neue Links entdeckten; so war z. B. das große Google-Logo auf der Homepage der Suchmaschine während einiger Tests mit weiteren Informationen verknüpft.

Weiterhin wurde gelobt, dass die Overlays „das Finden der Links einer Seite vereinfachen", „das Annavigieren der Links weniger anstrengend" sei und die Links „eine höhere Salienz" bekämen, sie also *einen stärkeren Reiz auf das Bewusstsein* ausübten und ihre Wahrnehmung eine geringere Konzentration erfordere.

Mehrere Teilnehmer hoben hervor, dass man ebenfalls einfach überprüfen könne, was *keine* Links seien. Ein Teilnehmer hielt beispielsweise die farbigen Überschriften einer Seite für Hyperlinks und erkannte dann anhand der Overlays, dass diese Annahme falsch war.

Kritisiert wurde, dass manche Navigationsseiten mit sehr vielen Links „überladen" seien, wenn die Overlays fast alles hervorheben. Ein Teilnehmer erwartete, dass die Overlays auch in einem PDF-Dokument funktionierten, das er während des Tests im Browser geöffnet hatte (dies war nicht gegeben).

6.5.1.2 Die Eignung von Farben zur Visualisierung von Link-Attributen

Den Einsatz mehrerer Farben für die Overlays fanden zehn der 13 Teilnehmer gut, da es „die Übersichtlichkeit der Seite erhöhe" und das „Scannen nach bestimmten Informationen erleichtere". Drei hielten die farbliche Unterscheidung hingegen nicht für bedeutsam. Einer von ihnen war den Overlays gegenüber grundsätzlich skeptisch eingestellt, zumal er die Tastatur beim Browsen nicht gerne verwendete, der zweite würde die Link-Informationen bei Tastendruck lieber als transluzente Icons *vor* oder *hinter* allen Link-Ankern eingeblendet bekommen, und der dritte meinte, es würde ausreichen, alle Links mit derselben Farbe hervorzuheben – die Unterscheidung sei lediglich eine „nette Beigabe" und aufgrund der Maus-Icons und Tooltips überflüssig.

Während der ersten Tests wurde von allen Teilnehmern mehrfach kritisiert, dass die Overlay-Farben bei Grafiken schwer auseinanderzuhalten seien. Zudem war zu beobachten, dass die Overlays bei dunkleren Grafiken häufiger nicht wahrgenommen wurden. Einer der Teilnehmer schlug daher vor, das Problem mit einem zusätzlichen Rahmen in der Farbe des Overlays um den Link-Marker zu reduzieren.

Nach den ersten fünf Tests wurden daher dünne opake Linien in der Overlay-Farbe ober- und unterhalb des Overlay-Bereiches eingefügt (s. Abb. 108). Dank dieser Modifikation gab es in den weiteren acht Tests kaum noch Probleme bei der Wahrnehmung von Farben und Ausmaßen der Overlays.

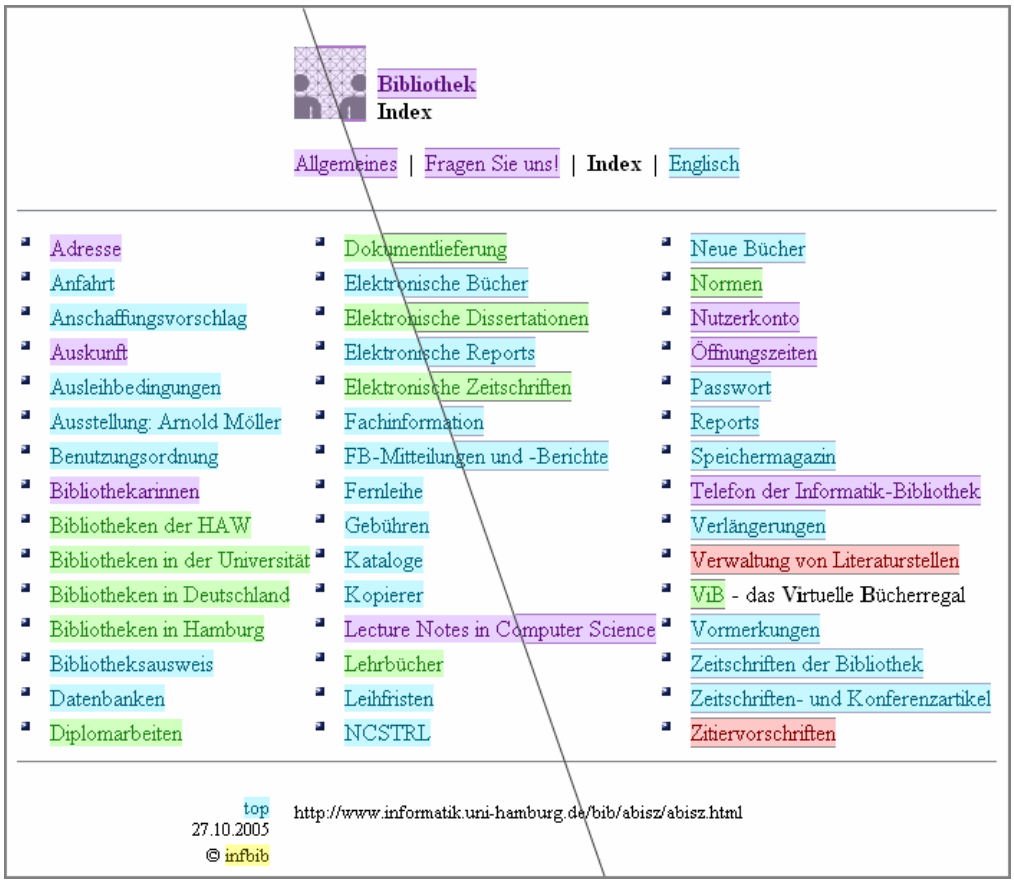


Abb. 108: Die zwei Link-Overlay-Varianten im Vergleich.

Allerdings störten diese Begrenzungslinien in einigen Fällen:

- Beim Überlappen zweier Overlays – dazu kam es unter anderem bei geringem Zeilenabstand – kann ein Overlay die Randlinie eines anderen überdecken (s. Abb. 101).
- Sofern eine Grafik über eine eigene Umrandung verfügt (z. B. bei grafischen Schaltflächen), kann die zusätzliche Linie des Overlays irritieren, da sie wie eine zweite Randlinie erscheint.

Eine mögliche Lösung zur Reduzierung dieser Probleme wären unterschiedliche Overlays für textliche und grafische Links. Da bei Text die Identifikation der Overlay-Farben keine Probleme bereitete, könnte man hier auf opake Bereiche verzichten und stattdessen Overlays mit konzentrischem Intensitätsverlauf verwenden, der die Grenzen sichtbar macht (s. Abb. 109 oben und rechts). Eine alternative Lösung für Grafiken wären kleine opake Dreiecke in den Ecken, die Farbe und Ausmaß des Link-Ankers eindeutig vermitteln (s. Abb. 109, linke Spalte). Diese Methoden wurden aber nicht mehr evaluiert.



Abb. 109: Mockup einer neuen Overlay-Technik für Grafiken und Textanker.

Die Zuordnung der fünf Overlay-Farben zu den ausgewählten Link-Attributen bereitete den Teilnehmern nur anfänglich Probleme. Bereits nach der Hälfte der Testzeit hatten sie die Bedeutung zumeist im Kopf. Dennoch gab etwa ein Drittel der Teilnehmer nach dem Test an, dass es für sie schwierig sei, sich die Bedeutung der Farben zu merken. Auf Nachfrage konnten dennoch alle die Farbzuordnung wiedergeben. Zwei Teilnehmer wünschten eine einblendbare Farblegende im Browser.[175]

Fast die Hälfte der Teilnehmer war mit der Zuordnung der Farben zu den Link-Attributen zufrieden, die anderen äußerten unterschiedliche Kritikpunkte. Einer der Teilnehmer wünschte bis auf Rot und Gelb keine weitere farbliche Unterscheidung der Overlays, da er die anderen Attribute als eher sekundär ansah. Einer empfand fünf Farben „bereits etwas viel", konnte aber nicht angeben, welche Farben er weglassen würde. Zwei hätten hingegen gerne eine farbliche Unterscheidung von „noch mehr Link-Arten", beispielsweise für Links innerhalb der Seite. Vorgeschlagen wurde eine alternative Farbzuordnung weiterer Link-Attribute beim Drücken einer anderen Steuertaste als STRG.

Die Farbzuordnungen sind von den Teilnehmern unterschiedlich aufgenommen worden:

- ***Rot*** scheint als einzige Zuordnung intuitiv verständlich zu sein und wurde von keinem Teilnehmer mehr als einmal nachgefragt. Zwei Personen waren der Ansicht, dass fehlerhafte Links bereits ohne Drücken der Taste rot markiert werden sollten. Lobend hervorgehoben wurde, dass diese Funktion auch bei Ergebnislisten von Suchmaschinen funktioniere und dass auch Web-Autoren so ihre Webseiten schnell auf Korrektheit prüfen könnten. Lediglich ein Teilnehmer merkte kritisch an, dass Rot gerade fehlerhafte und somit „nutzlose" Links am meisten hervorhebe.

[175] Erstaunlicherweise fiel nur drei Teilnehmern auf, dass die Farben im Tooltip mit den Linkfarben korrespondierten und sie somit bereits eine einfache Hilfe-Funktion hatten.

- Die Zuordnung der Farbe ***Gelb*** war mit mehr Schwierigkeiten verbunden als erwartet. Alle Teilnehmer fragten während des Tests mehrfach nach der Bedeutung der Farbe. Hilfreich fanden mehrere Personen, dass so PDF-Dokumente und Downloads schneller zu finden seien, da sie häufiger nach entsprechenden Links auf Seiten suchen würden. Etwa die Hälfte der Teilnehmer kritisierte allerdings, dass die Farbe ihres Erachtens „mehrfach belegt sei", da Links zu gesicherten Verbindungen und E-Mail-Adressen ebenfalls in Gelb erschienen. Zwei Teilnehmer würden eine eigene Farbe nur für PDF-Dateien reservieren, da diese für sie besonders bedeutsam seien.
- ***Violette*** Overlays für Hyperlinks zu bekannten Seiten verursachten kaum Probleme. Die meisten Teilnehmer waren mit der Farbe bereits von ihrem „normalen" Webbrowser vertraut. Die Meinungen über die Relevanz dieser Information für die Navigation variierte aber weitaus stärker als erwartet: Nur die knappe Hälfte der Teilnehmer fand diese Overlay-Farbe bedeutend für die Navigation, die anderen eher „nicht so relevant", da sie beispielsweise „nichts darüber aussage, wie nützlich die Seite vorher für mich war". Offensichtlich wird die Bedeutung von „lila Links" im Web von vielen Teilnehmern nicht als ganz so essenziell eingeschätzt, wie von vielen Experten (u. a. Spool, Scanlon et al. 1998; Siegel 1999; Nielsen 2004a; s. auch Abschnitte 5.1.3 und 5.3.3).
- Die Bedeutung der ***grünen*** Overlays (für externe Links) wurde ähnlich häufig wie die der gelben hinterfragt. Einer der Teilnehmer fand die Zuordnung kontraintuitiv, da er Grün mit „gut, also innerhalb der Site" assoziierte, Blau für ihn hingegen für „einen Ausgang" stünde, da „auch Autobahnausfahrten ein blaues Schild" hätten. Die Meinungen bezüglich der Nützlichkeit einer farblichen Unterscheidung externer Links gingen auseinander. Etwa ein Drittel der Personen fand sie sehr hilfreich, da man so „bewusst innerhalb einer Site bleiben" sowie „externe Links schneller finden" könne. Andere meinten hingegen, dass man für diese Link-Eigenschaft keine eigene Farbe reservieren sollte und dass die Bedeutung sehr von der aktuellen Aufgabe des Anwenders abhänge.
- ***Hellblau*** verursachte keine nennenswerten Probleme.

Alle Teilnehmer bewerteten Rot als die „wichtigste" Overlay-Farbe, die Vorrang vor den anderen haben müsste. Die Meinungen bezüglich der drei Farben Grün, Violett und Gelb waren uneinheitlich, und im Interview gab es für jede von ihnen mindestens zwei Teilnehmer, die sie bedeutsamer als die restlichen Farben bewerteten. Zwei Personen waren der Ansicht, dass die Link-Overlays auch mehrere Farben gleichzeitig zeigen müssten, wenn mehrere der Eigenschaften auf einen Link zuträfen, z. B. grün-violett gestreift für einen *externen* Link zu einer *bekannten* Seite. Vier Teilnehmer wollten selbst konfigurieren können, welche Eigenschaft mit welcher Farbe hervorgehoben wird (s. Abschnitt 6.5.4).

Einige Teilnehmer machten vielversprechende Vorschläge zur Erweiterung der Overlay-on-Demand-Technik: Einer regte an, dass auch das Ziel eines Referenz-Links innerhalb der Seite mit einem Overlay hervorgehoben werden sollte, damit man erkennen könne, wohin der

Link führt.[176] Eine weitere Person wollte die Begriffe der letzten Suchanfrage mittels der Overlays in Seiten hervorheben können, um relevante Teile einfacher zu identifizieren.[177]

6.5.2 Ergebnisse zu den multiplen Maus-Icons

Die Resonanz auf die multiplen Maus-Icons war bei über zwei Dritteln der Teilnehmer (neun der dreizehn Personen) positiv bis sehr positiv. Gelobt wurde vor allem, dass die Icons gut wahrzunehmen seien und viele wichtige Link-Eigenschaften schnell vermitteln. Vier Teilnehmer erachteten die Maus-Icons hingegen als unnötig, da sie diese Informationen ebenfalls im Tooltip fanden: Die Tooltips böten mehr Details, und die Wartezeit bis zu ihrem Erscheinen sei so kurz, dass sie nicht erst auf die Icons achten würden.

Mehrfach äußerten die Teilnehmer, dass sie das Erlernen der Icons als relativ mühsam empfanden, auch wenn man erfreulicherweise ihre Bedeutung immer im Tooltip neben dem Icon finden konnte. Zwei Teilnehmer gingen anfänglich davon aus, dass die Icons anklickbar wären; nach einer kurzen Erläuterung stellte dies aber keine weitere Schwierigkeit dar.

Die gleichzeitige Darstellung mehrerer Icons neben dem Mauszeiger wurde mit größerer Selbstverständlichkeit akzeptiert, als vom Autor der Studie erwartet. Die überwiegende Meinung war, dass mehrere Icons kein grundsätzliches Problem seien. Vier Teilnehmer waren der Ansicht, dass auch mehr als drei Icons kein Problem für sie darstellen würde, und zwei wollten grundsätzlich alle Icons aus dem Tooltip auch als Maus-Icons sehen. Lediglich ein Teilnehmer beanstandete, dass er eine recht lange Zeit benötigen würde, um drei oder mehr Icons wahrzunehmen und er dann „auch gleich den Tooltip beachten könnte".

6.5.2.1 Zur Gestaltung von Icons für die Darstellung von Link-Attributen

Zur Gestaltung der Icons gab es wesentlich mehr positive als kritische Äußerungen. Gelobt wurde, dass sich die Icons sehr gut von den meisten Webseiten abheben und schnell wahrnehmbar seien. Lediglich drei Teilnehmer empfanden die Icons als „zu bunt", und einer war der Ansicht, sie müssten noch deutlicher mit den Farben der Overlays übereinstimmen.

Zwei Teilnehmer äußerten, dass ihnen die Icons zueinander nicht konsistent genug seien, beispielsweise sähen die vier Icons für „altes Dokument", „neues Dokument", „gesicherte Verbindung" und „fremde Sprache" kleiner aus als die übrigen.

Als einprägsam und leicht verständlich wurden durchgängig die Icons für Links zur *Homepage*, zu einem *neuen Dokument*, für *fehlerhafte Verweise* und für *E-Mail-Adressen* angesehen (s. Tabelle 3 auf S. 219). Die meisten der anderen Icons wurden von den Teilnehmern überhaupt nicht oder nur einmal hinterfragt, da sie sich gegebenenfalls im Tooltip selbst über die Bedeutung informierten.

---

[176] Dies ist übrigens ein Browser-Feature, das bereits der Browser Mosaic bot, danach aber leider „abhanden" gekommen ist (Weinreich, Obendorf et al. 2001).

[177] Dies wird bereits als Option vom Browser-Add-On „Google Toolbar" geboten, sofern Google als Suchmaschine für die Internet-Suche eingesetzt wurde.

Es gab auch einige kritische Anmerkungen, die sich insbesondere auf Icons bezogen, die bereits bei der partizipativen Entwicklung Schwierigkeiten bereiteten:

- Das Icon, das als Hinweis auf inhaltliche Informationen im Tooltip diente (drei Punkte), wurde am häufigsten bemängelt. Äußerungen der Teilnehmer waren unter anderem, dass es „nicht aussehe wie ein Icon", „kaum sichtbar, da grau" sei, und dass „es einfach nur störe", da nach seiner Ansicht in jedem Tooltip inhaltliche Informationen zu finden sein sollten. Dies spricht dafür, das Icon entweder wegzulassen oder durch ein gänzlich anderes Symbol zu ersetzen, z. B. ein „i" für inhaltliche Informationen wie im Tooltip.
- Der Begriff „Alt" im Icon für alte Dokumente wurde oft missverstanden. Beispielsweise assoziierten es einige Teilnehmer mit „*Alt*ernative". Zudem wurde angemerkt, dass ein solches Icon das Zieldokument potenziell stigmatisiere. Das Icon sollte überarbeitet werden und sich deaktivieren lassen.
- Das „XY-Symbol" für fremde Sprachen hat kaum einem Teilnehmer zugesagt. Stattdessen wurde vorgeschlagen, wie im Tooltip die Landesflagge als Maus-Icon anzuzeigen.
- Statt der Diskette für alle Datei-Downloads wurde das System-Icon des entsprechenden Dokumenttyps gewünscht, so wie es auch im Tooltip zu finden ist.

Die beiden letzten Punkte zeigen, dass die Beschränkung der Icon-Anzahl kontraproduktiv für die Darstellung von Sprache und Dateityp war, zumal gerade diese Symbole (Landesflaggen und System-Icons der Dateitypen) in den Tooltips als intuitiv angesehen wurden.

Trotz der Kritikpunkte hat sich das partizipative Vorgehen (s. Abschnitt 6.4.2.1) bei der Gestaltung der Icons bewährt, da die meisten Icons gut bis sehr gut ihren Zweck erfüllten.

6.5.3 Ergebnisse zu den optimierten Tooltips

Dank der Überarbeitung der Tooltips trat ein erheblicher Teil der Probleme mit HyperScout I bei der Studie von HyperScout II nicht mehr auf. Die Evaluationsergebnisse zu den Tooltips waren auch bei dieser Studie recht umfangreich, zumal die Tooltips von den drei Darstellungsmethoden die vielfältigsten und detailliertesten Informationen anboten. Einige der Defizite aus dem ersten Test ließen sich allerdings weiterhin beobachten, weil sie noch nicht adäquat überarbeitet wurden oder die gewählten Lösungen die Probleme nicht wie erwartet behoben.

Weiterhin wurden zahlreiche neue Schwächen der Tooltips ermittelt. Dies ist darauf zurückzuführen, dass auch weniger technikvertraute Benutzer an dieser Studie teilnahmen und die Kombination mit den beiden anderen Visualisierungstechniken neue Herausforderungen stellte. Die Studienergebnisse werden im Folgenden wieder (gemäß Kapitel 6.2) den drei Kategorien *Interaktion* mit den Tooltips, *Präsentation* und *Inhalte* der Tooltips zugeordnet.

6.5.3.1 Erkenntnisse zur Interaktion mit Link-Tooltips im Web

Eine entscheidende Schwäche von HyperScout I wurde beseitigt: Da die Teilnehmer bei den ersten Tests häufig nicht auf die Tooltips geachtet hatten, waren ihnen oft auch kritische Link-Informationen (wie eine Warnung vor fehlerhaften Links) entgangen. Mit HyperScout II passierte dies kaum noch, da durch Link-Overlays und Maus-Icons bereits auf solche Probleme hingewiesen wurde. Die Teilnehmer mussten während der zweiten Studie deutlich *seltener* zur Berücksichtigung der Tooltips motiviert werden, da ihnen offenbar Overlays und Icons immer wieder vor Augen führten, dass sie gerade mit einem modifizierten Webbrowser arbeiteten. Zudem sahen mehrere Teilnehmer die Tooltips als willkommene Hilfe zum Erlernen der Maus-Icons an, weil sie die Icons mit ihren Bezeichnungen in den Tooltips wiederfanden.

Von zahlreichen Anwendern wurde es als störend empfunden, dass die Tooltips immer – und damit manchmal auch ungewollt – erschienen, wenn sie die Maus kurz über einen Link hielten. Sieben der dreizehn Teilnehmer würden daher die Tooltips gerne einfach an- und abschalten können, und drei Teilnehmer wünschten sich, dass die Tooltips nur erschienen, wenn sie gleichzeitig die STRG-Taste für die Overlays gedrückt hielten. Einer der drei war der Ansicht, dass man bei gedrückter Taste auf die Zeitverzögerung bis zum Erscheinen der Tooltips verzichten könnte, ein weiterer, dass diese Änderung alle drei Erweiterungen wie ein zusammengehörendes System erscheinen ließ.

Die Teilnehmerwünsche bezüglich der Interaktion mit den Tooltips von HyperScout II ließen sich durch ein einfaches Konfigurationsmenü im Browser realisieren, das die folgenden Optionen böte:

- *„Normal“*: Die Tooltips erscheinen wie bisher mit einer kurzen Verzögerung.
- *„Hotkey“*: Sie erscheinen nur, wenn auch die Taste für die Overlays gedrückt wird.
- *„Deaktiviert“*: Die Tooltips sind als einzige Erweiterung (zeitweise) ausgeschaltet.

6.5.3.2 Zur Präsentation von Link-Informationen in Tooltips

Die Gestaltung der Tooltips von HyperScout II wurde von mehreren Teilnehmern ausdrücklich gelobt. Positive Anmerkungen waren, dass sie einen „konsistenten Eindruck“ hinterließen und die Aufteilung der Tooltips durch die „zurückhaltenden Hintergrundfarben“ hilfreich sei, da so „die Zeilen einfach zu scannen“ und „die Informationen schneller wahrnehmbar“ seien. Die neue Gruppierung der Inhalte und der Einsatz der Hintergrundfarben erwiesen sich somit als vorteilhaft.

Die Bezeichnungen der einzelnen Link-Attribute bereiteten aufgrund der Überarbeitung kaum noch Verständnisprobleme: Lediglich der Begriff „*Wiederbesuch*“ – für Seiten, die ein Teilnehmer bereits gesehen hatte – wurde häufiger kritisiert und daher nach den ersten fünf Tests in „*Bekannt*“ geändert. Das Attribut „*Referenz*“ – für Links, bei denen der Browser inn-

erhalb einer Seite scrollt – ist zwar häufiger nachgefragt, aber dennoch beibehalten worden, da er einprägsam schien und es keine Verbesserungsvorschläge gab.

Im Gegensatz zur ersten Studie bemängelten bei HyperScout II nur noch drei Personen (weniger als ein Viertel der Teilnehmer), dass die Tooltips zu groß seien. Dies kann als Erfolg der Reduzierung und Optimierung der angebotenen Inhalte gewertet werden. Zwei Teilnehmer waren indes der Ansicht, dass die Tooltips grundsätzlich *zu klein* seien und sie deutlich mehr *inhaltliche Details* zum Zieldokument anbieten sollten: Eine umfangreichere Vorschau könnte ihrer Meinung nach einen genaueren Eindruck vom Link-Ziel vermitteln und würde es erlauben, noch häufiger unnötige Navigationsschritte einzusparen. Dennoch sollten die Informationen des Zieldokuments (wie bisher) zusammengefasst und nicht einfach als Bildschirmkopie dupliziert werden.

Einige Benutzer fänden es begrüßenswert, wenn die Tooltips noch konstanter in der Darstellung wären und zum Beispiel *immer* inhaltliche Informationen oben in den Tooltips zu finden seien, auch bei anderen Dateitypen als HTML.

Die Kritikpunkte und Vorschläge sind realisierbar und sollten für zukünftige Entwicklungen berücksichtigt werden. Eine *Konfigurierbarkeit* der Darstellung und des Umfangs der angebotenen Informationen könnte dazu beitragen, dass die Tooltips den Anforderungen einzelner Benutzer noch besser gerecht werden (s. Abschnitt 6.5.4).

6.5.3.3 Bedeutung der Informationen in den Tooltips für die Navigation

In dieser Studie gab es deutlich weniger kritische Anmerkungen zu den in den Tooltips dargestellten Link-Informationen als bei HyperScout I. Dies ist im Wesentlichen auf vier Punkte zurückzuführen:

- Erstens wurde ein Großteil der Informationen weggelassen, die die Studienteilnehmer von HyperScout I und die Teilnehmer der Online-Umfrage mehrheitlich als unbedeutend ansahen (wie den *topologischen Typ* und Hinweise auf *schnelle Server*, s. Abschnitt 6.5.3.3)
- Zweitens konnte die Redundanz inhaltlicher Informationen verringert werden, indem das System durch einen Vergleich von Anker-Text, Dokument-Titel und Beschreibung identische Angaben vermied.
- Drittens trugen die überarbeiteten Bezeichner der Attribute dazu bei, dass die Teilnehmer die Informationen einfacher verstanden.
- Da zudem viele Link-Eigenschaften bereits als Overlay und Maus-Icon erschienen, achteten die Teilnehmer insbesondere dann auf die Tooltips, wenn sie hier interessante oder weiterführende Informationen erwarteten oder Hilfe zur Bedeutung der Maus-Icons benötigten.

Im Folgenden wird auf die einzelnen Kategorien von Link-Informationen eingegangen, gruppiert nach der Kategorisierung von HyperScout II aus Abschnitt 6.4.3.

1. Fehlermeldungen beim Zugriff auf das Link-Ziel

Hinweise auf fehlerhafte Links wurden – ebenso wie bei den beiden anderen Visualisierungstechniken – von *allen Teilnehmern als sehr wichtig angesehen*. Einige Personen hatten allerdings Verständnisschwierigkeiten mit den Angaben zur Ursache des Fehlers. Ein Beispiel ist die Meldung „Timeout der Anfrage: Server antwortet nicht"; dem Benutzer war der Begriff *Timeout* nicht vertraut. Bei der Angabe „*Dokument existiert nicht mehr*" fragte ein Anwender irritiert, woher das System wisse, dass das Dokument früher einmal existiert habe.

Es gibt mehrere Möglichkeiten, solche Probleme zu reduzieren: Ein *Hilfesystem* kann Antworten auf Benutzerfragen geben, und die Meldungen sollten an die Anforderungen und die Technikkompetenz der Anwender *anpassbar* sein, beispielsweise indem unterschiedliche Modi für „Anfänger" und „Experten" angeboten werden.

2. Dateityp und Protokoll

Die Hinweise auf den *Dateityp* und das verwendete *Transferprotokoll* wurden ebenfalls zumeist als bedeutsam für die Navigation eingeschätzt, sofern diese Information nicht bereits im Link-Anker ersichtlich war. Probleme verursachten bei einigen Teilnehmern Dateiformate wie *Postscript*, sofern sie ihnen unbekannt waren. Es wurde eine *Hilfefunktion* für die einzelnen Formate und die benötigten Applikationen gewünscht. Bei komprimierten Dateien wollten zwei Anwender wissen, was sich *in* dem Archiv befände, beispielsweise ein ausführbares Programm („ohne Virus?") oder Mediendaten.

Bei den Informationen zum Protokoll wünschten fast ein Drittel der Teilnehmer, dass der Hinweis auf per *https* gesicherte Verbindungen nicht zuoberst und gelb hinterlegt im Tooltip erscheinen sollte: Dies sei eine eher „sekundäre Information", die Web-Anwender „gewohnt" seien und „keinerlei Schwierigkeiten bereiten".

3. Inhaltliche Informationen zum Link-Ziel

Die inhaltlichen Informationen wurden häufig als „eigentlicher Mehrwert der Tooltips" bezeichnet, da sie sich, im Gegensatz zu den meisten anderen Informationen, nicht aus den Link-Overlays oder den Maus-Icons ableiten ließen. Weitere Kommentare ähnelten denen bei der Evaluation von HyperScout I; die Bewertung konkreter Angaben hing entscheidend vom *Link-Anker*, dem *Zieldokument* und der *Aufgabe* des Benutzers ab.

Der *Seitentitel* wurde insbesondere kritisiert, wenn das Zieldokument nur über unzureichende Angaben verfügte. Mithilfe einer Liste von *Stoppwörtern* wie „Homepage" und „No Title" sollte in zukünftigen Entwicklungen unnütze Titel gefiltert und durch alternative Informationen ersetzt werden, wie der ersten Überschrift des Dokuments.

Auch bei diesen Tests wurde die von den Autoren der Webseiten erstellte *Beschreibung* im Meta-Tag (sofern vorhanden) meist positiver bewertet als die automatische *Inhaltszusammenfassung* des Ziel-Dokuments (vergl. Abschnitt 6.2.3.4). Trotz der Überarbeitung scheint der

Algorithmus zur Erstellung der Inhaltsangabe liefert er immer noch häufig unbefriedigende Ergebnisse. Zwei Vorschläge zur Optimierung des Algorithmus waren die *Auflistung aller Überschriften* des Dokuments (als eine Art Inhaltsverzeichnis) und die Nutzung der zuletzt bei Google eingegebenen Suchbegriffe zur Extraktion relevanter Seiteninhalte.

Allerdings sieht es der Autor dieser Arbeit aufgrund seiner Studien inzwischen als zweifelhaft an, dass eine automatische Zusammenfassung von Webseiten in wenigen Textzeilen repräsentativ genug sein kann, um in den meisten Fällen ausreichende Informationen für die Navigation zu bieten. Stattdessen sollten *optional* umfangreichere Zusammenfassungen in den Tooltips verfügbar sein, wie es von drei Teilnehmern gewünscht wurde. Eine Möglichkeit bestünde darin, dem Anwender zwei unterschiedliche Detailgrade anzubieten: Wird z. B. gleichzeitig eine Taste gedrückt, werden größere Tooltips mit ausführlichen Informationen zum Inhalt des Link-Ziel angezeigt, ansonsten kleine Tooltips, die als inhaltliche Information lediglich den Seitentitel enthalten.

Die Erstellung automatischer Zusammenfassungen von Webseiten hat sich insgesamt als große Herausforderung herausgestellt und könnte aufgrund der großen Relevanz von einem eigenen Forschungsprojekt wieter untersucht werden.

4. Weitere inhaltliche Metadaten zum Link-Ziel

Von den „weiteren" inhaltlichen Angaben zum Link-Ziel wurde das *Änderungsdatum* zumeist positiv bewertet, sofern es sich auf eine kürzlich überarbeitete Seite bezog. Hinweise auf ältere Dokumente wurden hingegen von über der Hälfte der Teilnehmer als weniger bedeutsam angesehen. Zwei Teilnehmer wollten darüber hinaus wissen, *was* auf einer überarbeiteten Seite seit dem letzten Besucht geändert wurde. Eine solche Funktion wäre durch eine Erweiterung des HyperScout-Clients realisierbar, indem er die letzte Kopie einer Seite bereithält und mit der neuen Version vergleicht (s. auch Adar, Teevan et al. 2009). Dieses Konzept wurde inzwischen durch Teevan et. al. erfolgreich realisiert und getestet (Teevan, Dumais et al. 2010).

Hinweise zur *Dateigröße* erschienen während der Tests nur relativ selten, da die Minimalgröße für die Anzeige auf 100kBytes angehoben wurde. Viele Teilnehmer sahen diese Angabe zumeist lediglich für Downloads als wichtig an; nur zwei Teilnehmer wünschten solche Daten auch für Webseiten, da sie zu Hause eine langsame Internetanbindung hätten (vergl. Chen, Subramanian et al. 2009).

Deutlich positiver als beim ersten Test wurden Informationen über die *Anzahl der Hyperlinks* im Zieldokument bewertet. Dies ist insbesondere auf die vereinfachte Präsentation zurückzuführen. Sieben Teilnehmer sagten aus, dass für sie solche Informationen oft wichtig seien, da sie viel über den „Charakter" und die „Navigationsmöglichkeiten" einer Seite hergäben. Eine Person wünschte eine Anzeige der Link-Anzahl ebenfalls für Seiten mit wenigen Links. Zwei Teilnehmer empfanden diese Meldungen teilweise auch als problematisch, da z. B. der

Hinweis auf 129 interne Links „eher abschrecken würde“ – hier wurden zu viele Informationen befürchtet. Zudem müsste das System zwischen „inhaltlichen Links und Werbe-Links“ unterscheiden, da letztere nicht zur Navigation beitrügen.

5. Die frühere Benutzung des Ziel-Objekts

Fast zwei Drittel der 13 Teilnehmer empfanden Informationen zu *früheren Zugriffen* auf ein Link-Ziel relevant für die Navigation. Sie könnten vermeiden, dass man „im Kreis navigiere“. Die Angabe, *wann* man das letzte Mal eine Seite angesehen hat, wurde vor allem dann als wichtig angesehen, wenn der letzte Zugriff schon einige Tage oder länger her sei, da der Benutzer sich ansonsten zumeist noch an den letzten Besuch erinnern könne.

6. Die Link-Aktion

Hinweise auf *Referenz-Links*, bei denen der Browser innerhalb der Seite scrollt, wurden zumeist als sinnvoll angesehen. Drei Personen kritisierten aber den im Tooltip dargestellten *Namen* des Zielankers als „wenig aussagekräftig“ (s. Abb. 107). Stattdessen sollte der Text des Link-Ziels im Tooltip erscheinen, gemäß dem Seiten-Titel bei „normalen“ Links.

Die Information, dass der Browser für das Zieldokument in einem *neuen Fenster* öffnet, fanden – wie beim ersten Test – nur wenige Teilnehmer hilfreich, zumal diese Information schon als Maus-Icon existiert. Die Anzeige des Attributs im Tooltip sollte deaktivierbar sein.

7. Die Verteilungscharakteristik des Links

Viele Kommentare bezogen sich auf die Darstellung externer Links in den Tooltips. Fünf der 13 Teilnehmer sahen diese Information als wesentlich für ihre Navigation an und sechs weitere meinten, dass es von der Site und der Aufgabe abhängig sei, ob ihnen diese Information weiterhelfe.

Obwohl in HyperScout II für die Kennzeichnung externer Links nur noch die *First-* und *Second-Level Domains* berücksichtigt wurden (s. 6.4.3), äußerten (auch) bei diesem Test viele Benutzer, dass die Semantik stärker berücksichtigt werden solle. Ein Vergleich der Domain-Namen alleine erwies sich als unzureichend, da beispielsweise „http://www.w3c.org/“ und „http://www.w3.org/“ dieselbe Website des World-Wide-Web-Konsortiums adressieren, obgleich die Second-Level-Domain unterschiedlich ist (s. Abschnitt 4.5.1.2 für weitere Lösungsansätze).

Für fünf Teilnehmer waren Informationen über den *Anbieter* der Informationen bei externen Links von großer Bedeutung. Allerdings wurden die entsprechenden (aus dem Seitentitel der Homepage, s. 5.5.4.2) gewonnenen Hinweise in den Tooltips unterschiedlich bewertet, je nachdem, ob der Benutzer den Anbieter kannte und er aufgrund der Angabe Rückschlüsse auf die *Qualität* und *Vertrauenswürdigkeit* der Site ziehen könnte. Vier Teilnehmer äußerten, dass ihnen verlässliche, explizite Informationen zur *Qualität* oder *Seriosität* des Link-Zieles in den Tooltips fehlen würden. Sie hätten gerne Hinweise auf Werbe-Links und Warnungen

vor möglicherweise „gefährlichen“ oder „zweifelhaften“ Seiten. Da diese Daten nicht automatisch zu generieren sind und die Anbieter entsprechender Websites solche Warnhinweise kaum selbst bereitstellen werden, muss man auf andere Verfahren zurückgreifen. Es bieten sich beispielsweise Phishing-Datenbanken (s. Abschnitt 3.3.3) und kollaborative Techniken aus dem Social-Navigation-Bereich an (s. Abschnitt 3.1.6), wie sie bereits im *CoInternet-Projekt* erprobt wurden (Baier, Weinreich et al. 2004; Wollenweber 2004).

8. Der URI

Die Meinungen zur Anzeige des URI[178] waren grundverschieden. Einige Personen befanden diese Information als „unverzichtbar“, andere als „völlig überflüssig“; manche bevorzugten zur Anzeige den Tooltip, andere die Statuszeile. Eine Konfigurierbarkeit der Darstellung dieses Attributs ist daher notwendig.

Zusammenfassung der Bedeutung der Tooltip-Inhalte für die Navigation im Web

Dank der Überarbeitung und Reduzierung der dargestellten Informationen in den Tooltips traten viele der Kritikpunkte aus der ersten Studie beim Test von HyperScout II nicht mehr auf. Dennoch gibt es weiteres Optimierungspotenzial. Auffällig waren insbesondere die unterschiedlichen Vorlieben und Anforderungen der Teilnehmer.

Obwohl auch bei dieser Studie insgesamt Hinweise auf fehlerhafte Links als wichtigster Navigationshinweis angesehen wurde, erachteten viele Teilnehmer die Angaben zum Inhalt der Link-Ziele als besonders relevant *für die Tooltips*, da diese Informationen weder in den Overlay-Farben noch den Maus-Icons codiert waren. Sie wurden insbesondere als Navigationshilfe empfunden, wenn:

- die Link-Anker wenig aussagekräftig oder missverständlich waren, sie nur aus ein oder zwei Worten bestanden oder Abkürzungen enthielten,
- auch aus dem Kontext nicht eindeutig ersichtlich wurde, wohin der Link führt,
- sich die Anker-Texte mehrerer Links ähnelten,
- die Internetanbindung langsam war und man auf das Ziel-Dokument warten musste,
- wenn die „Sorge“ bestand, dass ein Link zu einer möglicherweise unseriösen oder gefährlichen (da „virenverseuchten“) Ressource führt. Dies gilt insbesondere für externe Links.

Einige der Link-Eigenschaften, wie seine Verteilungscharakteristik und die Länge des Anker-Textes, lassen sich somit für eine Priorisierung der Link-URIs beim Prefetching (s. Abschnitt 7.2.1) heranziehen, sodass zusätzliche Daten für die Links, bei denen der Informationsbedarf des Anwenders potenziell am größten ist, zuerst bereitgestellt werden.

[178] Fast die Hälfte der Teilnehmer gab an, normalerweise auf den URI in der Statuszeile des Browsers zu achten, wenn die Navigation problematisch sei oder die angebotenen Link-Informationen nicht ausreichten.

6.5.3.4 Vorschläge der Teilnehmer zur Erweiterung der Link-Tooltips

Die im Umgang mit dem Web routinierten Studienteilnehmer trugen auch dieses Mal neue Ideen zur Erweiterung der Tooltips bei. Einige Vorschläge deckten sich mit denen der ersten Studie (s. Abschnitt 6.2.5); die neuen Vorschläge bezogen sich vornehmlich auf die Interaktionsmöglichkeiten mit den Tooltips:

- Ein Teilnehmer fand es „schade, dass die Tooltips verschwinden, wenn ich mit der Maus dort hingehen möchte". Er wollte im Tooltip einzelne Punkte auswählen können, um mehr Details zu erfahren und beispielsweise bei externen Links direkt zur Homepage der Site springen zu können.
- Bei der Anzeige der internen und externen Link-Anzahl sollten die populärsten der Links mittels eines im Tooltip aufrufbaren Popup-Menüs auswählbar sein. Auf diese Weise könnte man die Zielseite „überspringen" und die Navigation beschleunigen.
- Eine Anzeige der Anzahl von „Back-Links" – also aller anderen Seiten, die ebenfalls auf das Link-Ziel verweisen – würde einen Eindruck über den „Link-Kontext" und die Qualität des Zieldokuments vermitteln. Die Titel der referenzierenden Seiten sollten ebenfalls in einer Popup-Liste erscheinen (s. auch Yesilada, Lunn et al. 2007).
- Bei Tooltips für Links zu früher bereits besuchten Seiten sollte der Anwender sehen können, auf welchem Weg er zuvor dorthin gelangt war und welche Links er letztes Mal auf der Seite ausgewählt hatte. Eine solche Funktion würde das Reproduzieren früherer Pfade vereinfachen (s. auch Obendorf, Weinreich et al. 2007).

Alle vier Vorschläge wären auf Basis des vorgestellten Prototyps realisierbar; sie würden aber zum Teil über die Ziele des HyperScout-Konzepts hinausgehen, da sie dem Benutzer nicht nur *zusätzliche Informationen zur Hilfe bei der Navigation* bieten, sondern er ebenfalls neue *Navigationsmöglichkeiten* erhält. Die neuen Navigationsmöglichkeiten könnten gleichzeitig zu neuen Herausforderungen bei der Benutzung des Webs führen und sollten daher ebenfalls empirisch untersucht werden.

6.5.4 Anforderungen an die Konfigurierbarkeit von Link-Previews

Fast jeder Teilnehmer äußerte Wünsche zur Konfigurierbarkeit des Systems. Lediglich zwei Personen wollten selbst keine Einstellungen vornehmen (können), da sie befürchteten, die zusätzliche Komplexität könnte sie überfordern.[179]

Der HyperScout-II-Prototyp bot bereits zahlreiche Konfigurationsmöglichkeiten, diese wurden aber aufgrund der beschränkten Testdauer nicht erprobt; zudem war das Konfigura-

[179] Weitere Forschungsergebnisse zu individuellen Unterschieden und Anforderungen der Benutzer von Hypertext und der Konfigurierbarkeit entsprechender Schnittstellen findet man in (Nielsen 1989) und (Stanyer & Procter 1999).

tionsmenü im Prototyp nicht in den Browser integriert, sondern nur aus der Plug-in-Konfiguration von Scone erreichbar (s. Abschnitt 8.6.6ff).

Die häufigsten Anforderungen der 13 Teilnehmer an die Konfigurierbarkeit von HyperScout II waren Folgende:

- Die Farbzuordnung der Overlays sollte anpassbar sein (gewünscht von vier Personen).[180]
- Eine Personalisierbarkeit der Farbpalette wurde zwar nur einmal verlangt, hierfür spricht jedoch, dass Menschen unterschiede bei der Farbwahrnehmung zeugen. Beispielsweise sind (je nach Region) ca. 5-12 % der Männer von einer Rot-Grün-Sehschwäche betroffen (McIntyre 2002).
- Zwei Teilnehmer wollten die Maus-Icons an- und abschalten können, da sie die Tooltips bevorzugten und die Icons ihnen keinen Mehrwert boten. Beide merkten an, dass sich dies nach einer Eingewöhnungszeit ändern könnte.
- Drei Benutzer wünschten eine Anpassbarkeit der Maus-Icons. Zwei wollten *alle* Icons aus dem Tooltip auch beim Mauszeiger angezeigt bekommen, einer wollte die Anzeige auf die (seines Erachtens) wichtigsten beschränken.
- Sieben der Teilnehmer wollten die Tooltips aktivieren und deaktivieren können, da ihres Erachtens in vielen Situationen Link-Overlays und Maus-Icons bereits ausreichende Informationen lieferten. Dennoch sahen gleichzeitig zwei von ihnen die Tooltips als *bedeutendste Navigationshilfe* an, da sie „am aussagekräftigsten" seien.
- Mehrfach wurde vorgeschlagen, dass die Tooltips nur bei gleichzeitigem Drücken der Taste für die Overlays erscheinen sollten. Dies verhindere das ungewollte Einblenden der Tooltips. Dieses Verhalten sollte sich daher ebenfalls anpassen lassen.
- Ein Konfigurationsmenü für die in den Tooltips dargestellten Daten wurde von zwei Personen vermisst. Allerdings äußerten fast alle Teilnehmer Änderungswünsche zur Anzeige einzelner Attribute. Sie wollten bestimmte Informationen entfernen und Parameter wie die minimale Dateigröße für „große" Dateien und das maximale Alter für „neue" Dateien anpassen.
- Eine häufig vertretene Ansicht war, dass die Bedeutung der HyperScout-Erweiterungen und der angebotenen Zusatzinformationen entscheidend von der Benutzeraufgabe abhängig sei. Beim ungerichteten Browsen (z. B. für die Unterhaltung) und wenn man die Site gut kenne, könne man auf die meisten der HyperScout-II-Hinweise verzichten. Etwa die Hälfte der Teilnehmer wollte daher alle Erweiterungen schnell ein- und ausschalten

[180] Zudem zeigte sich während der Studie, dass unterschiedliche CRT- und LCD-Bildschirme die Farben der Overlays in anderer Tönung, Sättigung und Helligkeit anzeigten. Diese Parameter sollten daher anpassbar sein. Für die Tests wurde der Prototyp modifiziert, damit die Overlays bei allen Tests mit gleicher Intensität erschienen.

können. Auf diese Weise ließe sich der Browser in einen „Navigations-Modus“ schalten, wenn man beispielsweise nach Informationen auf unbekannten Websites sucht.

Da viele Teilnehmer ihre Einstellungen während der Nutzung des Webs vornehmen wollten, ist die Integration der Konfigurationsschnittstelle direkt in die Oberfläche des Browsers – beispielsweise in die *Browser-Toolbar*, oder ein Pull-down-Menü – wichtig. Eine direkte Browser-Integration erfordert neben dem Scone-Framework ein zusätzliches Browser-Add-On (s. Abschnitt 8.3.1).

6.5.5 Die drei Visualisierungstechniken im Vergleich

Zum Abschluss jedes Tests wurden die Teilnehmer befragt, welche der drei Visualisierungstechniken von HyperScout II sie bevorzugen und als besonders hilfreich für die Navigation einschätzen würden. Wie bereits die Äußerungen und das Benutzerverhalten während der partizipativen Tests aufzeigten, wichen die Präferenzen der Teilnehmer deutlich voneinander ab – weitaus mehr, als dies vor der Studie erwartet wurde.

Zwei Teilnehmer empfanden die *Link-Overlays* als nützlichste der drei Erweiterungen, da sie die „Orientierung auf der Seite erleichtern“ und man diese Informationen „als Erstes wahrnimmt“. *Fünf Teilnehmer* bevorzugten die *Maus-Icons*: Sie seien informativer als die Overlays, würden nicht wie die Tooltips wichtige Teile der Seite verdecken und die Informationen schnell wahrnehmbar präsentieren. *Vier Personen* bevorzugten die *Tooltips*, da sie am verständlichsten seien, fast genauso schnell wie die Maus-Icons erscheinen, aber mehr Details böten. Zudem müsse man die Bedeutung der in den Tooltips angebotenen Informationen nicht erst erlernen. Zwei der vier Personen äußerten allerdings, dass die Maus-Icons möglicherweise für sie nach längerer Benutzung an Bedeutung gewinnen, wenn sie erst besser mit ihnen vertraut seien.

Letztendlich standen zwei Teilnehmer allen drei Techniken kritisch gegenüber. Einer meinte, dass die meisten der Zusatzinformationen bei seinem schnellen Internetzugang überflüssig seien (s. auch Akscyn, McCracken et al. 1988) und ihm lediglich die Overlays bei der Identifikation von Link-Ankern helfen würden. Der Zweite sah keine der drei Darstellungsformen als optimal an und wünschte sich stattdessen eine gleichzeitige Anzeige *aller* Maus-Icons *hinter allen* Link-Ankern, sobald er die Overlay-Taste drückt. Eine solche Erweiterung ist auf Basis des hier vorgestellten Prototyps ebenfalls realisierbar, hätte aber nach Ansicht des Autors dieser Arbeit den bedeutenden Nachteil, dass Webseiten mit vielen Links durch die zahlreichen zusätzlichen Icons unübersichtlich werden und die Benutzer überfordern könnten.

Die Nennung einer bevorzugten Visualisierungstechnik bedeutete nicht, dass die jeweiligen Teilnehmer die Nutzung der anderen Techniken ausschlossen: Drei Personen sahen sogar alle drei Techniken als zusammengehörig an – sie wirken „wie aus einem Guss“ und seien

„eigentlich nur gemeinsam wirklich sinnvoll“. Für fünf Anwender waren Maus-Icons und Tooltips eine Erweiterung, die Link-Overlays eine davon unabhängige zweite.[181]

6.5.6 Fazit der Evaluation der Konzepte zur Erweiterung der Link-Benutzungsschnittstelle

Die Evaluation von HyperScout II hat zahlreiche Verbesserungen bei der Gebrauchstauglichkeit der untersuchten Konzepte gegenüber HyperScout I aufgezeigt. Durch die Überarbeitung der Tooltips konnte ein Großteil der in der ersten Studie ermittelten Probleme behoben oder reduziert werden, obgleich die Teilnehmer weiterhin häufig nicht auf das Erscheinen der Tooltips warteten. Die Einführung der Link-Overlays und der Maus-Icons hat die Nützlichkeit der Erweiterung wesentlich erhöht, da so die wichtigsten Link-Eigenschaften ohne Verzögerung erschienen, schnell wahrgenommen wurden und die Icons gleichzeitig auf die Existenz der Tooltips hinwiesen. Des Weiteren konnte HyperScout II durch das Angebot von drei verschiedenartigen Darstellungsmethoden den unterschiedlichen Vorlieben und Anforderungen der einzelnen Teilnehmer besser gerecht werden.

Die wesentlichen Unterschiede im Teilnehmerverhalten und bei der Bewertung der drei Techniken (s. Abschnitt 6.5.5) **zeigt, wie individuell Menschen heute mit Webbrowsern interagieren**. Im Bereich der Hypertext-Forschung kam bereits (Nielsen 1989) bei einer Auswertung von 92 quantitativen Studienergebnissen aus 30 wissenschaftlichen Artikeln der Hypertext-Usability zu einer ähnlichen Erkenntnis: Aufgrund der großen Abhängigkeit der Studienergebnisse von den Benutzern und den bearbeiteten Aufgaben hatte er kaum Hoffnung auf ein einzelnes, universelles Hypertext-System, das allen Anforderungen gerecht werde. Der Autor der vorliegenden Arbeit hat gemeinsam mit drei Kollegen in einer Langzeitstudie zur Benutzung des World Wide Webs im Jahr 2006 ebenfalls eine unerwartet große Variabilität der Benutzer festgestellt, die sich nicht nur auf ihre Gewohnheiten und Aufgaben, sondern auch auf die Eigenschaften der regelmäßig besuchten Sites zurückführen ließen (Obendorf, Weinreich et al. 2007;Weinreich, Obendorf et al. 2008).

Ein relevanter Faktor für die Gesamtbewertung der Konzepte von HyperScout II durch die Teilnehmer war ihr IT-Hintergrundwissen und ihre Routine beim Umgang mit dem Web. Personen ohne IT-Ausbildung und solche, die im Teilnehmerdurchschnitt eher selten das Web nutzten, nahmen die Änderungen gegenüber ihrem normalen Browser tendenziell eher als störende neue Herausforderungen wahr. Beispielsweise gab ein solcher Benutzer an, dass er Neuerungen an seinem Computer „grundsätzlich kritisch“ gegenüberstände, da er sich dann umgewöhnen müsse; er konnte sich aber vorstellen, dass die getesteten Konzepte nach einer Eingewöhnungszeit für ihn hilfreich seien. Ein weiterer Teilnehmer fand zwar zahlreiche Ideen von HyperScout II gut, meinte aber, dass für ihn so etwas eher irrelevant sei, da er das Web nur selten nutze.

[181] Als Gründe für diese Zweiteilung wurde aufgeführt, dass nur für die Aktivierung der Overlays eine Taste gedrückt werden musste und dass Maus-Icons und die Tooltips dieselben Symbole verwendeten.

Die sechs Teilnehmer aus dem IT-Bereich nahmen die Konzepte von HyperScout II hingegen durchweg mit großem Interesse auf und machten mehrere Vorschläge für ihre Optimierung und Erweiterung. Einige der positiven Äußerungen waren:

- „Die inhaltlichen Informationen in den Tooltips ersparen einem viele Klicks."
- „Das passt sich gut in das Ganze ein und wirkt, als wenn es ein Teil des Webs sei."
- „Die Tooltips helfen häufig, Suchergebnisse besser zu verstehen, da sie oft eine bessere Vorschau auf das Dokument bieten als die Ergebnisliste."
- „Bei Download-Links erhält man genau die Informationen, die einem sonst zumeist fehlen."

Dies deutet darauf hin, dass die Zielgruppe der untersuchten Methoden für automatische Link-Previews eher häufige und erfahrene Nutzer des Webs sind. Man könnte aber vermutlich den Bedenken und Problemen wenig routinierter Anwender mit einer vereinfachten Variante gerecht werden, da sie sich großteils primär durch die *Menge* der Änderungen überfordert fühlten. Eine Ausgangskonfiguration für „Einsteiger", die nur die wichtigsten Link-Attribute berücksichtigt – Fehlermeldungen, den Datentyp und den Titel des Link-Zieles (s. Umfrage zur Bedeutung von Link-Attributen in Abschnitt 6.3) –, würde die Einstiegshürde für viele Benutzer potenziell reduzieren. Dies sollte erprobt werden.

Vor dem Einsatz von HyperScout II als Produktivsystem sind zudem folgende Defizite zu beheben:

- Es wird ein **Hilfesystem** benötigt, das die Bedeutung der einzelnen Attribute erklärt und eine Legende der Overlay-Farben und der Maus-Icons bietet.
- Eine einfache und schnelle **Konfigurierbarkeit** des Systems ist für viele Anwender entscheidend: Die drei Erweiterungen sollten sich einzeln als auch gemeinsam anpassen und ein- und ausschalten lassen.
- In der Ausgangskonfiguration sollten die *Tooltips nur erscheinen*, wenn der Anwender die *Taste für die Overlays* gedrückt hält.
- Die *inhaltlichen Informationen* in den Tooltips müssen das Link-Ziel noch *präziser* beschreiben, damit sie eine für die Navigation verlässliche Vorschau darstellen.

Die Studien zeigten, dass Benutzer im Web häufig **Probleme bei der Identifikation von Link-Ankern haben** und Methoden zu ihrer eindeutigen Hervorhebung eine wichtige Benutzungshilfe darstellen. Ebenfalls ist festzuhalten, dass in vielen Fällen zusätzliche Informationen über die Funktion und das Ziel eines Links für die Navigation relevant sind und diese Attribute unterschiedliche Bedeutung haben, wobei die genauen Anforderungen vom Benutzer, seinen Erfahrungen, Aufgaben und den Rahmenbedingungen abhängen.

Trotz der umfangreichen Studienergebnisse muss angemerkt werden, dass bisher nicht zu ermitteln war, welche konkreten Auswirkung der Einsatz von Link-Previews im täglichen

Gebrauch des Webs haben wird. Einige Teilnehmer waren der Ansicht, dass sie den Umgang mit einer solchen Erweiterung erst erlernen müssten, um das Potenzial tatsächlich zu nutzen. Andere sagten aus, dass sie nach einer gewissen Eingewöhnungszeit wahrscheinlich „anders surfen" würden, da die Link-Informationen von HyperScout II ihre Navigationsgewohnheiten beeinflusse (vergl. auch die Ergebnisse von: Stanyer & Procter 1999). Obgleich sich auch mehrfach beobachten ließ, dass die erweiterte Link-Schnittstelle Einfluss auf das Vorgehen der Teilnehmer bei der Link-Auswahl hatte, lässt sich hieraus kein statistisch quantifizierbar anderes Verhalten bei der Link-Navigation im Web durch Link-Previews ableiten. Dies war jedoch auch ***nicht*** das Ziel der vorgestellten Studien, bei denen es primär darum ging, Defizite bei der Link-Navigation und bei den vorgestellten Konzepten auf qualitative Weise zu bestimmen und Erkenntnisse zu ihrer Optimierung zu gewinnen.[182]

Die vorgestellten Konzepte können zudem gewährleisten, dass Browser bei der Darstellung von Link-Ankern in Webseiten selbsttätig wesentliche Kriterien der ISO-Norm zur Benutzbarkeit von webbasierten Systemen (ISO 9241-151:2008, Sektionen 8 und 9) erfüllen. So werden Link-Anker durch die Overlays einfach erkennbar („*Links should be easily recognizable*", siehe: ISO9241-151 2008, 9.4.2) und Links zu besonderen Dateitypen (ISO9241-151 2008, 9.4.9), Links, die neue Fenster öffnen (ISO9241-151 2008, 9.4.10) und "*Transaction Links*" (d. h., Links, die Aktionen ausführen, siehe: ISO9241-151 2008, 9.4.4) werden durch Farben, Maus-Icons und Tooltips – wie gefordert – gekennzeichnet (siehe auch Kapitel 9.2).

Das nächste Kapitel 7 geht auf technische Konzepte ein, mit denen sich HyperScout effizient im gegenwärtigen Web realisieren lässt und stellt die notwendigen Erweiterungen an den Protokollen und Sprachen des Webs hierfür vor. Danach geht Kapitel 8 auf softwaretechnische Konzepte zur Realisierung von Web-Erweiterungen ein und stellt die systematische Entwicklung eines plattformunabhängigen Frameworks vor, das es ermöglicht, Systeme zur Erweiterung der Funktionalität und Benutzungsschnittstelle des Webs zu gestalten und das als Basis für die Implementation der hier getesteten Prototypen diente.

[182] Für statistische signifikante Ergebnisse zum geänderten Navigationsverhalten bei Nutzung der erweiterten Link-Schnittstelle sind spezifisch hierfür gestaltete Studien notwendig, bei denen z. B. entweder mehrere Teilnehmer dieselben Testaufgaben mit alternierenden Bedingungen bearbeiten (vergl. Weinreich, Obendorf et al. 2001) oder bei denen das Verhalten von vielen Teilnehmern beim täglichen Einsatz über einen längeren Zeitraum berücksichtigt wird (vergl. Weinreich, Obendorf et al. 2006b). Solche zusätzlichen Studien waren jedoch im Rahmen dieser Arbeit aus zeitlichen Gründen nicht durchführbar (s. auch Kapitel 9.3.4:„*Zukünftige Schritte*").

7 Konzeption von Protokollen und Diensten zur Erweiterung der Link-Schnittstelle

In den vorhergehenden Kapiteln wurden Konzepte zur Darstellung zusätzlicher Link-Informationen in Hypertext-Systemen erarbeitet und mit Benutzern evaluiert. Die vorgestellten Schnittstellenkonzepte können zur Verbesserung der Benutzbarkeit des Webs beitragen, indem den Anwendern bereits vor dem Folgen eines Links wesentliche Eigenschaften des Links und des Link-Zieles zugänglich gemacht werden. Aus technischer Sicht ist es dabei entscheidend, dass die für diese *Link-Previews* (s. Abschnitt 4.1.1) benötigten Daten aktuell sind und bei Bedarf verzögerungsfrei zur Verfügung stehen. Die bei den Prototypen *HyperScout I* und *HyperScout II* eingesetzte Methode des *Prefetching* (s. Abschnitt 7.2.1) stellt zwar die Aktualität der Daten sicher, ist aber relativ aufwendig in Bezug auf die Menge der zu übertragenden Daten und die benötigte Zeit, bis die Informationen für alle Links einer Seite bereitstehen. Beides ist in der Praxis in vielen Situationen unzumutbar (s. Abschnitt 3.3.1).[183]

Dieses Kapitel setzt sich mit den technischen Herausforderungen und Möglichkeiten bei der automatischen Bereitstellung der für das HyperScout-System benötigten zusätzlichen Link-Daten auseinander. Da das World Wide Web gegenwärtig das bedeutendste Hypertext-Informationssystem ist, orientieren sich die hier präsentierten Lösungsmöglichkeiten an den technischen Eigenschaften des Webs.

Zuerst werden die Anforderungen und Ziele bei der Entwicklung der technischen Konzepte zur Bereitstellung zusätzlicher Link-Daten im Web diskutiert (Abschnitt 7.1). Dann werden drei hierfür zur Verfügung stehende grundlegende Ansätze präsentiert und miteinander verglichen (Abschnitt 7.2). Kapitel 7.3 stellt ein auf diesen Ansätzen aufbauendes Konzept vor, mit dem sich die zuvor definierten Ziele erreichen lassen. Die hierfür geeigneten Sprachen und Protokolle werden in Abschnitt 7.4 analysiert, und Teilkapitel 7.5 spezifiziert die für das HyperScout-Konzept entwickelten Erweiterungen der Sprache XHTML und des Protokolls HTTP, welche eine effiziente Übertragung zusätzlicher Link-Daten ermöglichen. Das Kapitel schließt mit einer Diskussion über die vorgestellten Erweiterungen (Abschnitt 7.6).

7.1 Technische Anforderungen für die Erweiterung der Link-Schnittstelle im Web

Die in dieser Arbeit vorgestellten Konzepte zur Erweiterung der Benutzungsschnittstelle von Links in Hypertext-Informationssystemen sollten nicht nur, wie in Kapitel 5.5.3 vorgestellt,

[183] Diese Probleme traten bei den Benutzbarkeitsstudien von HyperScout kaum zutage, da die Benutzer vorher festgelegte Aufgaben bearbeiteten und die lokale Datenbank vor dem Test präpariert wurde. Zudem verfügten die eingesetzten Computer über eine sehr schnelle Internetanbindung.

prototypisch implementiert werden, sondern ebenfalls zum täglichen Einsatz im Web geeignet sein. Dieses Ziel setzt ein praxistaugliches technisches Konzept voraus, das die für die HyperScout-Erweiterung benötigten zusätzlichen Daten verfügbar macht und dabei die besonderen Herausforderungen großer verteilter Softwaresysteme (Tanenbaum & Steen 2002; Sommerville 2004) und die Eigenheiten des Webs berücksichtigt. Hierzu gehören insbesondere die Anforderungen an *Skalierbarkeit* und *Performanz* des Systems, die *Konsistenz der Daten* als auch die *Offenheit* gegenüber und die *Kompatibilität* mit den vielen unterschiedlichen Client- und Server-Systemen des Webs:

1. Eine elementare Anforderung an jedes global verteilte Informationssystem sind *Skalierbarkeit* und *Performanz*. Das Web hat sich nicht zuletzt deshalb gegenüber anderen Hypertext-Systemen durchgesetzt, weil es auch auf globaler Ebene sehr gut skalierte und dem nahezu exponentiellen Wachstum der Benutzerzahlen in den ersten Jahren gewachsen war. Die in vielen Aspekten überlegenen offenen Hypertext-Systeme *Intermedia* und *Microcosm* (s. Anhänge B.10 und B.13) zeigten aufgrund ihrer zentralen Dokumenten- und Link-Server bereits bei der Nutzung durch eine größere Zahl lokaler Anwender ihre Grenzen (Hall 1997). Selbst Hyper-G, das mit dem Ziel der globalen Einsetzbarkeit entwickelt wurde, hatte schon bei wenigen hundert Servern und einer bei weitem geringeren Nutzerzahl als das Web Performanzprobleme (Weinreich 1997: 31).[184] Ein System zum Bereitstellen erweiterter Link-Informationen im Web müsste ebenso skalierbar sein wie das gegenwärtige Web.

2. Zur *Performanz* des Systems gehört aus Benutzersicht auch die Minimierung des *Overheads*, der durch den Transfer der Daten für die Link-Previews von HyperScout entsteht. Zusätzliche Daten können nicht nur die Übertragungszeit erhöhen, sondern auch zusätzliche Kosten verursachen (z.B. beim mobilen Zugriff). Die Übertragungszeit ist insbesondere vor dem Hintergrund kritisch zu sehen, dass das Web bereits ohne eine solche Erweiterung häufig Performanzdefizite aufweist (s. Abschnitt 3.3.1).

3. Die *Konsistenz* der zusätzlichen Link-Daten ist wichtig, damit die zusätzlichen Link-Informationen dem Anwender einen echten Mehrwert bieten und er beispielsweise *zuverlässig* auf fehlerhafte Links hingewiesen wird. Dem Anspruch einer hohen Konsistenz der Link-Daten wird das Web gegenwärtig oft nicht gerecht (s. Abschnitt 3.3.4), daher stellt dies eine besondere Herausforderung dar.

[184] Obwohl die langsamen Antwortzeiten bei dem in (Weinreich 1997) aufgeführten Tests nicht alleine auf konzeptionelle Schwächen zurückzuführen waren, deutet Einiges darauf hin, dass beispielsweise die Konsistenzmechanismen von Hyper-G bereits bei Größenordnungen von einigen zigtausend Systemen an ihre Grenzen gestoßen wären (Pitkow & Jones 1996).

4. Die *Kompatibilität* der zu entwickelnden Erweiterung mit dem aktuellen Web ist unverzichtbar, da ein Einsatz mit existierenden Websites möglich sein soll. Die vorgestellten Konzepte müssen daher mit den Markup-Sprachen HTML und XHTML (Raggett, Hors et al. 1999; Pemberton 2002) sowie dem Protokoll HTTP (Fielding, Gettys et al. 1999) harmonieren und mit den technischen Eigenschaften von Webservern und Browsern vereinbar sein.

Diese Anforderungen sind zum Teil konträr: Eine optimale Lösung zur Bereitstellung konsistenter Link-Informationen ist nicht unbedingt mit den aktuellen Protokollen und Sprachen des Webs kompatibel oder kann Schwächen bei der Skalierbarkeit zeigen (s. Abschnitt 7.2.4).

Zur weiteren Analyse der möglichen Vorgehensweisen werden im Folgenden die grundlegenden technischen Konzepte zur Bereitstellung zusätzlicher Link-Daten im Web vorgestellt und ihre jeweiligen Potenziale und Grenzen diskutiert.

7.2 Grundlegende Konzepte zur Bereitstellung zusätzlicher Link-Informationen im Web

In einem lokalen Hypertext-Informationssystem stellt der schnelle Zugang zu allen Informationen in der Regel keine besondere Herausforderung dar. Für ein global verteiltes Informationssystem ist der Zugriff aufgrund der Datenverteilung oft mit erheblichen Verzögerungen verbunden. Wie bereits dargelegt (s. Abschnitt 3.3.1), gilt dies auch für das Web. Obwohl alle Links eines Web-Dokuments im HTML-Quellcode eingebettet sind und damit alle technisch benötigten Informationen zum Aufruf des Zielobjekts vorliegen, fehlen aus Benutzersicht häufig viele semantische Informationen über Link und Ziel (vergleiche Kapitel 3.2ff). Insbesondere sind die meisten Daten über das Zielobjekt im Browser erst nach dem „Folgen" des Links, also der Übertragung des entsprechenden Objekts, zugänglich.

Dieser Abschnitt stellt die drei grundlegenden technische Herangehensweisen zur Bereitstellung erweiterter Link-Informationen für verteilte Hypertext-Informationssysteme vor und vergleicht ihre jeweiligen Potenziale und Grenzen miteinander.

7.2.1 Konzepte zur clientseitigen Bereitstellung zusätzlicher Link-Daten

Ein Verfahren, um Informationen zu den Link-Zielen eines Dokuments zu ermitteln, ist das sogenannte *Prefetching* (Zhang, Xu et al. 2000; Fisher 2002). Hierzu wird eine eigene Komponente im Client benötigt, die im Hintergrund für alle Links des aktuellen Dokuments die (noch nicht besuchten) Zielobjekte im Voraus lädt (s. Abb. 110).

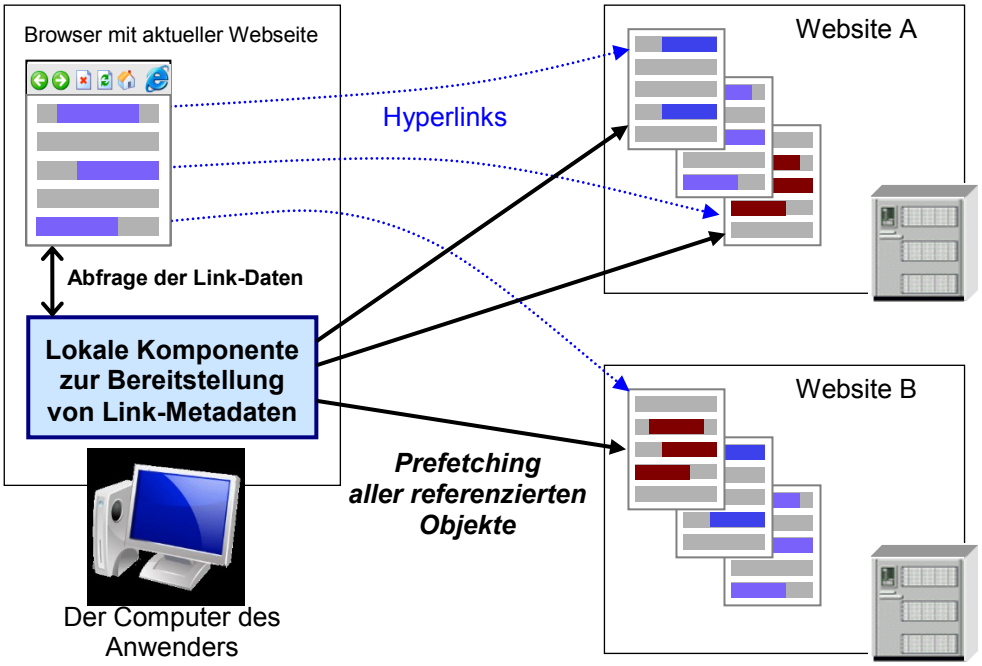


Abb. 110: Ein Agent, der die Metadaten für alle Verweise eines Dokuments per Prefetching bereitstellt.

Das *Prefetching* weist zahlreiche Stärken auf. Die gewonnen Informationen sind *aktuell* und *konsistent* mit den Link-Zielen, wie sie der Benutzer zu sehen bekommt: Regionale und temporäre Probleme (z. B. nationale Zugriffsbeschränkungen oder vorübergehende Netzwerkausfälle) sowie benutzerspezifische Parameter (Anmeldeinformationen und Sitzungs-Cookies, siehe: Kristol & Montulli 2000) werden berücksichtigt. Ein clientseitiges Prefetching setzt keine Erweiterungen der Sprachen und Protokolle des Webs voraus und kann die Benutzung des Webs sogar beschleunigen, wenn die nächste vom Benutzer besuchte Seite bereits im Hintergrund übertragen wurde (Zhang, Xu et al. 2000). Zudem bleiben nur wenige der in Abschnitt 4.5 klassifizierten impliziten Link-Informationen des Webs unzugänglich (dies sind unter anderem *Social-Navigation-Hinweise*, s. Abschnitt 4.5.5.2).

Den Vorteilen steht eine Reihe von Nachteilen gegenüber. Da Hypertext-Dokumente oft zahlreiche Links aufweisen, kann das Prefetching dazu führen, dass der Client ein Vielfaches der Datenmenge des aktuellen Dokuments zusätzlich zeitnah laden muss. Dies ist mit einer nicht unerheblichen *zusätzlichen Belastung* sowohl für den Netzwerkzugang des Anwenders als auch für die beteiligten Webserver verbunden. Würden viele Benutzer diese Technik einsetzen, könnte es die Performanz von Websites auch für andere Benutzer kritisch einschränken.

Ein weiteres Defizit stellt die *Verzögerung* bei der Bereitstellung der Daten dar. Die Bandbreite jedes Internetzugangs sowie die Anzahl gleichzeitig geöffneter TCP/IP-Verbindungen zwischen Client und Server sind beschränkt. Somit ist mit einer nicht unerheblichen Latenz zu rechnen, bis alle benötigten Daten per Prefetching übertragen sind. Als Konsequenz

stehen die Link-Hinweise oft nicht rechtzeitig zur Verfügung, wenn der Anwender sie benötigt.

Zur Reduzierung beider Probleme gibt es mehrere Ansätze. Eine Möglichkeit ist die Eingrenzung der relevanten Link-Ziele. Dabei werden nur die Objekte im Voraus geladen, die der Benutzer mit einer gewissen *Wahrscheinlichkeit* als nächstes auswählt. Diese Wahrscheinlichkeit lässt sich aufgrund eines Benutzerprofils, persönlicher Voreinstellungen oder der aktuellen Aufgabe des Anwenders berechnen (s. El-Saddik, Griwodz et al. 1998; Zhang, Xu et al. 2000). Zusätzlich können Webserver die Gewohnheiten aller Anwender auswerten und dem Client die populärsten Links der aktuellen Seite mitteilen, damit er nur diese im Voraus lädt (Padmanabhan & Mogul 1996). Weiterhin lässt sich die Browser-Interaktion des Benutzers auswerten, sodass sich das Prefetching auf die aktuell auf dem Bildschirm sichtbaren Link-Anker beschränkt oder der Client sogar erst aktiv wird, wenn der Mauszeiger in die Nähe eines Link-Ankers geführt wird (Stanyer & Procter 1999).

Ein anderer Ansatz zur Reduzierung der Datenmenge besteht darin, lediglich die HTTP-Header abzufragen.[185] So können zumindest Status-Informationen ermitteln werden wie die *Zugreifbarkeit* des Objektes, der *Dateityp*, das *Alter* und die *Größe* (Stanyer & Procter 1999). Werden zusätzlich die ersten Zeilen eines Dokuments übertragen,[186] erhält man zumeist auch Zugang zum Titel des Dokuments und zu Meta-Elementen mit weiteren grundlegenden Informationen (s. Abschnitt 4.5.2ff).

Der Umfang der zu übertragenden Daten ließe sich erheblich mithilfe einer Erweiterung des HTTP-Protokolls reduzieren, durch das der Client in *einer* Anfrage die *aufbereiteten* Metadaten zu *mehreren* Objekten, also z. B. allen Zielobjekten einer Webseite, abfragen könnte. Ein Projekt, das bereits prototypisch einen solchen Ansatz für das Web verfolgte, war der *ATLAS Hyperlink Database Server* (Pitkow & Jones 1996).

Die aufgeführten Optimierungen rücken das Prefetching für das Web für Anwender mit ADSL oder ähnlichem Breitbandzugang in den Bereich der praktischen Einsetzbarkeit, wie die Benutzbarkeitstests mit *HyperScout I* und *II* gezeigt haben. Bei geringerer Bandbreite ist aber bereits das Laden des eigentlichen Dokuments oft zu zeitaufwendig (s. Abschnitt 3.3.1; (Weinreich 1997: 21ff; Weinreich, Obendorf et al. 2006b). In diesen Fällen sind Dienste und Protokolle vonnöten, die die zusätzlichen Daten in optimierter Form übertragen (s. nächster Abschnitt 7.2.2) und die lediglich dann Bandbreite beanspruchen, wenn der Anwender nicht gerade selbst auf das Internet zugreift (Fisher 2002).

---

[185] Dies ist einfach möglich mittels des HTTP-„HEAD"-Befehles und wird von Browsern beispielsweise genutzt, um die Aktualität der Daten im lokalen Browser-Cache zu überprüfen (Fielding, Gettys et al. 1999).

[186] In HTTP kann mittels des Befehles `Content-Range` auch ein beliebiger Teil einer Datei übertragen werden.

7.2.2 Ein Dienst zur Bereitstellung erweiterter Link-Daten

Das zweite grundlegende Konzept zur Bereitstellung zusätzlicher Informationen zu den Links in einem verteilten Hypertext-Informationssystem basiert auf einem zusätzlichen (autonomen) Dienst, der für alle öffentlich zugänglichen Dokumente die zusätzlichen Link-Informationen bereitstellt. Das Client-System des Anwenders stellt dabei zur Ermittlung der zusätzlichen Link-Informationen *eine* Anfrage an den Dienst, sobald der Benutzer ein Dokument im Browser aufruft. Es erhält dann die gewünschten Informationen über alle Links und Zielobjekte der Seite auf einmal. Technisch ist diese Lösung verwandt mit früheren Projekten wie *Chimera* (Anderson 1997), *DLS* (Carr, DeRoure et al. 1998) oder dem *Connectivity Server* (Bharat, Broder et al. 1998), die als Link-Dienste zu jeder angezeigten Webseite *zusätzliche* Links und Link-Informationen bereitstellten. Der hier vorgestellte autonome Metadaten-Dienst soll allerdings keine zusätzlichen Links anbieten, sondern lediglich *weiterführende Informationen* zu den bereits vorhandenen Links (s. Abb. 111).

Dieser Ansatz weist gegenüber dem clientseitigen Prefetching wesentliche Vorteile auf. Zum einen muss der Webbrowser pro Seite nur eine zusätzliche Anfrage stellen, anstatt allen Links zu folgen. Zudem lässt sich der Umfang der zu übertragenden Daten durch ihre entsprechende Aufbereitung wesentlich reduzieren.

Eine weitere Stärke sind die Möglichkeiten zur Einbindung von Social-Navigation-Informationen (s. Abschnitt 4.5.5.2): Durch die zentrale Architektur des Ansatzes sind fast beliebige Konzepte von Empfehlungs-, Bewertungs- und Kommunikationssystemen für verteilten Hypertext realisierbar (vergl. „WOT“, Abschnitt 9.2.3).

Aufseiten der Webserver kommt es ebenfalls zu einer deutlich geringeren zusätzlichen Belastung als bei Prefetching. Der Metadaten-Dienst müsste zwar – analog zu dem Vorgehen globaler Suchmaschinen – regelmäßig alle Objekte des Webs mittels eines Crawlers abrufen und in einer lokalen Datenbank speichern. Da die so gewonnenen Daten aber durch viele Anwender gemeinsam genutzt werden, relativiert sich die zusätzliche Belastung der Webserver. Wird dieser Dienst zugleich von dem Anbieter einer globalen Suchmaschine wie *Google* betrieben, entsteht für Webserver gar kein zusätzlicher Datenverkehr, da die indexierten Daten von beiden Systemen gemeinsam genutzt werden können.

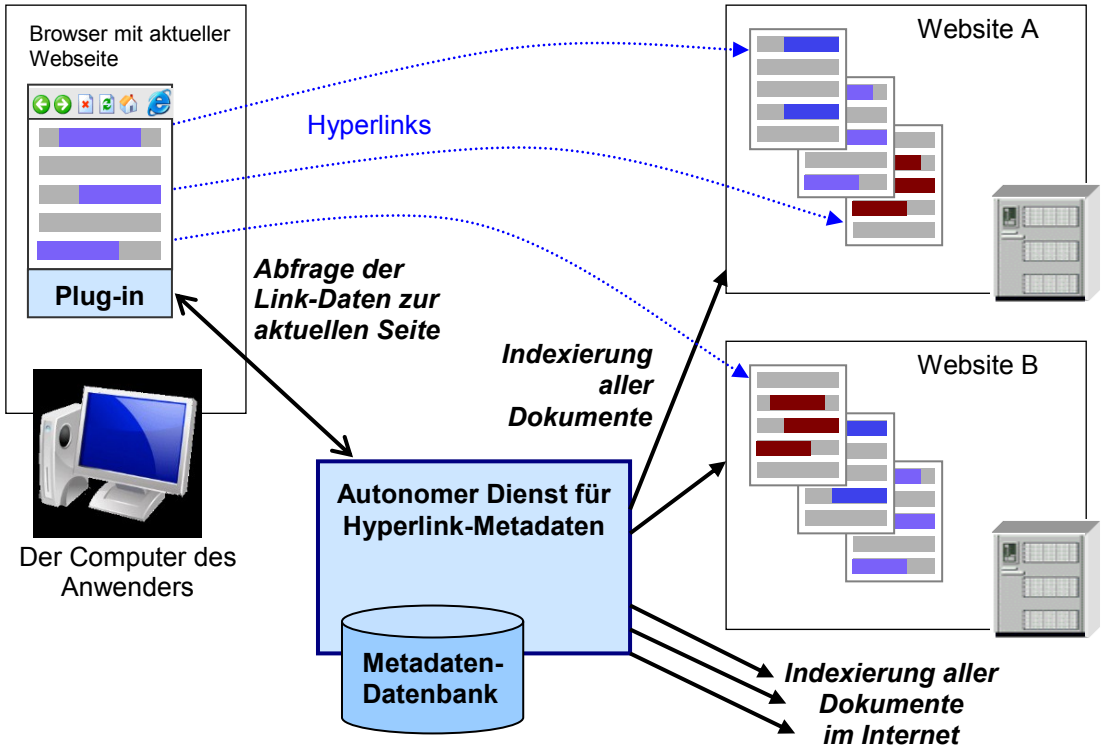


Abb. 111: Ein autonomer Dienst zur Bereitstellung von Metadaten zu Webseiten.

Ein autonomer Dienst für Link-Metadaten weist den gravierenden Vorzug auf, dass er ohne eine Anpassung aktueller Webserver einsetzbar wäre. Bereits mit seiner Realisierung und einer entsprechenden Browser-Erweiterung, die über eine Schnittstelle zu dem Dienst verfügt und der die Visualisierung der Link-Informationen durchführt, ließe sich das gesamte Web um die zusätzlichen Navigationshinweise des HyperScout-Konzepts erweitern. Ein Beispiel für ein solches Konzept liefert das Empfehlungssystem *WOT* (s. Kapitel 9.2.3)

Diesen Vorteilen steht allerdings eine Reihe von Herausforderungen gegenüber: Zunächst ist die *Skalierbarkeit* eines solchen zentralen Dienstes nicht unproblematisch, da die Webbrowser aller Nutzer des Dienstes für *jedes* angezeigte Dokument eine Anfrage an ihn stellen würden. Gleichwohl zeigen globale Suchmaschinen (s. Abschnitt 3.1.1), dass sich eine derartige Last gut auf ein globales Netz von Rechenzentren mit vielen lose gekoppelten Komponenten verteilen lässt (Brin & Page 1998), und Browser-Erweiterungen wie die Toolbars von *Google* und *Alexa* stellen ebenfalls für jede besuchte Webseite eine kurze Anfrage an eine zentrale Adresse: Bei der Google Toolbar wird der *PageRank*[187] der Seite ermittelt, und Alexa bietet unter anderem Informationen zur Popularität der Website an (s. Abschnitt 3.3.3 und Abb. 22).

Alternativ kann ein solcher Dienst auch innerhalb von Firmennetzen für die Mitarbeiter oder von Internetprovidern für die Kunden angeboten werden. Die gewonnenen Daten ließen

[187] Dieser Wert wird aufgrund der Hyperlinkstruktur der Seiten berechnet und zum Ranking der Suchmaschinenergebnisse herangezogen (Page, Brin et al. 1998).

sich dann von vielen Personen mit ähnlichen Tätigkeitsbereich (z. B. für Arbeitsgruppen im Firmennetz) und mit vergleichbaren Netzwerkbedingungen gemeinsam nutzen. Die derartige Eingrenzung des Nutzerkreises reduziert die Anforderungen an die Skalierbarkeit, stellt eine kurze Antwortzeit für die Anwender sicher und erlaubt die Einbindung gruppenspezifischer Daten.

Eine kritische Herausforderung für einen autonomen Link-Metadaten-Dienst ist die Aktualität und Konsistenz der Daten. Systembedingt treten dieselben Probleme wie bei globalen Suchmaschinen auf, da ein autonomer Dienst ebenfalls ein lokales Abbild des Webs erstellen muss, um die Informationen schnell zur Verfügung zu stellen. Folglich ist zur Konsistenzsicherung eine regelmäßige Aktualisierung der Daten notwendig (s. Abschnitt 3.3.1).

Ein Defizit des autonomen Dienstes für das HyperScout-Konzept ist, dass mehrere der vorgesehenen Link-Informationen (s. Abschnitte 4.5 und 5.5) durch ihn alleine *nicht* bereitstellbar sind. Beispielsweise können benutzerspezifische und regional abhängige Informationen – wie die Verzögerung beim Zugriff auf ein Dokument – nicht zuverlässig ermittelt werden. Zudem erfassen die Crawler solcher Systeme das sogenannte „*Deep Web*“ nicht,[188] wodurch für diese Links zusätzliche Informationen fehlen.

Darüber hinaus bringt ein solcher Metadaten-Dienst *rechtliche* Fragen mit sich. Es ist ungewiss, ob und in welcher Form überhaupt ergänzende Informationen zu beliebigen Dokumenten im Web angeboten werden dürfen. Webseiten sind in der Regel durch Autorenrechte geschützt, und ergänzende Informationen zu den Links einer Seite können als eine *Änderung des Seiteninhalts* angesehen werden. Dies kann in Konflikt mit den Urheberrechten des Seitenautors stehen und sogar die Rechte der Verfasser der Zieldokumente verletzen. Mehrere Gerichtsentscheidungen weisen auf die komplizierte Urheberrechtsproblematik bei Online-Medien hin, sogar wenn es sich um eingebettete Informationen handelt, die nicht direkt sichtbar sind oder nur auf Anforderung dargestellt werden. Beispielsweise dürfen Webseiten keine in Meta-Elementen „versteckten“ Schlüsselwörter aufführen, welche die Markenrechte anderer Firmen verletzen, da sie die Ergebnislisten von Suchmaschinen – und damit das Navigationsverhalten der Benutzer – beeinflussen (Ard & Musil 2001). Darüber hinaus können bereits Urheberrechte verletzt werden, wenn man Informationen einer Website außerhalb Ihres Kontextes zugänglich macht. So wurden Links auf Dokumente *innerhalb* einer Website (die sogenannten „Deep Links“) als unzulässig erklärt, wenn der Anbieter möchte,

[188] Das *Deep Web* umfasst Dokumente und Dienste des Webs, die nur mit einem Benutzerkonto zugänglich sind oder beispielsweise Formulareingaben erfordern. Hierzu gehören die Inhalte vieler Bibliotheksdatenbanken und Online-Dokumentensysteme (Bergman 2001; Lyman, Varian et al. 2003).

dass die Leser nur über seine Homepage zu den weiteren Angeboten der Site gelangen (Rötzer 1999).

Ein zentraler, von einem Dritten betriebener Dienst für Link-Metadaten weist überdies gewisse *Risiken* für die Benutzer auf. Da der Web-Client für jedes besuchte Dokument eine Anfrage an den Dienst stellt, lassen sich ausführliche Benutzungsprofile erstellen. Darüber hinaus könnten die Betreiber des Dienstes die angebotenen Daten manipulieren, um die Aktionen des Benutzers zu beeinflussen. Beispielsweise würde ein fälschlich als defekt markierter Link dazu führen, dass Benutzer ihn nicht auswählen.

Solche Risiken des Urheberrechts, des Persönlichkeits- und des Datenschutzes stellen die Akzeptanz eines von dritten angebotenen Link-Metadaten-Dienstes für das Web in Frage. Beispielsweise löste der Annotationsdienst *Third Voice* – eine Browser-Erweiterung, die das Hinzufügen von Links und Bemerkungen zu beliebigen Webseiten ermöglichte – heftigen Protest aus: Datenschützer und Firmen kritisierten, dass die Anbieter der Websites keine Kontrolle über die zusätzlich eingebrachten Informationen und Verknüpfungen auf ihren Seiten hätten. Insbesondere war die Besorgnis des Missbrauches groß, da sich leicht falsche oder verwerfliche Informationen einblenden ließen.[189] Ein vergleichbares Publicity-Fiasko erlebte Microsoft, als sie den Internet Explorer 6 um die sogenannten „*Smart Tags*“ (s. Abschnitt 5.3.7) erweitern wollten, mit denen Microsoft gegen Gebühr „hilfreiche“ Links in beliebige Webseiten einblenden konnte (Kaminski 2001). Die aufgeführten Fallstricke sind bei der Konzeption des autonomen Metadaten-Dienstes zu berücksichtigen.

7.2.3 Konzepte zur serverseitigen Bereitstellung zusätzlicher Link-Daten

Der dritte grundlegende Ansatz zur Bereitstellung zusätzlicher Meta-Informationen zu den Links eines verteilten Hypertext-Informationssystems basiert darauf, dass jeder Server die zusätzlichen Link-Daten für *alle lokalen Dokumente* selbst anbietet. In Bezug auf das Web müsste zu diesem Zweck jeder Webserver über eine entsprechende Erweiterung, den *serverseitigen Link-Metadaten-Dienst,* verfügen, der diese Aufgabe übernimmt (s. Abb. 112).

Gegenüber den beiden vorherigen Konzepten weist diese Lösung eine ganze Reihe von Vorteilen auf. Zum einen sind serverseitig alle Objekte und damit die meisten der in den Abschnitt 5.5 aufgeführten impliziten Link-Informationen *direkt lokal* zugreifbar. Es müssten lediglich zusätzlich solche Dokumente auf anderen Servern indexiert werden, auf die ein (externer) Link in den lokalen Seiten verweist (s. Abb. 112 rechts). Da viele Websites nur über relativ wenige externe Links verfügen (s. Abschnitt 4.5.1.2), würde sich der Netzwerkverkehr zur Indexierung dieser Ressourcen in der Regel in Grenzen halten.

---

[189] Die Website von Angela Lee „Say No To Third Voice” fasst einige Kritikpunkte zusammen: http://saynotothirdvoice.com/ (zuletzt aktualisiert im Juni 2010).

Der Dienst kann zudem serverspezifische Social-Navigation-Informationen in die Navigation mit einbringen. Beispielsweise lassen sich die Popularität von Dokumenten und beliebte Pfade der Anwender innerhalb einer Site sichtbar machen (Wexelblat 1999).

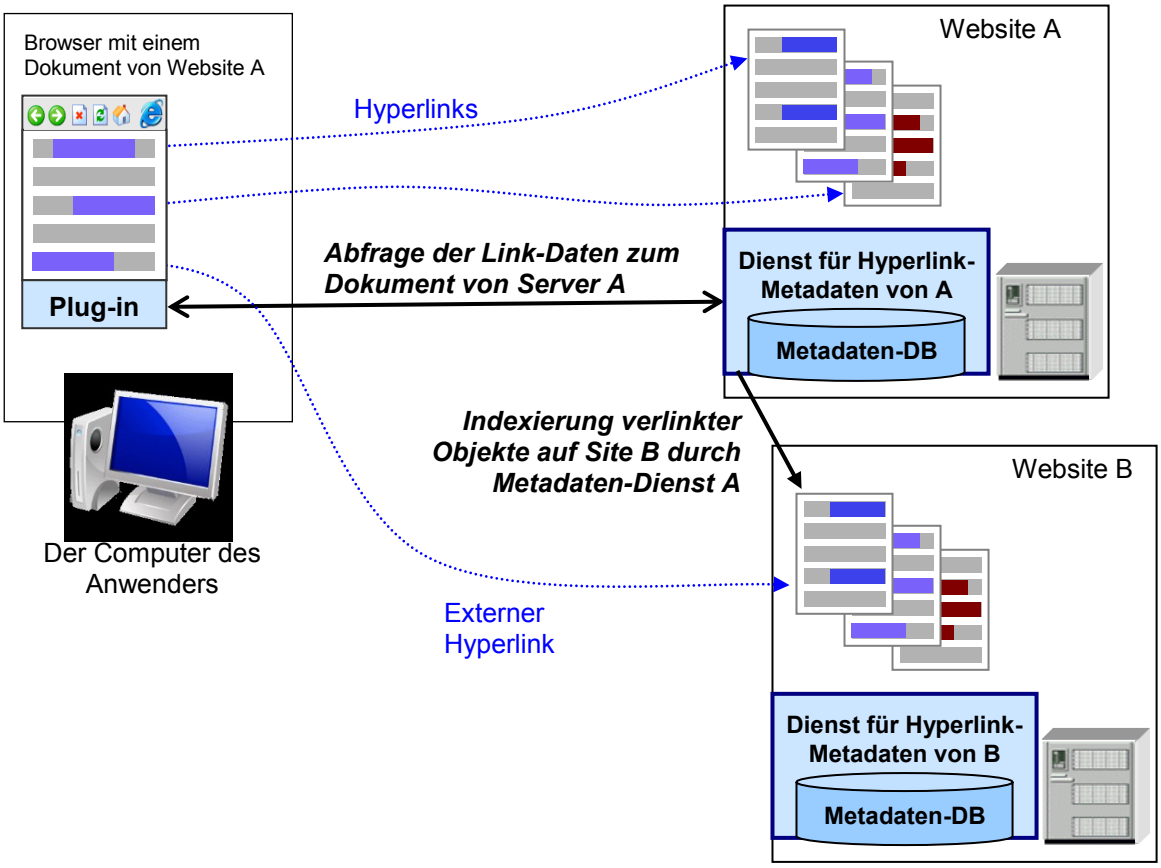


Abb. 112: Jeder Webserver verfügt über eine eigene Komponente zur Bereitstellung erweiterter Link-Informationen.

Ein weiterer entscheidender Vorteil ist die im Vergleich zu den beiden anderen Konzepten *insgesamt minimale zusätzliche Netzwerkbelastung* für den Anwender, da der Webbrowser für jedes abgerufene Dokument bereits eine HTTP-Verbindung zum Webserver aufbaut[190] und sich diese ebenfalls zur Übertragung der zusätzlichen Link-Informationen nutzen lässt.

Dieser Ansatz weist darüber hinaus eine ausgezeichnete *Skalierbarkeit* auf, da jeder Webserver nur für die Bereitstellung der erweiterten Link-Informationen aller lokalen Dokumente zuständig ist.

Die wesentliche praktische Herausforderung dieses Konzepts besteht darin, dass sämtliche Webserver über eine entsprechende Erweiterung verfügen müssen, um die im HyperScout-Konzept vorgesehenen zusätzlichen Link-Daten für das ganze Web zugänglich zu machen. Zudem wird ein einheitlicher *Standard* benötigt, der Protokoll und Format der Datenüber-

[190] De facto bauen die meisten Webbrowser zumeist sogar mehrere HTTP-Verbindungen zu einem Server auf, damit die Übertragung der eingebetteten Objekte und Bilder schneller vonstatten geht (vergl. Abschnitt 3.3.1).

tragung festlegt. Websites ohne den Link-Metadaten-Dienst würden dem Benutzer zwar lediglich die zusätzlichen Link-Informationen vorenthalten, eine lückenhafte Verfügbarkeit könnte aber Akzeptanz und Nützlichkeit des Gesamtsystems einschränken.

Eine solche Erweiterung des „gesamten" Webs ist eine nicht zu unterschätzende Praxisbarriere. Es ist zumindest damit zu rechnen, dass die Einführung der neuen Server-Funktionalität längere Zeit in Anspruch nimmt und viele Anbieter eine entsprechende Anpassung ihrer Systeme scheuen. Allerdings könnte man für eine Überganszeit für alle nicht erweiterten Webserver auf die beiden vorherigen Konzepte zurückgreifen (s. Abschnitt 9.2.2).

7.2.4 Diskussion der drei Konzepte

Die drei vorgestellten Konzepte weisen jeweils Stärken und Schwächen auf und keines stellt alleine eine optimale Lösung dar.

Die *Performanz* des Systems ist aus Sicht der Anwender sicherlich einer der primären Faktoren. Der Umfang der zusätzlich zum Client zu übertragenden Daten sollte daher minimiert werden. Dabei schneidet die *serverseitige* Lösung am besten ab, dicht gefolgt vom autonomen Metadaten-Dienst: Sie können jeweils die Link-Metadaten in aufbereiteter, kompakter Form bereitstellen. Die Server-Erweiterung erfordert dabei keine zusätzliche HTTP-Verbindung, und die Informationen könnten sogar gemeinsam mit dem Dokument übertragen werden. So lässt sich sicherstellen, dass die Link-Hinweise gleichzeitig mit dem Dokument zur Verfügung stehen.

Die *zwischen den Servern* zu übertragende Datenmenge ist beim serverseitigen Dienst von der Anzahl ein- und ausgehender Links der Site abhängig. Da für die lokalen Objekte alle Link-Metadaten auch lokal zur Verfügung stehen, müssten lediglich externe Ressourcen, auf die direkt verwiesen wird, übertragen werden. Ein zentraler, autonomer Dienst müsste hingegen sämtliche Dokumente des Webs regelmäßig indexieren. Der beim Prefetching zu erwartende zusätzliche Datenverkehr ist hingegen so groß, dass er sogar für einige Webserver problematisch werden könnte, da für jeden Navigationsschritt aller Benutzer viele zusätzliche Zugriffe auf das Web verursacht werden.

Die beste *Skalierbarkeit* bietet ebenfalls die serverseitige Lösung, da die Last auf alle Websites entsprechend der Benutzerzahlen verteilt wird und sich die Anzahl der Anfragen an den serverseitigen Link-Metadaten-Dienst linear zur Anzahl der Seitenzugriffe verhält.

Eine weitere Stärke der serverseitigen Lösung ist die *Vollständigkeit* der verfügbaren Metadaten. Das System könnte sie für *alle* lokal vorhandenen Dokumente – auch diejenigen, die dem „*Deep* Web" zuzuschreiben sind – verfügbar machen, also beispielsweise auch, wenn

ein Login für den Zugriff notwendig ist, indem die Site für den Dienst entsprechend angepasst wird.

Ähnlich sieht es für die *Konsistenz* der Daten aus. Durch den lokalen Zugriff auf die Daten kann der serverseitige Metadaten-Dienst jederzeit konsistente Informationen für alle lokalen Links bereitstellen, lediglich externe Links wären regelmäßig zu überprüfen (s. Abschnitt 7.3.2). Die clientseitige Lösung stellt die Konsistenz aufgrund der Aktualität der Abfrage sicher, ein autonomer Dienst hätte hingegen ähnliche Probleme wie heutige Suchmaschinen.

Der autonome Link-Metadaten-Dienst ist aus Sicht des *Persönlichkeits-* und *Datenschutzes* als kritisch zu betrachten: Die Navigationsaktionen aller Benutzer können vollständig nachverfolgt werden; darüber hinaus besteht die Gefahr, dass die angebotenen Informationen manipuliert werden und Anbieter durch die geänderte Darstellung der Hyperlinks ihrer Dokumente ihr Copyright verletzt sehen. Bei der serverseitigen Lösung steht es dem Anbieter hingegen frei, nur solche zusätzlichen Informationen zu den Links in seinen Dokumenten anzubieten, die er für richtig hält. Lediglich die clientseitige Lösung verhindert weitgehend die Manipulation durch den Anbieter oder Dritte, und der Benutzer hat die größte Kontrolle über die angezeigten Informationen.

Bezüglich der *praktischen Einführung* bestehen gegenwärtig die geringsten Hürden bei der clientseitigen Realisierung, da über die Browser-Erweiterung hinaus keine zusätzlichen Dienste benötigt werden – jeder Anwender kann ein solches System bei Bedarf lokal installieren. Demgegenüber setzt die serverseitige Lösung voraus, dass (möglichst) alle Anbieter ihre Server entsprechend erweitern und die notwendige Komponente installieren. Eine Motivation kann die Verbesserung der Benutzbarkeit einer Website sein, mit der sich die Zufriedenheit der Anwender steigert, was wiederum Benutzer- und Verkaufszahlen positiv beeinflusst. Darüber hinaus kann der neue Dienst sogar die Arbeit der Autoren vereinfachen: Die Gestaltung von Links wird erleichtert, wenn sich Autoren und Benutzer darauf verlassen können, dass zusätzliche Hinweise auf besondere Link-Eigenschaften automatisch verfügbar gemacht werden. Zusätzlich kann der Anbieter jederzeit Informationen über fehlerhafte Links und Änderungen an den Zieldokumenten seiner externen Verweise einsehen. Dies vereinfacht die Verwaltung eines Servers und dient der Qualitätssicherung.

Darüber hinaus ist zu berücksichtigen, dass bei keiner der drei Techniken alleine alle der in Abschnitten 5.5 aufgeführten Link-Informationen des HyperScout-Konzepts zugänglich sind. Persönliche Benutzungsinformationen, vom System des Benutzers abhängige Parameter (wie die unterstützten Dateiformate) sowie regionale und temporäre Eigenschaften der Objekte (hierzu gehören personalisierte Seiten und lokale Zugriffsprobleme) können nur clientseitig zuverlässig bereitgestellt werden. Andere Informationen sind einen zentralen

autonomen Dienst oder einem serverseitigen Dienst vorbehalten, z. B. solche aus dem Social-Navigation-Bereich.

Hier wird daher als optimale Lösung eine Kombination des client- und des serverseitigen Konzepts vorgeschlagen: Die meisten Hyperlink-Informationen sollten mittelfristig von allen Websites durch entsprechende serverseitige Dienste übertragen werden. Ergänzend müsste der clientseitige Dienst *grundsätzlich* wichtige benutzerspezifische Parameter bereitstellen, wie eine Historie aller früheren Benutzungsaktionen und die aktuelle Zugreifbarkeit von Websites. Als Überganglösung – solange viele Sites einen solchen Dienst nicht anbieten – ließe sich zusätzlich ein autonomer Metadaten-Dienst hinzuziehen, der beispielsweise von Betreibern globaler Suchmaschinen angeboten werden könnte (s. Abschnitt 9.2.2).

Im folgenden Abschnitt 7.3 werden die Architektur des HyperScout-Konzepts zur technischen Bereitstellung zusätzlicher Link-Daten und danach die hierfür entwickelten Sprachen und Protokolle vorgestellt (s. Abschnitte 7.4 und 7.5).

7.3 Konzepte zur effizienten Bereitstellung erweiterter Link-Informationen im Web

Dieses Kapitel präsentiert technische Konzepte, mit deren Hilfe die in Kapitel 5.5 vorgestellten impliziten Link-Informationen auf schnelle und effiziente Weise zugänglich gemacht werden können. Es kombiniert die jeweiligen Stärken der zuvor präsentierten Methoden zur Bereitstellung zusätzlicher Link-Informationen, um die Latenz und den Umfang der zusätzlich zu übertragenden Daten zu minimieren. Das Konzept basiert auf einer client- und einer serverseitigen Erweiterung des Webs. Die verwendeten Techniken sind *kompatibel* mit den aktuellen Sprachen und Protokollen des Webs und lassen sich problemlos praktisch realisieren, wie prototypisch gezeigt wird (s. folgende Abschnitte 7.3.1 und 7.3.2).

Die erste Komponente ist der clientseitige *HyperScout-Agent für Link-Metadaten*, der auf dem Computer des Anwenders läuft und die Bereitstellung und Visualisierung der zusätzlichen Informationen steuert. Die benötigten Daten werden entsprechend den jeweiligen Gegebenheiten von einem *serverseitigen Dienst* für Link-Metadaten, aus der Datenbank des clientseitigen HyperScout-Agenten mit History-Informationen zu den Aktionen des Benutzers und per *Prefetching* zusammengestellt (s. Abb. 113, links unten). Die *Visualisierungskomponente* sorgt für die Darstellung der erweiterten Link-Informationen im Browser (s. Abb. 113, links Mitte). Abschnitt 7.3.1 beschreibt die Funktionsweise des clientseitigen *HyperScout-Agenten* im Detail.

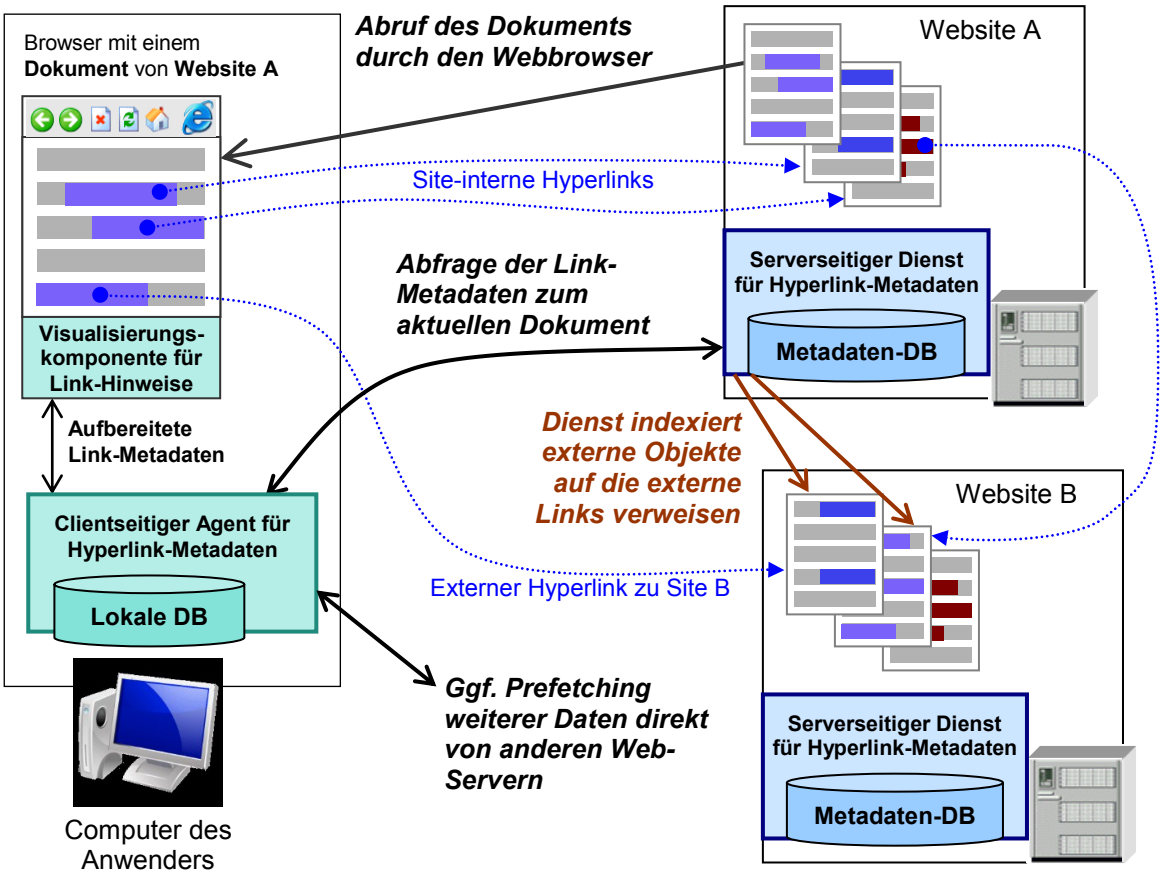


Abb. 113: Funktionsweise des dezentralen HyperScout-Konzepts zur Bereitstellung zusätzlicher Link-Metadaten.

Die meisten der zusätzlich benötigten Daten zu den Links und Link-Zielen einer Webseite soll der clientseitige Agent direkt von dem Server erhalten, von dem auch das im Browser dargestellte Dokument stammt. Damit dies auf effiziente Weise erfolgen kann, muss der Server über einen zusätzlichen *serverseitigen Dienst für Link-Metadaten* verfügen, der die notwendigen Daten vorbereitet und überträgt (s. Abb. 113 rechts). Abschnitt 7.3.2 geht auf die Realisierungsmöglichkeiten für einen solchen Dienst für aktuelle Webserver ein und beschreibt, auf welche Weise die Metadaten zusammengestellt werden können.

7.3.1 Funktionsweise des clientseitigen Agenten für Link-Daten

Das hier vorgestellte technische Konzept zur Erweiterung der Benutzungsschnittstelle von Hyperlinks im Web setzt eine zusätzliche Komponente auf dem Computer des Anwenders voraus. Dieser *HyperScout-Agent* für *erweiterte Link-Metadaten* sorgt für die Bereitstellung und Darstellung der zusätzlichen Informationen zu den Links einer Seite.

Mehrere Gründe sprechen für den Einsatz eines Software-Agenten auf dem Computer des Anwenders. Zum einen kann der Agent die Übertragung der zusätzlich benötigten Daten steuern und optimieren (s. u.). Zweitens sind einige der in Kapitel 4.5 vorgestellten impliziten Link-Informationen nur auf Clientseite erfassbar. Des Weiteren stellt die clientseitige Visualisierung der zusätzlichen Link-Informationen sicher, dass die erweiterte Benutzungs-

schnittstelle der Links für das gesamte Web *konsistent* ist und die angebotenen Link-Informationen sowie die Art der Visualisierung nicht mehr (wie bisher) von den Vorlieben des Autors einer Website abhängen. Zudem erhält der Anwender die Möglichkeit, die Benutzungsschnittstelle der Links zu *personalisieren*, also beispielsweise die Darstellung von Link-Markern und den Umfang der zusätzlichen Informationen zu kontrollieren. Insbesondere dies wurde von den Teilnehmern der Studien des HyperScout-Systems häufig als wichtig angesehen (s. Abschnitte 6.2.5 und 6.5.4).

Die fünf Hauptaufgaben des clientseitigen HyperScout-Agenten sind wie folgt (Abb. 114):

1. Anfordern der zusätzlichen Daten vom serverseitigen Dienst

Aus Gründen der Performanz sollen die meisten der zusätzlichen Link-Informationen durch *serverseitige* Metadaten-Dienste zusammengestellt und in vorbereiteter Form an den Client übertragen werden (s. auch Abschnitte 7.3.2 und 7.5ff). Damit diese Dienste nur die vom Anwender benötigten Daten übertragen, muss der HyperScout-Agent in der Lage sein, sie gezielt anzufordern. Im Rahmen dieser Arbeit wurde eine entsprechende Erweiterung des HTTP-Protokolls zur Spezifikation der gewünschten *Link-Metadaten* definiert (s. Abschnitt 7.5.3). Eine Aufgabe des clientseitigen HyperScout-Agenten besteht somit darin, die Anfrage an den Server um die entsprechenden Parameter zu erweitern.

2. Auswerten der vom Server bereitgestellten Daten

Webserver, die über den *HyperScout-Dienst* für erweiterte Link-Informationen verfügen (s. Folgeabschnitt 7.3.2), sind in der Lage, die im HTTP-Protokoll spezifizierten zusätzlichen Anweisungen zu interpretieren und entsprechende Link-Informationen (sofern verfügbar) an den Client zu übertragen. Hieraus folgt die zweite Aufgabe des HyperScout-Agenten: Die *Entgegennahme* und *Auswertung* der übermittelten Daten. Dabei kann es notwendig sein, sie in bestimmter Weise zu filtern und zu verarbeiten, damit sie sich in der vom Benutzer gewünschten Form präsentieren lassen. Fehlen einige der angeforderten Daten oder sind sie nicht in der gewünschten Qualität (z. B. Aktualität) verfügbar, so soll der Agent selbsttätig auf anderen Wegen versuchen, weitere Daten einzuholen.

3. Ergänzen fehlender Link-Metadaten

Da zum einen nicht zwangsweise alle Webserver über einen entsprechenden Metadaten-Dienst verfügen, zum anderen serverseitig nicht immer alle benötigten Daten zugänglich sind, kann es notwendig sein, dass der Metadaten-Agent zusätzliche Informationen auf anderem Wege, beispielsweise durch Prefetching, zusammenstellt. Hierzu gehören *benutzer-* und *ortsspezifische* Informationen (s. Abschnitt 7.2.4).

Dabei muss jeweils eine Abwägung zwischen der Bedeutung dieser Informationen und der verfügbaren Bandbreite bzw. der Kosten für das *Prefetching* getroffen werden. Einige der nur clientseitig zu ermittelnden Daten, wie die voraussichtliche Verzögerung bei der Übertragung und die aktuelle Erreichbarkeit eines Servers, lassen sich aber bereits mittels *einer* kurzen HTTP-„HEAD"-Anfrage pro Website abschätzen (s. Abschnitt 6.2.3.9). Zudem müssen für solche Zwecke nur vergleichsweise wenige Daten übertragen werden, sodass die Informationen in der Regel vorliegen, bevor der Anwender sie benötigt bzw. sich anzeigen lässt.

4. Aufzeichnen und Bereitstellen der Benutzeraktionen

Eine weitere Aufgabe des HyperScout-Agenten liegt darin, die *Aktionen des Benutzers* beim Browsen im Web aufzuzeichnen und zusammen mit den Metadaten der besuchten Objekte (z. B. Titel, Autor und Inhalt) in einer Datenbank dauerhaft zu speichern. Eine solche *umfassende History*[191] kann nicht nur dazu genutzt werden, früher angesehene Seiten über eine erweiterte Suchfunktion wiederzufinden (s. Abschnitte 3.1.1 und 3.1.8), sondern dem Anwender auch bei der Navigation mit Hyperlinks helfen. Wenn bei den Link-Ankern dargestellt wird, auf welche Seiten er wann zuletzt zugegriffen hat, werden frühere Navigationsschritte nachvollziehbar und der Benutzer kann entlang früherer Pfade navigieren (s. Kapitel 5.5ff).

Zusätzliche History-Informationen bei den Hyperlinks sind aber nicht nur eine Hilfe, um vor längerer Zeit gelesene Dokumente wieder zu finden; es erleichtert oft auch die Einschätzung der Qualität und Relevanz eines Dokuments, wenn der Anwender erkennt, in welchem Kontext bereits bekannten Materials es sich befindet (vergl. Abschnitt 4.5.5.1).

5. Visualisierung der Link-Metadaten

Die fünfte und letzte Aufgabe des Metadaten-Agenten besteht darin, dem Benutzer die gewonnenen Link-Informationen darzustellen. Zur Anpassung der Benutzungsschnittstelle von Links in Webbrowser bieten sich zwei Wege: Entweder muss das Programm zur Anzeige der Dokumente – sprich der Webbrowser – erweitert werden oder es sind die Dokumente selbst anzupassen. Die Vor- und Nachteile beider Ansätze werden in Kapitel 8.3ff diskutiert.

Das hier vorgestellte Konzept wurde auf Basis des *Scone-Frameworks* (s. Kapitel 8) als *Proof-of-Concept-Implementation* realisiert. Dieser Prototyp basiert auf dem Code des zur Evaluation von *HyperScout I* entwickelten Prototyps (s. Abschnitt 5.5.3), der zusätzlich einige der Anker-Attribute aus der im Abschnitt 7.5.2 vorgestellten XHTML-Erweiterung berücksichtigt.

---

[191] Im Kontrast zu den Anfangszeiten des Webs stellt eine solche Archivierung aller Benutzeraktionen heute keine nennenswerte Herausforderung für PCs mehr dar.

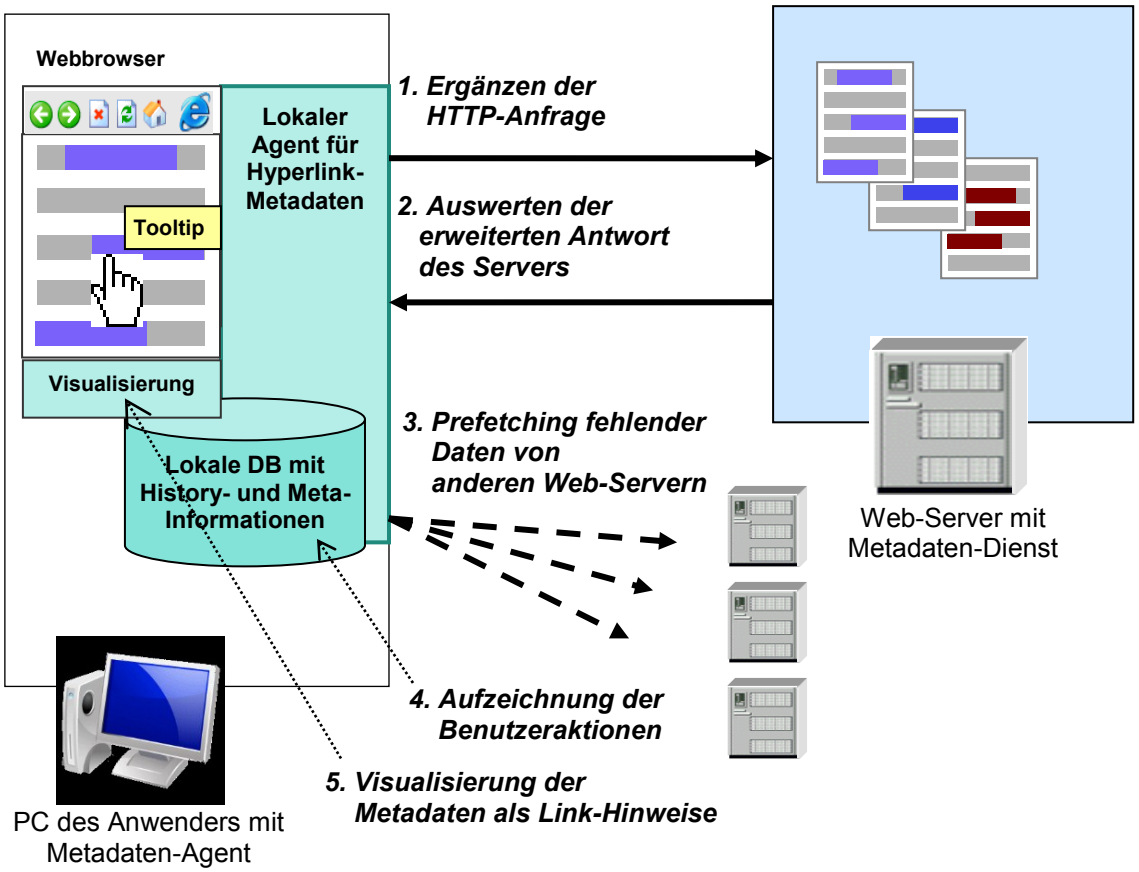


Abb. 114: Schematische Funktionsweise des clientseitigen HyperScout-Agenten.

7.3.2 Funktionsweise des serverseitigen Dienstes für Link-Metadaten

Damit Websites die erweiterten Link-Informationen des HyperScout-Konzepts automatisch zu allen angebotenen Dokumenten zur Verfügung stellen können, benötigen sie einen zusätzlichen Dienst. Dieser *HyperScout-Dienst* für *Link-Metadaten* muss zahlreiche Anforderungen erfüllen: Um eine akzeptable Performanz zu erreichen, sind die Link-Metadaten nahezu verzögerungsfrei bereitzustellen. Sie sollten daher in einer Datenbank zugreifbar sein und nicht erst zusammengestellt oder konvertiert werden müssen. Insbesondere müssen Informationen über referenzierte Objekte auf anderen Websites[192] bereits in der lokalen Datenbank vorliegen (s. Abschnitt 7.2.3).

Für die genaue Funktionsweise und Integration eines solchen serverseitigen Dienstes ist die Systemarchitektur des jeweiligen Webservers ausschlaggebend. Das Spektrum existierender Systeme ist vielfältig: Es reicht von „konventionellen" Webservern, die alle Dokumente im Dateisystem vorliegen haben, bis zu komplexen *Content-Management-Systemen (CMS)*, die sämtliche Inhalte in Datenbanken verwalten und alle Seiten dynamisch generieren.

[192] Hiermit sind alle Objekte gemeint, auf die ein Link innerhalb einer lokalen Seite verweist.

Bei einem *Content-Management-System* ließe sich ein solcher Metadaten-Dienst als Teil der Systemarchitektur realisieren, da sich (in der Regel) bereits alle Dokumente und Links in einer Datenbank befinden; folglich ist der schnelle, zentrale Zugriff auf alle Informationen gewährleistet (Boiko 2001; Zschau, Traub et al. 2002). Allerdings wäre zumeist eine Erweiterung des Datenbankschemas notwendig, um den Anforderungen an das HyperScout-Konzept zu genügen, sodass alle vorgesehenen Link-Attribute berücksichtigt werden und auch zusätzliche Daten über die von externen Links referenzierten Objekte zu erfassen sind. Da sehr viele unterschiedliche Content-Management-Systeme existieren, wurde ein solches Vorgehen im Rahmen dieser Arbeit nicht konkretisiert.[193]

Ein anderer, hier favorisierter Lösungsansatz ist von der Technik des Webservers weitgehend unabhängig. Er bietet sich ebenfalls für Systeme an, bei denen die Dokumente als statische Dateien vorliegen und ist auch für die recht verbreiteten, heterogen aufgebauten Websites geeignet, die viele individuelle Programme und Skripte zur Generierung gleichartiger Seiten verwenden. Dies setzt voraus, dass der Metadaten-Dienst über eine lokale Datenbank für die Link-Metadaten verfügt und in der Lage ist, die entsprechenden Informationen über alle auf der Website angebotenen Objekte in der Datenbank zu erfassen (s. Abb. 115). Ferner sind Änderungen am Datenbestand festzustellen, damit Aktualisierungen in die Datenbank übernommen werden. Das Erfassen der Objekte kann mit einem *Web-Crawler* (ähnlich wie bei einer Suchmaschine) geschehen; allerdings ist eine Überwachung der Dokumente auf Ebene des Dateisystems sinnvoll, um Modifikationen unmittelbar zu erkennen.[194]

Des Weiteren muss der serverseitige HyperScout-Dienst auch die Link-Informationen für alle *site-übergreifenden* Verknüpfungen mithilfe des Crawlers indexieren und in die lokale Datenbank übernehmen (s. Abb. 115). Hierbei ergeben sich potenzielle Konsistenzprobleme, da keine Technik im Web existiert, um Änderungen am Datenbestand einer externen Website automatisch festzustellen. Statt dessen muss der Dienst regelmäßig aktiv die Konsistenz überprüfen muss. Der durchschnittliche Anteil inkonsistenter Informationen in der Datenbank des HyperScout-Dienstes ist von dem Aktualisierungsintervall abhängig (vergl. Abschnitt 3.3.4 zur Konsistenz im Web).

---

[193] Inzwischen gibt es Hunderte von Web-Content-Management- und Portal-Systemen, die grundlegend unterschiedliche Architekturen aufweisen können (Zirpins, Weinreich et al. 2001; Zschau, Traub et al. 2002). Populäre Beispiele sind das in PHP geschriebene *Typo3* (www.typo3.com), das auf Zope/Python basierende *Plone* (www.plone.org) und das kommerzielle System *OpenText Web Experience Management* (*früher: Vignette StoryServer* (www.opentext.com). Einen Überblick geben www.cmsreview.com und www.cmsmatrix.org.

[194] Ein Beispiel für ein System, das eine vergleichbare Technik zur Überwachung der angebotenen Objekte verwendet, ist der *Microsoft Index Server*, der das lokale Suchsystem des *Microsoft Internet Information Service (IIS)* bereitstellt (Swank & Kittel 1996).

Verglichen mit globalen Suchsystemen ist die Konsistenzsicherung des HyperScout-Dienstes einfach. Nach den Analysen und Berechnungen von (Brewington & Cybenko 2000) werden für Datenbanken von Webseiten durchschnittlich bereits bei einem Update-Intervall von 5 Tagen 95% aller Inkonsistenzen innerhalb von zwei Stunden korrigiert. Da viele Websites nur über relativ wenige externe Links verfügen und Modifikationen der Ressourcen meist mit kurzen HTTP-HEAD-Anfragen festgestellt werden können,[195] hält sich der Umfang der regelmäßig zwischen den Servern zu übertragenden Daten in Grenzen.

Die Datenkonsistenz des HyperScout-Dienstes ließe sich wesentlich effizienter sicherstellen, wenn alle Server über ein entsprechendes Server-To-Server-Protokoll zum Austausch von Verknüpfungsdaten verfügten. Ein entsprechendes Konzept wurde bereits für *Hyper-G* entworfen und implementiert (s. Abschnitt 3.3.4 und Anhang B.14). Dabei zeigte sich allerdings auch, dass selbst durch solche Techniken bei global verteilten Informationssystemen nur ein Zustand der schwachen Konsistenz („*Weak Consistency*“) zu erreichen ist: Die Übermittlung der Änderungen an alle relevanten Server nimmt immer einige Zeit in Anspruch, die Datenkonsistenz wird folglich nur mit einer gewissen Verzögerung wiedererlangt (Kappe 1995). Zudem neigte das damals vorgestellte Konzept aufgrund seiner Komplexität zu Skalierungsproblemen (Pitkow & Jones 1996): Frank Kappe hatte seine Berechnungen auf einige Tausend Hyper-G-Server ausgelegt, die heutige Dimension des Webs liegt mehr als fünf Größenordnungen darüber.[196]

Eine optionale Komponente des HyperScout-Dienstes ist ein Interpreter zur Auswertung des Zugriffsprotokolls der Website (s. Abb. 115). Er analysiert die Benutzungsdaten der Anwender aus und macht sie als *Social-Navigation-Hinweise* zugänglich. Die Visualisierung populärer Pfade und bedeutsamer Dokumente kann die Navigation innerhalb einer Website erleichtern (vergl. Abschnitt 3.1.6).

Das hier vorgestellte Konzept wurde ebenfalls prototypisch mithilfe des Scone-Frameworks implementiert. Scone arbeite dabei als serverseitiger Intermediary (auch „Reverse Proxy“ gennant, s. Abschnitt 8.6.5.6); das heißt, die gesamte Kommunikation mit der Website wird für den Anwender auf transparente Weise gefiltert, um zusätzliche Link-Daten ergänzt und weitergeleitet. Die Informationen stellt der Scone-Crawler zusammen (s. Abschnitt 8.6.2), wobei der Persistenzmechanismus der *Scone-NetObjects* zum Speichern der Link-Informationen diente (s. Abschnitt 8.6.3).

---

[195] Der Server gibt auf eine HTTP-Anfrage mit dem HEAD-Kommando nicht das in der URI angegebene Objekt, sondern nur den entsprechenden *Response-Header* zurück. Dieser Protokollkopf enthält wichtige Statusinformationen, wie das Datum der letzten Änderung und die Dateigröße des adressierten Objektes (Fielding, Gettys et al. 1999).

[196] Das Web bestand Ende 2011 aus schätzungsweise über 500 Millionen aktiven Websites (Netcraft 2011).

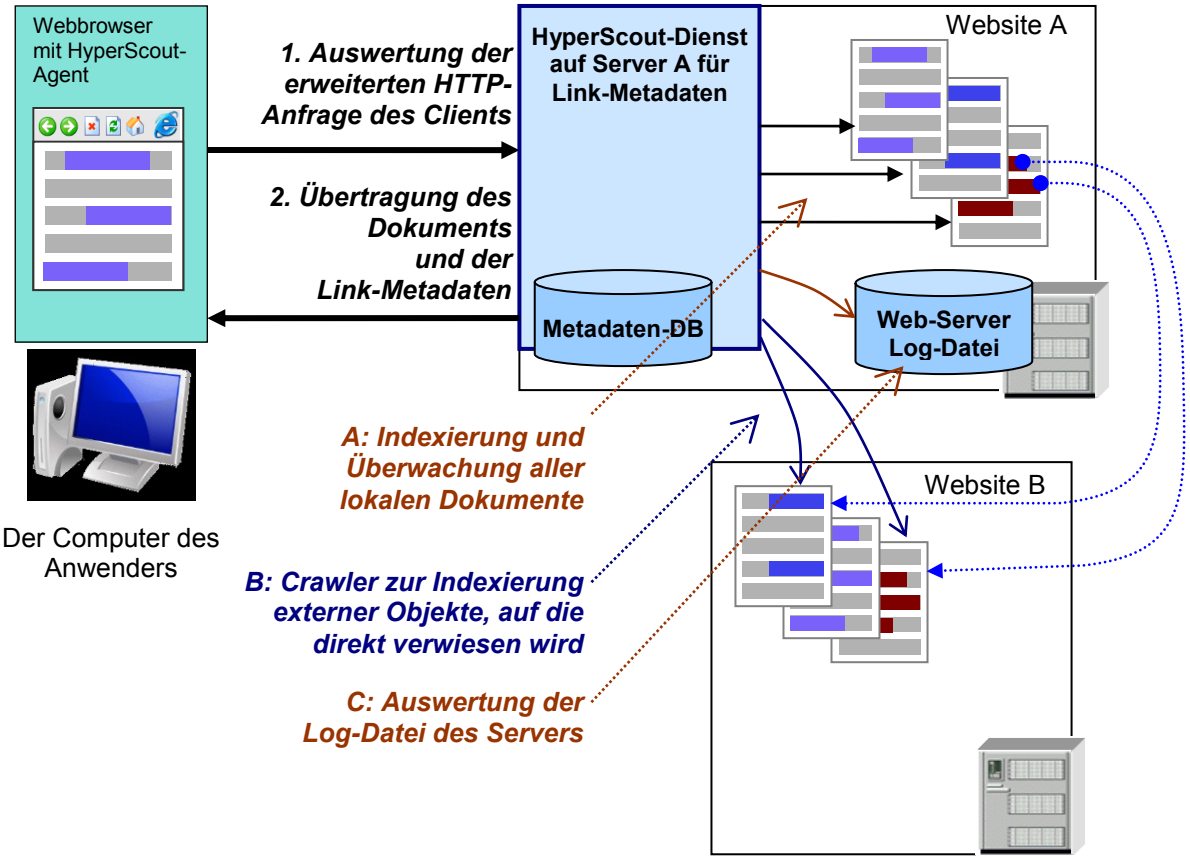


Abb. 115: Schematische Funktionsweise des serverseitigen Metadaten-Dienstes.

7.4 Protokolle und Sprachen zur Übertragung erweiterter Link-Daten

Eine konzeptionelle Herausforderung des im vorherigen Abschnitt 7.3 vorgestellten Konzepts stellt die effiziente Übertragung der zusätzlichen Link-Metadaten vom serverseitigen Metadaten-Dienst zum Browser bzw. dem HyperScout-Agenten dar. Bisher sehen weder HTTP noch HTML Möglichkeiten zur Bereitstellung der in Abschnitt 4.5 vorgestellten zusätzlichen Link-Informationen vor. Dies macht es notwendig, ein *Protokoll* und eine *Sprache* zur Übermittlung der zusätzlichen Daten zu entwickeln oder, ausgehend von existierenden Standards, entsprechende Erweiterungen zu spezifizieren.

Als Grundlage zur Übermittlung von Hyperlink-Daten ließen sich beispielsweise die Protokolle früherer offener Hypertext-Systeme wie Microcosm/DLS (s. Anhang B.13) und Hyper-G (s. Anhang B.14) heranziehen, die zusätzliche *typisierte* Hyperlinks für beliebige Dokumente zur Verfügung stellten (Anderson 1997). Obwohl sie aufgrund ihrer Offenheit für beliebige Dokumentformate geeignet sind, stellen sie für den Einsatz im Web keine optimale Wahl dar, weil sie unter anderem Schwächen bei der Skalierbarkeit aufweisen (Hall 1997) und nicht mit den anderen Techniken des Webs kompatibel sind. Beides spricht dafür, für HyperScout auf aktuellen Standards des W3C aufzubauen.

Für das *Protokoll* fiel die Wahl daher relativ schnell auf das omnipräsente HTTP. Es ist gut erweiterbar und eignet sich für die Übertragung beliebiger Dateiformate. Zudem unterstützen Web-Clients und Server dieses Protokoll, und es liegen Bibliotheken in jeder gängigen Programmiersprache vor. Des Weiteren sind kaum praktische Probleme zu erwarten, wie sie sich bei neuen Protokollen beispielsweise mit Firewalls oder Routern ergeben können: Ist der Zugriff mit dem Browser auf eine Website möglich, so lassen sich in der Regel auch beliebige andere Datenformate per HTTP abrufen. Seine Flexibilität stellt das Protokoll unter anderem bei den *Web Services* unter Beweis, wo es zusammen mit *SOAP* (Gudgin, Hadley et al. 2007) und *XML Schema* (Biron, Permanente et al. 2004; Thompson, Beech et al. 2004) sogar genutzt wird, Programmparameter auszutauschen und entfernte Prozeduren aufzurufen.[197] Abgesehen von den in Abschnitt 3.3.1 aufgeführten Schwächen bezüglich der Effizienz, stellt es somit eine gute Basis für verteilte Informationssysteme dar. Für das HyperScout-Konzept wurde das Protokoll um Kommandos zur Anforderung der zusätzlichen Link-Informationen ergänzt, damit nur die benötigten Daten übertragen werden. Die entsprechende HTTP-Extension wird in Abschnitt 7.5.3 vorgestellt.

Als *Sprache* zur Repräsentation der zusätzlichen Link-Informationen eignen sich mehrere Standards des W3C. Drei hervorzuhebende Sprachen sind *RDF*, *XLINK* und *XHTML*.

7.4.1 RDF und REST

Das *Resource Description Framework (RDF)*[198] wurde im Rahmen der *Semantic Web Initiative* entworfen, beliebige „Web-Ressourcen" mit Metadaten zu versehen (Beckett 2004). Einzige Voraussetzung für den Einsatz von RDF ist, dass die Ressource per URI adressierbar ist. Demzufolge können mithilfe der sogenannten „*Ressource-Property-Statement*"*-Tripel* von RDF beliebige zusätzliche Attribute für alle (per URI adressierten) *Zielobjekte* von Web-Hyperlinks ausgedruckt werden (s. Abb. 116 oben).

Eigenschaften der HTML-Hyperlinks selbst – beispielsweise ein *Link-Typ* oder eine *Link-Aktion* – sind hingegen nicht so unmittelbar per URI zu adressieren, da die Link-Anker *in* den (X)HTML-Code einer Seite eingebettet sind. Es wird ein Adressierungsschema benötigt, das es erlaubt, sowohl das Dokument als auch die Anker-Elemente im Dokument zu identifizieren.

---

[197] Zur Spezifikation der Web Services gehört auch *WSDL* als Metasprache zur Definition der Funktionalität und Schnittstelle eines Web Services (Christensen, Curbera et al. 2001) sowie UDDI, das primär als Protokoll für Namens- und Verzeichnisdienste dient (Clement, Hatley et al. 2004).

[198] RDF ist nicht zu verwechseln mit RDFa, das einen festen Satz an Attributen für XHTML-Dateien definiert, mit dem sich Webseiten (bzw. beliebige XML-Dateien) mit semantischen Annotationen versehen lassen. Dies lässt sich beispielsweise von speziellen Programmen und Suchmaschinen auswerten. Die Möglichkeiten von RDFa sind für die hier vorgestellten Konzepte zu unflexibel (Adida, Birbeck et al. 2008).

Eine Sprache zur Adressierung von Elementen *in* XML-Dokumenten (also auch in XHTML-Seiten) ist XPath (s. Anhang C.3.2). XPath-Ausdrücke können aber vergleichsweise komplex sein und müssten jeweils aktuell berechnet werden, falls sich Aufbau oder Inhalt eines Dokuments ändern.

Eine einfachere Alternative, die weniger anfällig für Änderungen in der Dokumentstruktur ist, besteht aus einer Konkatination der URI des Dokuments und der Adresse des Link-Zieles, getrennt durch ein reserviertes Symbol. Im folgenden Beispiel (Abb. 116 unten) wird das Rautenzeichen „#“ als Trennsymbol verwendet.[199]

Neben der Adressierung der Link-Anker als *RDF-Ressourcen* sind zweitens zur Beschreibung der einzelnen Link-Attribute *URNs* als *Properties* zu definieren. Eine Spezifikation der Properties für das HyperScout-Konzept findet man in Abschnitt 7.5.1.

Drittens geben die sogenannten *Statements* der RDF-Ausdrücke die Werte der Attribute an. Sie können als Literal oder URN dargestellt werden. Literale eigenen sich für textliche Werte, wie den Titel und das letzte Änderungsdatum einer Seite, URNs sind z. B. für die Auswahl aus einer Menge vorgegebener Eigenschaften prädestiniert. Abb. 116 zeigt zwei Beispiele für den Einsatz von RDF zur Übertragung der HyperScout-Metadaten. Dieser Ansatz eignet sich ebenfalls für einen autonomen Metadaten-Dienst, da die Link-Informationen unabhängig von der Dokumentstruktur übertragen werden können.

Ein RDF-Ausdruck, der Erstellungsdatum und Titel der Ressource „http://www.conftool.net/“ spezifiziert:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://www.hyperscout.org/RDF">
<rdf:Description rdf:about="http://www.conftool.net/">
  <hs:hreftitle>ConfTool: Conference Management Software</hs:hreftitle>
  <hs:hrefcreationdate>Tue, 13 Dec 2005 09:12:31 GMT</hs:hrefcreationdate>
</rdf:Description>
</rdf:RDF>
```

Folgender Ausdruck weist darauf hin, dass der Link von der Ressource http://www.weinreichs.de/ auf die Ressource http://www.conftool.net/ ein neues Fenster öffnet (charakterisiert durch eine *Property-URN* für *Popup-Links*):

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://www.hyperscout.org/RDF">
<rdf:Description rdf:about="http://www.weinreichs.de/#http://www.conftool.net/">
  <hs:hreflinkaction>http://www.hyperscout.org/RDF/popup</hs:hreflinkaction>
</rdf:Description>
</rdf:RDF>
```

Abb. 116: Zwei Beispiele für die Formulierung der HyperScout-Metadaten in RDF.

[199] Das *Rautenzeichen* oder *Hash-Symbol* ist der sogenannte „Fragment Identifier“ der URI, mit dem sich Positionen innerhalb von Webseiten adressieren lassen. In Analogie wird dies hier auch genutzt, allerdings um Link-Anker in Seiten mittels der URI zu adressieren.

Die Anfrage an den Metadaten-Dienste kann beispielsweise über eine *REST-Schnittstelle*[200] erfolgen (s. Fielding 2000: Kapitel 5), dem man per GET den URI des aktuellen Dokuments übergibt und der dann für alle enthaltenen Links die Metadaten per RDF zurückliefert.

7.4.2 XLink

Eine zweite Sprache, um zusätzliche Link-Informationen zu Web-Dokumenten hinzuzufügen, ist die *XML Linking Language*: XLink-Ausdrücke können unterschiedliche durch Menschen oder Maschinen lesbare Typinformationen enthalten. Allerdings reichen die Ausdrucksformen von XLink bei weitem nicht aus, um die vielen zusätzlichen Link-Attribute des HyperScout-Konzepts zu repräsentieren. Überdies wurde XLink konzipiert, um unabhängige *zusätzliche* Verknüpfungen zu Dokumenten in XML-konformen Formaten zu definieren, und nicht, um existierende, eingebettete Verweise in XHTML-Seiten mit weiteren Informationen anzureichern (detaillie Informationen findet man in Anhang C.3). Dieser Ansatz wurde daher nicht weiter verfolgt.

7.4.3 XHTML und HTTP

Die dritte Alternative besteht darin, eine der Sprachen zur Repräsentation von Web-Dokumenten zu erweitern. Als Grundlage eignet sich insbesondere der Standard XHTML 1.1 (McCarron & Ishikawa 2010), der mit dem Standard HTML 4.01 weitestgehend abwärtskompatibel ist, einen modularen Aufbau aufweist und sich durch zusätzliche Module, die als DTDs formuliert werden, ergänzen lässt (Altheim, Boumphrey et al. 2001; auf weitere Gründe für Wahl von XHTML 1.1 wird in Kapitel 7.5.2 eingegangen).

Für die Erweiterung der Seitenbeschreibungssprache XHTML sprechen für eine effiziente und skalierbare Lösung zur Übertragung zusätzlicher Link-Daten mehrere Faktoren: Die Anker-Elemente (vergl. Abschnitt 4.4.1ff) enthalten bereits alle grundlegenden Verknüpfungsinformationen und sie bieten bereits einige der benötigten Attribute für erweiterte Typinformationen an (s. Abschnitt 7.5.1), auch wenn diese kaum genutzt werden. Es wären nach demselben Schema lediglich neue Elemente oder Attribute zu (X)HTML zu ergänzen, um die zusätzlichen Link-Metadaten einzubetten. Eine Kompatibilität mit älteren Systemen wird gewährleistet, da Webbrowser für sie unbekannte Elemente und Attribute ignorieren. Das Beispiel in Abb. 117 zeigt die Einbettung zweier zusätzlicher Attribute mit Informationen zum Ziel des Links (den Dokumententitel und das Erstellungsdatum).

---

[200] REST steht für „Representational State Transfer“ und bezeichnet das grundlegende Konzept für Web-Anwendungen, Daten und Kommandos über die Standard-HTTP-Kommandos auszutauschen. Eine Sprache oder einen Dienst spezifiziert REST nicht, es geht vielmehr um das Paradigma, dass die bekannten HTTP-Kommandos nach einem bestimmten Schema auch für den Datenaustausch zwischen Servern eingesetzt werden.

```
<p>Ein Link zur Website
  <a href="http://www.conftool.net"
     hreftitle="ConfTool: Conference Management Software"
     hrefcreationdate="Tue, 13 Dec 2005 09:12:31 GMT">ConfTool</a>
</p>
```

Abb. 117: Ein Ausschnitt aus einem XHTML-Dokument mit erweiterter Syntax.

Ein weiterer Vorteil dieser Lösung besteht darin, dass die zusätzlichen Metadaten *gleichzeitig* mit dem Dokument übertragen werden. Somit sind die ergänzenden Link-Hinweise ohne Verzögerung verfügbar. Werden sie als neue Attribute des Anker-Elements definiert, entfällt zudem die Problematik von Adressierung und Zuordnung der zusätzlichen Meta-Information zu den Hyperlinks durch den Browser.

Diese Gründe waren ausschlaggebend dafür, XHTML 1.1 als Basis für die Übertragung der zusätzlichen Link-Metadaten zu verwenden. Die Definition des neuen HyperScout-XHTML-Sprachmoduls wird im Abschnitt 7.5.2 vorgestellt.

7.5 Die Protokoll- und Spracherweiterungen für das HyperScout-Konzept

In diesem Abschnitt werden erst die zusätzlich benötigten Attribute für das HyperScout-Konzept spezifiziert (s. Abschnitt 7.5.1) und darauf aufbauend eine Erweiterung der Sprache XHTML 1.1 (Abschnitt 7.5.2) und eine Extension des HTTP-1.1-Protkolls definiert (Abschnitt 7.5.3). Beide Definitionen sind kompatibel mit den existierenden Standards.

7.5.1 Definition der neuer Attribute für Link-Metadaten

Für die Übertragung der zusätzlichen Link-Informationen wurden 23 zusätzliche Attribute spezifiziert. Sie orientieren sich an der Klassifikation der impliziten Eigenschaften von Links und Knoten im Web aus Kapitel 4.5 und werden im Folgenden vorgestellt. Dabei sind die angegebenen Wertebereiche als Vorschläge zu sehen, die im Sinne der RFC 2119 eingehalten werden *„SOLLTEN“*[201] (Bradner 1997). Auf den Einsatz dieser Attribute wird in Kapitel 7.5.2 eingegangen.

[201] Die RFC 2119 definiert die technische Bedeutung von Begriffen wie „MUST“, „SHOULD“, „MAY“ für Spezifikationen. Sie werden zur Kennzeichnung ihrer Bedeutung in Großbuchstaben hervorgehoben. „SOLLTE“ bzw. „SHOULD“ steht dabei für eine Empfehlung, von der man nur in gut begründeten Fällen abweichen darf (Bradner 1997).

Der semantische Typ des Links (s. Abschnitt 4.5.1.1)

`linktype`	Mit diesem Attribut kann Links ein semantischer Typ zugewiesen werden. Vorschläge für sinnvolle Werte sind „navigational“ und „associative“, um die beiden wesentlichen Link-Typen im Web zu charakterisieren. Beide Typen können weiter detailliert werden, z. B. „navigational/up“ und „navigational/down“, zur Eingrenzung der Navigationsrichtung innerhalb der Struktur einer Website, sowie „associative/expansion“ und „associative/ressource“, um unterschiedliche assoziative Verknüpfungen zu beschreiben (vergl. Abschnitt 4.5.1.1).

Die Verteilungscharakteristik des Links (s. Abschnitt 4.5.1.2)

`linkdistribution`	Mithilfe dieses Parameters wird angegeben, ob ein Link innerhalb einer Site, zu einer anderen Site desselben Anbieters oder zu einem externen Angebot verweist (vergl. Catledge & Pitkow 1995). Sinnvolle Werte sind entsprechend: „within-site“, „within-realm“ (gleicher Anbieter aber andere Domain) und „between-site“.

Die Link-Aktion (s. Abschnitt 4.5.1.3)

`linkaction`	Mit diesem Attribut kann der Server die Link-Aktion übermitteln, da sie nicht eindeutig aus den anderen Anker-Attributen hervorgeht, beispielsweise wenn JavaScript eingesetzt wird. Gültige Werte sind „replacement“, „popup“, „note“, „reference“, „stretch“, „command“ (s. Abschnitt 4.5.1.3).

Informationen zum Inhalt des Zielobjekts (s. Abschnitt 4.5.2)

`hreftitle`	**Der Titel des Zieldokuments** Als Attributwert sollte der Inhalt des „`title`“-Elementes vom Link-Ziel übernommen werden.
`hrefdescription`	**Eine *Beschreibung* des Inhalts** Dieses Attribut soll eine vom Autor des Zielobjekts erstellte Beschreibung enthalten.
`hrefabstract`	**Eine kurze *Zusammenfassung* des Inhalts** Hier soll eine möglichst kurze Zusammenfassung des Dokuments zu finden sein. Im Gegensatz zu hrefdescription ist hierfür auch ein automatisch generierter Text zulässig.
`hrefauthor`	**Der *Autor*** Dieses Attribut soll den oder die Verfasser des Link-Ziels aufführen. Je nach Art des Objektes kann es aber auch sinnvoll sein, stattdessen den inhaltlich Verantwortlichen anzugeben.

`hrefprovider` **Der *Anbieter* der Website, der das Zielobjekt bereitstellt**

Im Gegensatz zum vorherigen Attribut ist hier der Anbieter der *gesamten* Website aufzuführen. Dies kann z. B. eine öffentliche Einrichtung, eine Firma oder auch eine Privatperson sein. Auf diese Weise soll der Benutzer den Kontext einer Webseite erkennen und Rückschlüsse auf die Qualität der Dokumente und die Interessen des Anbieters ziehen können.

`hrefcreationdate` **Der *Erstellungszeitpunkt***

Diese Information muss zumeist manuell vom Autor zu den Webseiten hinzugefügt werden und ist dann zumeist aus dem Dokument zu extrahieren. Es gibt entsprechende Meta-Elemente (Daviel 1996), die aber bisher wenig Verbreitung gefunden haben.

`hreflastmodified` **Der Zeitpunkt der *letzten Aktualisierung***

Diese Daten sollten ebenfalls vom Autor bereitgestellt werden. Zudem gibt es ein entsprechendes Attribut im HTTP-Protokoll, das das Datum der letzten Änderung einer Datei auf dem Server übermittelt.

`hreflang`[202] **Die *Sprache***

Mithilfe dieses Wertes können Benutzer auf potenzielle sprachliche Barrieren hingewiesen werden. Eine Angabe ist sinnvoll, wenn das Zieldokument in einer anderen Sprache vorliegt als das Ausgangsdokument und der Benutzer diese nicht beherrscht.

Der technische Typ des Zielobjektes (s. 4.5.3.1)

`hrefmimetypes`[203] **Der oder die MIME Media Types**

Dieses Attribut gibt den Dateityp des Link-Zieles als Mime-Type an. Bei Dokumenten, die andere Objekte einbinden – beispielsweise (X)HTML-Seiten – soll eine Liste der Mime-Types aller eingebundenen Objekte angehängt werden. So kann der Browser den Benutzer auf potenzielle Darstellungsprobleme hinweisen. Ein Beispiel:

```
hrefmimetypes="text/html, image/gif, image/png"
```

Der Medientyp des Zielobjektes (s. 4.5.3.2)

`hrefmediatype` **Der inhaltliche Medientyp**

Vorschläge für sinnvolle Werte sind „text/plain“, „text/illustrated“, „image“, „video“ und „audio“ (s. Nation 1998).

---

[202] Die Sprache des Zielobjektes kann bereits im XHTTP Hypertext Module als „`hreflang`“ spezifiziert werden. Sie wird hier aus Konsistenzgründen noch einmal aufgeführt, da diese Erweiterung einen anderen Namespace verwendet und die hier angebotenen Informationen auch automatisch generiert werden können.

[203] Der MIME Media Type des Zielobjektes lässt sich in HTML5 bereits durch das Anker-Attribut „`type`“angeben (Hickson 2011). Im Unterschied zu dem hier aufgeführten Attribut „`hrefmimetypes`“ darf es allerdings nur einen MIME-Type aufführen, wogegen das neue Attribut auch Aufzählungen von MIME-Types erlaubt; dies wird dem Charakter von Webseiten eher gerecht, da sie zumeist weitere Objekte einbinden.

Informationen zum topologische Typ des Zielobjektes (s. 4.5.3.3)

`hreftopologicaltype`

Der topologische Typ

Dieser Parameter soll den site-spezifischen[204] topologischen Typ des Zieldokuments angeben.

Sinnvolle Werte für dieses Attribut sind (vergl. Abschnitt 4.5.3.3):
- „homepage“: Startseite einer Website,
- „contentpage“: Seite mit primär inhaltlichem Charakter,
- „indexpage“: Navigationsseite mit primär internen Verweisen,
- „hubpage“: Seite mit vielen Verweisen auf externe Quellen.

`hreflocalinlinks`
`hreflocaloutlinks`
`hrefglobalinlinks`
`hrefglobaloutlinks`

Die Anzahl ein- und ausgehender Hyperlinks

Diese vier Attribute können alternativ genutzt werden, um topologische Eigenschaften einer Seite zu beschreiben. Es wird jeweils auf *site-spezifischer* und *site-übergreifender* Ebene die Anzahl der ein- und ausgehenden Links[205] des Zieldokuments angegeben. Benutzern kann auf dieser Basis eine personalisierte Auswertung angeboten werden.

Die Verfügbarkeit des Link-Zieles (s. Abschnitt 4.5.4.1)

`hrefstatus`

Dieser Parameter gibt den HTTP-Status-Code des Zielobjektes an, sofern der Code ungleich 200 („OK“) ist (Fielding, Gettys et al. 1999). Entsprechende Werte weisen auf passwort-geschützte Seiten und nicht zugreifbare Dokumente hin. So wird es unter anderem möglich, Benutzer vor defekten Links zu warnen.[206]

Daten zur Latenzzeit bei der Navigation (s. Abschnitt 4.5.4.2)

`hrefsize`

Die *Größe* des Zielobjektes

Dieses Attribut soll die Dateigröße des Zieldokuments aufführen, wie sie auch im HTTP-Feld „`content-length`“ angegeben wird (Fielding, Gettys et al. 1999). Dabei sind etwaige eingebundene Objekte nicht zu berücksichtigen (s. u. „`hreftotalsize`“). Der Wert gibt einen Hinweis auf die zu erwartende (minimale) Zeit, bis die ersten Teile einer Seite dargestellt werden.

---

[204] Dieses Attribut beschränkt sich auf den site-spezifischen Typ (s. Abschnitt 4.5.3.3), da sich site-übergreifende Typen wie Kleinbergs *Hubs* und *Authorities* immer auf bestimmte Stichworte beziehen (Kleinberg 1999).

[205] Die Anzahl der eingehenden Verweise von anderen Servern („`hrefglobalinlinks`“) lässt sich nur schätzen, da Verweise im Web unidirektional sind. Hierfür können die Daten von Suchmaschinen hinzugezogen oder das „`Referer`“-Attribut des Zugriffs-Protokolls des Webservers ausgewertet werden (Fielding, Gettys et al. 1999), das den URI der zuletzt besuchten Ressource übermittelt.

[206] Erfolgt unter der Adresse des Link-Zieles eine sofortige Umleitung auf eine andere Adresse (z. B. per HTTP-Status 301 bis 307), so soll das HyperScout-Attribut den Status des finalen Weiterleitungszieles enthalten. Da solche Umleitungen für den Benutzer kaum wahrnehmbar sind, ist das letztendlich angezeigte Objekt die für den Anwender relevante Information. Dies gilt ebenfalls für die anderen hier aufgeführten Attribute.

`hreftotalsize`	**Die *Größe* des Zielobjektes *inklusive* aller eingebetteten Objekte** Dieser Parameter ist für Formate wie (X)HTML relevant, die andere Ressourcen einbetten können. Es enthält die Summe der Dateigrößen des Hauptdokuments und aller eingebetteter Objekte. Der Wert gibt einen Hinweis auf die insgesamt zu erwartende Übertragungszeit.
`hrefdelay`	**Die *durchschnittliche Antwortzeit* des Servers in Millisekunden** Zur Ermittlung dieses Wertes kann z. B. eine HTTP-HEAD-Anfrage verwendet werden. Die Dauer der Antwort kann einen Hinweis auf die Belastung des Servers und die zu erwartende Übertragungszeit geben (vergl. Campbell & Maglio 1999). Dieser Wert ist allerdings nur aussagekräftig, wenn er zeitnah ermittelt wurde und sich das den Test durchführende System im selben Teilnetz befindet wie der Computer des Anwenders (s. auch Abschnitt 3.3.1). Ansonsten sollte der HyperScout-Agent auf eigene Methoden zur Bestimmung der Latenzzeit zurückgreifen.

Informationen zur Benutzung (s. 4.5.5.2)

`hrefpopularity`	**Die *Popularität* des Zieldokuments** Dieses Attribut ist ein Vorschlag für die Übertragung von *Social-Navigation Informationen* an den Client (s. Abschnitt 3.1.6). Der Inhalt dieses Attributes kann z. B. der prozentuale Wert der Personen sein, die das Zieldokument im Verhältnis zu allen anderen Seiten der Site angesehen haben (vergl. Wexelblat 1999).
`hrefreputation`	**Die *Reputation* des Zieldokuments** Mithilfe dieses Attributs kann die Reputation einer Website übertragen werden, die sich beispielsweise durch ein zentrales Empfehlungssystem bereitstellen kässt. Es sind 6 Werte vorgesehen: „excellent“, „good“, „ok“, „weak“, „bad“ und „unknown.

Diese Attribute dienen als Grundlage für die im Folgenden vorgestellten Erweiterungen von XHTML 1.1 und HTTP 1.1. Obwohl bei der Herleitung der Attribute systematisch vorgegangen wurde (s. Kapitel 4.5), ist es wahrscheinlich, dass der praktische Einsatz weitere Optimierungen und Konkretisierungen mit sich bringen wird: Die Evaluationen von *HyperScout I* und *II* zeigten bereits, dass die die Anforderungen der Benutzer an die Link-Schnittstelle sehr unterschiedlich sein können (s. Kapitel 6.2ff und 6.5ff).

7.5.2 Deklaration des HyperScout-Moduls in XHTML 1.1

HTML- als auch XHTML-Sprachen werden durch sogenannte DTDs definiert. **HTML 4.01** gilt seit 1999 als der offizielle Standard für die Definition von Webseiten im HTML-Format (Raggett, Hors et al. 1999). Der Sprachstandard **XHTML 1.0** ging aus HTML 4.01 hervor und war weitestgehend kompatibel, aber im Gegensatz zum Vorgänger eine *XML-konforme* Spra-

che (Pemberton 2002, 4; Münz & Gull 2010), entsprechende Webseiten lassen sich in Folge durch XML-Werkzeuge verarbeiten.

Der wesentliche Unterschied zwischen XHTML 1.0 und seinem Nachfolger **XHTML 1.1** liegt darin, dass die DTD von Version 1.1 *modularisiert* ist und sich die Sprache dadurch einfach erweitern lässt (Altheim, Boumphrey et al. 2001; McCarron & Ishikawa 2010).

Die Modularisierung wurde in **XHTML 2.0** noch weiter vorangetrieben (Altheim, Boumphrey et al. 2010; Axelsson, Epperson et al. 2010), allerdings ist XHTML 2.0 nicht mehr mit HTML 4.01 oder XHTML 1.1 kompatibel und wird von keinem populären Browser unterstütz. Die Entwickler hatten XHTML 2.0 nicht als Weiterführung von HTML betrachtet, sondern als ein *Neubeginn*, der die Schwächen der Vorgänger vermeidet. Aufgrund der mangelnden Akzeptanz wurde die Arbeit an XHTML 2.0 jedoch im Dezember 2010 vor Vollendung der Spezifikation eingestellt (Axelsson, Epperson et al. 2010). XHTML 2.0 wird daher in dieser Arbeit nicht weiter berücksichtigt.

Die zur Zeit in der Entwicklung befindliche Sprache **HTML5** und das dazu kompatible XML-Format **XHTML5** haben einen anderen Fokus: Bei der 2004 angefangenen Entwicklung[207] von HTML5 ging es darum, eine einheitliche Basissprache und standardisierte Schnittstellen zum Zugriff auf damit erstellte Dokumente zu schaffen, um die Realisierung von interaktiven Web-Anwendungen zu vereinfachen. Interaktivität ist ein Schwerpunkt von HTML5, so werden 2D- und 3D-Ausgaben als auch die Einbettung von Audio- und Video-Medien direkt berücksichtigt (vergl. Abschnitt 9.2). Eine Erweiterbarkeit der Sprachen wird gegenwärtig[208] eher am Rande thematisiert und eine Modularisierung ist bisher nicht geplant (Hickson 2011). Es werden zwar zahlreiche Möglichkeiten für die Erweiterung der Dokumente vorgeschlagen,[209] gleichzeitig wird jedoch von einigen abgeraten („strongly discouraged“) und andere sind nur im Zusammenhang mit JavaScript interessant (Hickson

[207] Die Entwicklung von HTML5 wurde im Jahr 2004 von einer Gruppe namens *WHATWG* („Web Hypertext Application Technology Working Group“) bestehend aus Mitarbeitern mehrerer Browserhersteller und Suchmaschinenbetreiber eingeleitet. Ursprünglich hatten sie den Namen *Web Applications 1.0* für ihre Sprache vorgesehen. Erst seit 2008 wird die Entwicklung gemeinsam mit dem W3C unter dem Namen HTML5 fortgeführt. Mehr Informationen findet man unter: http://www.whatwg.org/.

[208] Diese Aussage bezieht sich auf den Stand des Entwurfes vom 7. Oktober 2011 (Hickson 2011). Fragen zur Modularisierung und Erweiterbarkeit von HTML5 werden zusätzlich auf der FAQ-Seite zu XHTML thematisiert: http://www.w3.org/2009/06/xhtml-faq.html.

[209] Zu den empfohlenen Erweiterungsmöglichkeiten gehören *Microformats*, ein Ansatz, der existierende Attribute wie „`class`“ und „`rel`“ verwendet, um XHTML5-Dateien mit semantischen Metadaten anzureichern. Zusätzlich wird die Verwendung von sogenannten „`data-`“-Attributen vorgeschlagen, die aber vor allem für die Verarbeitung durch Skripte vorgesehen sind (Hickson 2011).

2011). Die Fertigstellung der Spezifikation (der Status „Recommendation“) ist für Dezember 2014 vorgesehen.[210]

Als Basis für die HyperScout-Spracherweiterung wurde **XHTML 1.1** gewählt, da die endgültige Spezifikation der Sprache seit November 2010 vorliegt (McCarron & Ishikawa 2010) und sie den besten Kompromiss zwischen Kompatibilität mit dem heutigen Web und Erweiterbarkeit der Sprache bietet.[211] Das Ende dieses Abschnitts geht auf die (voraussichtlichen) Möglichkeiten in XHTML5 ein, da sich bereits jetzt abzeichnet, dass diese Sprache zukünftig ebenfalls große Relevanz haben wird.

Die Einbettung der im vorhergehenden Abschnitt 7.5.1 eingeführten Link-Attribute in Web-Dokumente erfolgt in XHTML 1.1 mithilfe eines zusätzlichen Moduls. Im Folgenden wird erst das sogenannte *Hypertext Module* von XHTML 1.1 vorgestellt und danach das hierauf aufbauende neu entwickelte *HyperScout-Modul* für XHTML definiert.

Das Hypertext-Modul von XHTML 1.1

Die Elemente und Attribute zum Ausdruck von Hyperlinks in XHTML 1.1 sind in einem eigenen Teilmodul – dem sogenannten *Hypertext Module* – deklariert.[212] Es ist entsprechend den *assoziativen* Verknüpfungsmöglichkeiten von XHTML recht simpel und umfasst als einziges Element das Anker-Element „`a`“ mit den zulässigen Attributen. Zur Deklaration des *HyperScout-Moduls* wird zuvor die Definition des Hypertext-Moduls vorgestellt.

Der erste Teil der Definition in Abb. 118 beschreibt das „`a`“-Element: Zuerst wird eine entsprechende XML-Entity für das Element definiert und angegeben, das es beliebige Zeichen[213] und die meisten XHTML-Elemente (außer dem „`a`“-Element) umschließen kann.[214] Die Entity „`a.qname`“ repräsentiert die möglichen Attribute des Anker-Elementes. Sie ist folglich für die Definition der HyperScout-Attribute von besonderem Interesse.

---

[210] Der Zeitplan der für die Entwicklung von (X)HTML5 verantwortliche HTML Working Group ist unter folgender Adresse zu finden: http://www.w3.org/2007/03/HTML-WG-charter.html.

[211] XHTML 1.1 wird von allen aktuellen Browsern interpretiert und im Zuge der Gestaltung barrierefreier Websites werden Dokumente häufig in diesem XML-konformen Format angeboten.

[212] XHTML5 ist im Gegensatz zu XHTML 1.1 nicht modularisiert. Das Anker-Element wurde einfach den Elementen zur semantischen Beschreibung von Text („Text-level semantics“) untergeordnet. Hierzu gehören auch Elemente wie „em“ (hervorgehobener Text, meist kursiv dargestellt), „strong“ (Text mit großer Bedeutung) und „br“ (für Zeilenumbruch) (Hickson 2011).

[213] PCDATA steht für “parsed character data” und bedeutet, dass der Text geparst werden soll und beispielsweise *Entities* wie *ö* in das entsprechende Zeichen „ö“ ungewandelt werden.

[214] Die Entity „`ImlNoAnchor.mix`“ fasst alle Teilmodule, bis auf das Hypertext-Modul, von XHTML zusammen.

```
<!-- XHTML Hypertext Module  ..................................... -->

<!-- 1. Definition des a-Elementes -->
<!ENTITY % a.element  "INCLUDE" >
<![%a.element;[
<!ENTITY % a.content
    "( #PCDATA | %InlNoAnchor.mix; )*"
>
<!ENTITY % a.qname  "a" >
<!ELEMENT %a.qname;  %a.content; >
<!-- end of a.element -->
]]>

<!-- 2. Definition der Attribute -->
<!ENTITY % a.attlist  "INCLUDE" >
<![%a.attlist;[
<!ATTLIST %a.qname;
      %Common.attrib;
      href         %URI.datatype;           #IMPLIED
      charset      %Charset.datatype;       #IMPLIED
      type         %ContentType.datatype;   #IMPLIED
      hreflang     %LanguageCode.datatype;  #IMPLIED
      rel          %LinkTypes.datatype;     #IMPLIED
      rev          %LinkTypes.datatype;     #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
      tabindex     %Number.datatype;        #IMPLIED
>
]]>
```

Abb. 118: Das Hypertext-Teilmodul von XHTML 1.1 (ohne Kommentare, siehe: McCarron & Ishikawa 2010).

Die zulässigen Attribute des „a“-Elementes werden im zweiten Teil der Definition von Abb. 118 aufgeführt: Die Entity „Common.attrib“ steht für die Menge der Attribute, die in *allen* XHTML-Elementen benutzt werden können, wie beispielsweise „id“, „title“ und „class“. Es folgen die acht spezifischen Attribute des Elementes, wobei die Entities „URI.datatype“, „Charset.datatype“ etc. die unterschiedlichen Variablentypen von XHTML repräsentieren. Diese Definitionen werden auch im HyperScout-Modul verwendet.

Das HyperScout-Modul für XHTML 1.1

Zur Erweiterung von XHTML um ein eigenes Modul müssen mehrere DTDs mit unterschiedlichen Aufgaben definiert werden.[215] Dies sind erstens ein *Qualified-Name*-Teilmodul, zweitens ein oder mehrere *Content-Model*-Teilmodule und drittens die *DTD Driver Declaration.* In den DTDs der *Qualified-Name-* und *Content-Model-Module* finden die eigentlichen Sprachspezifikationen statt, die *Driver Declaration* führt die neuen Teilmodule mit den Modulen der XHTML-Definition zusammen.

Im Folgenden werden die drei für die HyperScout-XHTML-Spracherweiterung notwendigen DTDs vorgestellt. Obwohl sie umfangreich erscheinen mögen, ist die Definition des HyperScout-Moduls *vergleichsweise* einfach, da keine neuen Elemente definiert, sondern lediglich neue Attribute zu einem bereits existierenden Element hinzugefügt werden.

---

[215] Im Prinzip könnte eine solche Definition natürlich auch innerhalb einer einzigen DTD geschehen, aber XHTML 1.1 gibt die Aufteilung und Struktur der DTDs für XHTML-Extensions vor.

1. Das Qualified-Name-Teilmodul von HyperScout

Die Definition eines neuen XHTML-Moduls setzt voraus, dass zuerst in einem *Qualified Name-Teilmodul* (bzw. *QName-Teilmodul*) folgendes deklariert wird:

- Ein XML-Namensraum (Namespace),
- die Eigenschaften des Prefixing für den Namensraum sowie
- die gültigen Qualified Names (die zulässigen Elemente und Attribute).

Der Namensraum eines XHTML-Moduls wird mittels einer URN identifiziert. Für das HyperScout-Modul wird die in Abb. 119 angegebene URN verwendet.

```
http://www.hyperscout.org/xmlns/hyperscout
```

Abb. 119: Die URN des Namensraums (Namespace) des HyperScout-Moduls

Das *QName*-Teilmodul definiert zweitens ein entsprechendes Präfix, das die Zuordnung der Elemente und Attribute zum Modul ermöglich. Das Präfix des Namensraumes vermeidet, dass andere Module mit identischen Elementen oder Attributen hiermit in Konflikt geraten. Zudem vereinfachen Präfixe das Validieren und Parsen von Dokumenten. Allerdings zeigt die Definition dieses Teilmoduls auch, dass DTDs durch das *Prefixing* komplexer werden, zumal XHTML 1.1 aus Kompatibilitätsgründen erfordert, dass alle Elemente und Attribute solcher Erweiterungen auch ohne Präfix nutzbar sind (Altheim, Boumphrey et al. 2001).

Die Definition des Moduls in Abb. 120 beginnt mit der Deklaration der Entity „`HyperScout.xmlns`“, die den Namensraum des Moduls repräsentiert. Der zweite Abschnitt legt fest, ob als Standard für die Elemente und Attribute dieses Moduls Präfixe verwendet werden sollen. Da Präfixe im XHTML-Framewok im Normalfall nicht genutzt werden, lautet dieser Wert „IGNORE“, d. h., Präfixe werden auch im HyperScout-Modul in der Regel nicht eingesetzt. Diese Angabe kann im Dokument überschrieben werden (vergl. Abb. 123 und Abb. 124).

Es folgt drittens die Entity „`HyperScout.prefix`“, die das *Standard-Präfix* des Moduls repräsentiert. Er lautet „hs“. Der vierte Teil definiert die weitere Entity „`HyperScout.pfx`“, die als „Hilfsvariable“ für das *Prefixing* sämtlicher Elemente und Attribute des Moduls verwendet wird. Abhängig davon, ob Präfixe aktiviert sind oder nicht, enthält die Entity das Standard-Präfix des Namensraumes, einen im Dokument definierten alternativen Präfix oder einen Leerstring. Somit können alle Elemente und Attribute des Moduls mit und ohne Prefixing verwendet werden. Abschnitt fünf definiert in ähnlicher Weise – ebenfalls abhängig vom Einsatz des *Prefixing* – das *XML Namespace*-Attribut für dieses Modul. Danach wird das entsprechende Namespace-Attribut für XHTML-Elemente deklariert, das natürlich denselben Wert wie das vorige Attribut hat.

```
<!-- ..................................................................... -->
<!-- XHTML HyperScout Qname Module ..................................... -->
<!-- file: hyperscout-qname.mod

     PUBLIC "-//HYPERSCOUT//ELEMENTS XHTML HyperScout Qnames 1.0//EN"
     SYSTEM "http://www.hyperscout.org/DTD/hyperscout-qname.mod"

     xmlns:hyperscout="http://www.hyperscout.org/xmlns/hyperscout"

     ..................................................................... -->

<!-- 1. Deklaration des Namensraumes für das Modul.                      -->
<!ENTITY % HyperScout.xmlns "http://www.hyperscout.org/xmlns/hyperscout" >

<!-- 2. Festlegung, ob im Regelfall Präfixe verwendet werden sollen.    -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % HyperScout.prefixed "%NS.prefixed;" >

<!-- 3. Definition des Standard-Präfixes.                               -->
<!ENTITY % HyperScout.prefix "hs" >

<!-- 4. Definition einer Entity für das optionale Prefixing.            -->
<![%HyperScout.prefixed;[
<!ENTITY % HyperScout.pfx "%HyperScout.prefix;:" >
]]>
<!ENTITY % HyperScout.pfx "" >

<!-- 5. Deklaration des XML-Namespace-Attributes für dieses Modul.      -->
<![%HyperScout.prefixed;[
<!ENTITY % HyperScout.xmlns.extra.attrib
    "xmlns:%HyperScout.prefix; %URI.datatype; #FIXED '%HyperScout.xmlns;'" >
]]>
<!ENTITY % HyperScout.xmlns.extra.attrib "" >

<!-- 6. Angabe des Namespace Attributes für XHTML-Elemente.             -->
<!ENTITY % XHTML.xmlns.extra.attrib "%HyperScout.xmlns.extra.attrib;" >

<!-- 7. Deklaration der "öffentlichen" Elemente und Attribute           -->

<!ENTITY % HyperScout.a.linktype.qname            "%HyperScout.pfx;linktype">
<!ENTITY % HyperScout.a.linkdistribution.qname    "%HyperScout.pfx;linkdistribution">

<!ENTITY % HyperScout.a.linkaction.qname          "%HyperScout.pfx;linkaction">

<!ENTITY % HyperScout.a.hreftitle.qname           "%HyperScout.pfx;hreftitle">
<!ENTITY % HyperScout.a.hrefdescription.qname     "%HyperScout.pfx;hrefdescription">
<!ENTITY % HyperScout.a.hrefabstract.qname        "%HyperScout.pfx;hrefabstract">
<!ENTITY % HyperScout.a.hrefauthor.qname          "%HyperScout.pfx;hrefauthor">
<!ENTITY % HyperScout.a.hrefprovider.qname        "%HyperScout.pfx;hrefprovider">
<!ENTITY % HyperScout.a.hrefcreationdate.qname    "%HyperScout.pfx;hrefcreationdate">
<!ENTITY % HyperScout.a.hreflastmodified.qname    "%HyperScout.pfx;hreflastmodified">
<!ENTITY % HyperScout.a.hreflang.qname            "%HyperScout.pfx;hreflang">

<!ENTITY % HyperScout.a.hrefmimetypes.qname       "%HyperScout.pfx;hrefmimetypes">
<!ENTITY % HyperScout.a.hrefmediatype.qname       "%HyperScout.pfx;hrefmediatype">

<!ENTITY % HyperScout.a.hreftopologicaltype.qname "%HyperScout.pfx;hreftopologicaltype">
<!ENTITY % HyperScout.a.hreflocalinlinks.qname    "%HyperScout.pfx;hreflocalinlinks">
<!ENTITY % HyperScout.a.hreflocaloutlinks.qname   "%HyperScout.pfx;hreflocaloutlinks">
<!ENTITY % HyperScout.a.hrefglobalinlinks.qname   "%HyperScout.pfx;hrefglobalinlinks">
<!ENTITY % HyperScout.a.hrefglobaloutlinks.qname  "%HyperScout.pfx;hrefglobaloutlinks">

<!ENTITY % HyperScout.a.hrefstatus.qname          "%HyperScout.pfx;hrefstatus">

<!ENTITY % HyperScout.a.hrefsize.qname            "%HyperScout.pfx;hrefsize">
<!ENTITY % HyperScout.a.hreftotalsize.qname       "%HyperScout.pfx;hreftotalsize">
<!ENTITY % HyperScout.a.hrefdelay.qname           "%HyperScout.pfx;hrefdelay">

<!ENTITY % HyperScout.a.hrefpopularity.qname      "%HyperScout.pfx;hrefpopularity">
<!ENTITY % HyperScout.a.hrefreputation.qname      "%HyperScout.pfx;hrefreputation">
```

Abb. 120: Definition des Qualified-Name-Teilmoduls von HyperScout

Zuletzt werden die *Qualified Names* des Moduls angegeben. Dies sind die Elemente und Attribute des Moduls, die öffentlich sind, also direkt in die XHTML-DTD eingebunden werden. Normalerweise findet man hier die Wurzelelemente des Moduls, aber in diesem Falle werden lediglich die neuen Attribute aufgeführt, da sie alle für das bereits existierende „a"-Element des XHTML-Hypertext-Moduls vorgesehen sind. Die zuvor definierte Entity „HyperScout.pfx" sorgt nun dafür, dass ein Präfix automatisch vor jedes Attribut gestellt wird, wenn *Prefixing* im Dokument gefordert wird.

2. Das Content Model Modul: Ein Teilmodul zur Deklaration der Elemente und Attribute

Zur konkreten Spezifikation der neuen Elemente und Attribute des HyperScout-XHTML-Moduls wird ein weiteres Teilmodul benötigt: Dieses sogenannte *Content-Model-Module* legt die Syntax der Elemente und Attribute fest.

Da das HyperScout-Modul lediglich zusätzliche *Attribute* definiert, die zudem alle demselben Element zugeordnet sind, ist ein einzelnes derartiges Teilmodul zur Deklaration ausreichend (s. Harold & Means 2002).

Die Definition eines *Content-Model-Moduls* (Abb. 121) besteht aus mehreren Abschnitten: Zuerst wird (normalerweise) ein Namespace-Attribut deklariert, das innerhalb des Moduls u.a. für die Attribute der neu deklarierten Elemente benötigt wird. Danach wird die Syntax der neuen Elemente des Moduls definiert, und es werden die Attributlisten der Elemente des Moduls aufgeführt. Da das HyperScout-Modul keine neuen Elemente definiert, sind diese beiden Bereiche in diesem Falle leer (Altheim, Boumphrey et al. 2001; Altheim, Boumphrey et al. 2010).

Stattdessen werden hier die für die HyperScout-Attribute verwendeten XHTML-Datentypen eingebunden (s. Abb. 121, Teil 3), die im XHTML-Teilmodul „xhtml-datatypes.mod" definiert sind. Zwei Beispiele für die so eingebundenen Datentypen sind „Datetime.datatype", das ein Datum im ISO-Format repräsentiert, und „Number.datatype", das für eine beliebige Zahl steht (Altheim, Boumphrey et al. 2001).

Nun werden die neuen Attribute für das „a"-Element spezifiziert (s. Abb. 121, Teil 4). Da die Definition von Attributen an beliebiger Stelle innerhalb der DTDs geschehen kann und mehrere Attributlisten dazu führen, dass diese Attribute gemeinsam genutzt werden können, wird einfach die Entity „a.qname" aus dem XHTML-Hypertext-Modul (s. Anfang dieses Abschnitts) übernommen, um die neuen Attribute zu deklarieren.

```
<!-- ..................................................................... -->
<!-- HyperScout Content Model Module   ............................... -->
<!-- file: hyperscout-1.mod

     PUBLIC "-//HYPERSCOUT//ELEMENTS XHTML HyperScout 1.0//EN"
     SYSTEM "http://www.hyperscout.org/DTD/hyperscout-1.mod"

     xmlns:hyperscout="http://www.hyperscout.org/xmlns/hyperscout"
     ..................................................................... -->

<!-- 1. Definition des Namespaces für die Attribute                       -->

<!-- 2. Spezifikation der neuen Elemente und Attributlisten des Moduls. -->

<!-- 3. Einbinden der Entities für die XHTML-Datentypen                  -->
<!ENTITY % xhtml-datatypes.mod
   PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
   "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod">
%xhtml-datatypes.mod;

<!-- 4. Deklaration von Attributen für externe Elemente                  -->
<!ATTLIST %a.qname;
  %HyperScout.a.linktype.qname             %Text.datatype;           #IMPLIED
  %HyperScout.a.linkdistribution.qname     %Text.datatype;           #IMPLIED
  %HyperScout.a.linkaction.qname           %Text.datatype;           #IMPLIED

  %HyperScout.a.hrefmimetypes.qname        %ContentTypes.datatype;[216] #IMPLIED
  %HyperScout.a.hrefmediatype.qname        %Charset.datatype;        #IMPLIED
  %HyperScout.a.hreftopologicaltype.qname  %Text.datatype;           #IMPLIED
  %HyperScout.a.hreflocalinlinks.qname     %Number.datatype;         #IMPLIED
  %HyperScout.a.hreflocaloutlinks.qname    %Number.datatype;         #IMPLIED
  %HyperScout.a.hrefglobalinlinks.qname    %Number.datatype;         #IMPLIED
  %HyperScout.a.hrefglobaloutlinks.qname   %Number.datatype;         #IMPLIED

  %HyperScout.a.hreftitle.qname            %Text.datatype;           #IMPLIED
  %HyperScout.a.hrefauthor.qname           %Text.datatype;           #IMPLIED
  %HyperScout.a.hrefprovider.qname         %Text.datatype;           #IMPLIED
  %HyperScout.a.hrefcreationdate.qname     %Datetime.datatype;       #IMPLIED
  %HyperScout.a.hreflastmodified.qname     %Datetime.datatype;       #IMPLIED
  %HyperScout.a.hrefdescription.qname      %Text.datatype;           #IMPLIED
  %HyperScout.a.hrefabstract.qname         %Text.datatype;           #IMPLIED
  %HyperScout.a.hreflang.qname             %LanguageCode.datatype;   #IMPLIED
  %HyperScout.a.hrefstatus.qname           %Number.datatype;         #IMPLIED
  %HyperScout.a.hrefsize.qname             %Number.datatype;         #IMPLIED
  %HyperScout.a.hreftotalsize.qname        %Number.datatype;         #IMPLIED
  %HyperScout.a.hrefdelay.qname            %Number.datatype;         #IMPLIED
  %HyperScout.a.hrefpopularity.qname       %Text.datatype;           #IMPLIED
  %HyperScout.a.hrefreputation.qname       %Text.datatype;           #IMPLIED
>
```

Abb. 121: Definition des Content-Model-Teilmodul von HyperScout

Die aufgeführten Entities der Form „HyperScout.a.hreftitle.qname“ sind bereits im *Qualified-Name*-Teilmodul definiert worden (s.o.) und stehen für den Namen des Attributes mit und ohne das optionale Präfix. An dieser Stelle müssen daher nur noch die *Datentypen* und die *Vorgaben*[217] für die einzelnen Attribute konkretisiert werden.

---

[216] Im Gegensatz zum regulären Anker-Attribut „type“ ist der Datentyp dieses Attributes nicht „ContentType“, sondern „ContentTypes“. Es handelt sich somit um eine durch Kommata getrennte Liste von MIME-Type-Angaben (Altheim, Boumphrey et al. 2001).

[217] Die Vorgaben legen fest, ob es sich um ein benötigtes Attribut handelt („REQUIRED“) oder nicht („IMPLIED“), ob ein Wert vorgegeben ist und ob dies sogar ein konstanter Wert ist („FIXED“). Alle Attribute der HyperScout-Erweiterung sind optional, also „IMPLIED“.

3. Der DTD Driver: Die DTD Dokumenttyp-Deklaration für das Modul

Die letzte Definition, die benötigt wird, um das neue HyperScout-Modul zum Einsatz zu bringen, ist das *DTD-Driver-Modul*. Diese DTD führt die neuen Teilmodule mit der XHTML-Definition zusammen.

```
<!—
    Dies ist die einzubindende Driver DTD für die
    Hyperscout-Erweiterungen an XHTML 1.1

    Diese DTD kann mittels der folgenden Schlüssel identifiziert werden:
    PUBLIC "-//HyperScout//DTD XHTML11-Hyperscout//EN"
    SYSTEM "http://www.hyperscout.org/DTD/hyperscout-1.dtd"
-->

<!-- 1. Spezifikation eines Bezeichners für die DTD dieses Moduls      -->
<!ENTITY % HyperScout.version  "-//HyperScout//DTD XHTML11-Hyperscout//EN" >

<!-- 2. Einbinden der XHTML 1.1 DTD                                    -->
<!ENTITY % xhtml11.dtd
    PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
%xhtml11.dtd;

<!-- 3. Einbinden des HyperScout Qualified Name Moduls                 -->
<!ENTITY % hyperscout-qname.mod
   PUBLIC "-//HYPERSCOUT//ELEMENTS XHTML HyperScout Qnames 1.0//EN"
   "http://www.hyperscout.org/DTD/hyperscout-qname.mod">
%hyperscout-qname.mod;

<!-- 4. Einbinden des HyperScout Content Model Moduls                  -->
<!ENTITY % hyperscout.mod
   PUBLIC "-//HYPERSCOUT//ELEMENTS XHTML HyperScout 1.0//EN"
   "http://www.hyperscout.org/DTD/hyperscout-1.mod">
%hyperscout-1.mod;
```

Abb. 122: Die Definition des DTD-Driver-Moduls von HyperScout

Die *Driver DTD* definiert zunächst den Identifikator zur Adressierung des Moduls (vergl. Beispiele in Abb. 123 und Abb. 124).

Zweitens wird die normale XHTML 1.1 DTD eingebunden. Da die HyperScout-XHTML-Erweiterung abwärtskompatibel zu XHTML 1.1 sein soll, werden alle Definitionen der Ausgangssprache integriert. Nun werden die beiden zuvor definierten Teilmodule hinzugefügt, erst das *QName-Modul* und dann das *Content-Model-Modul*. Damit ist die formale Definition des neuen HyperScout-Gesamtmoduls abgeschlossen.

Zur Verwendung des neuen Moduls muss lediglich die neue *HyperScout Driver DTD* als Dokumenttyp angegeben werden. Konkret heißt dies, dass ein solches HyperScout-XHTML-Dokument nicht mehr mit der folgenden Zeile für „normale“ XHTML-Dokumente beginnt:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
                  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Stattdessen ist am Anfang des Dokuments die HyperScout-DTD anzugeben:

```
<!DOCTYPE html PUBLIC "-//HyperScout//DTD XHTML11-Hyperscout//EN"
                  "http://www.hyperscout.org/DTD/hyperscout-1.dtd">
```

Ein entsprechend erweiterter Browser mit HyperScout-Agent kann dank dieses Parameters ein Dokument mit den neuen zusätzlichen Link-Metadaten erkennen und korrekt parsen.

Beispiele zur Nutzung der HyperScout-XHTML-Erweiterung

Zwei Beispiele illustrieren die Verwendung der HyperScout-Erweiterung mit und ohne Namensräume.

Das erste Beispiel in Abb. 123 verwendet keine XML-Namensräume (Namespaces) und nutzt lediglich das neue Anker-Attribut „`hreftitle`“. Die ersten beiden Zeilen geben den neuen Dokumenttyp an. Die hervorgehobene Zeile weiter unten zeigt, wie das neue Attribut verwendet werden kann. Weitere HyperScout-Attribute sind in äquivalenter Weise einfach zum öffnenden Tag des Anker-Elementes hinzuzufügen.

```
<!DOCTYPE html PUBLIC "-//HyperScout//DTD XHTML11-Hyperscout//EN"
                      "http://www.hyperscout.org/DTD/hyperscout-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
  <head>
    <title>Ein beispiel-Dokument</title>
  </head>
  <body>
    <p>Ein Link zur Website des Projekts
      <a href="http://www.scone.de"
         hreftitle="Scone Web Enhancement Framework"> Scone
        </a>
    </p>
  </body>
</html>
```

Abb. 123: Ein HyperScout-XHTML-Dokument mit einem zusätzlichen Link-Attribut.

```
<!DOCTYPE html PUBLIC "-//HyperScout//DTD XHTML11-Hyperscout//EN"
                      "http://www.hyperscout.org/DTD/hyperscout-1.dtd" [
    <!ENTITY % NS.prefixed "INCLUDE">
    <!ENTITY % XHTML.prefix "xhtml" >
    <!ENTITY % HyperScout.prefix "hs">
]>
<xhtml:html xmlns:xhtml="http://www.w3.org/1999/xhtml"
            xmlns:hs="http://www.hyperscout.org/xmlns/hyperscout">
  <xhtml:head>
    <xhtml:title>Ein Beispiel-Dokument, das Präfixe verwendet</title>
  </xhtml:head>
  <xhtml:body>
    <xhtml:p>Ein Link zur Website des Projekts
      <xhtml:a xhtml:href="http://www.scone.de"
               hs:hreftitle="Scone Web Enhancement Framework"> Scone
      </xhtml:a>
    </xhtml:p>
  </xhtml:body>
</xhtml:html>
```

Abb. 124: Ein HyperScout-XHTML-Dokument, das Namespace-Prefixing einsetzt.

Das HyperScout-Modul erlaubt – ebenso die XHTML 1.1 – die Verwendung von *Namespace-Prefixing*. Das zweite Beispiel in Abb. 124 zeigt, wie dasselbe Dokument mit Präfixen aussehen kann. Die dritte Zeile gibt an, dass für dieses Dokument *Prefixing* aktiviert werden soll. Dazu wird der Wert von „`NS.prefixed`“ (Standardwert „IGNORE“) mit „INCLUDE“ überschrieben. Die vierte und fünfte Zeile sind optional und geben die Zeichenketten für die Präfixe der entsprechenden Namespaces an. Die zwei hervorgehobenen Zeilen im unteren

Bereich verdeutlichen, wie das „a“-Element mit zwei Attributen unterschiedlicher Namensräume (dem Standard-XHTML-Namensraum und dem des HyperScout-Moduls) bei der Verwendung von *Prefixing* aussieht.

Die Nutzung von Namensräumen ist auch für Erweiterungen von **XHTML5** vorgesehen. Allerdings haben einige populäre Browser wie der Internet Explorer bis Version 8 hiermit noch Probleme, da sie Namensraum-Präfixe nicht (vollständig) unterstützen. Dennoch könnten solche Dokumente in näherer Zukunft durchaus eine Rolle spielen, zumal sie bei anderen XML-Anwendungen schon seit vielen Jahren gang und gäbe sind.

HTML5 (ohne „X“)unterstützt die Nutzung von Namensräumen nicht. Stattdessen wird im Entwurf[218] vorgeschlagen, neue Attribute durch das voransetzen des Schlüsselwortes „data-“ (z. B. `data-hreftitle`) zu kennzeichnen oder bei anbieterspezifischen Erweiterungen den String „`x-anbietername-`“ (z.B. „`x-hs-hreftitle`“) voranzusetzen. Allerdings ist die erste Methode (nur) für Daten vorgesehen, die durch Skripte verarbeitet werden sollen und für beide Methoden ist bisher nicht eindeutig festgelegt, wie man die Erweiterung und den entsprechenden HTML5-Sprachdialekt formal definiert.

7.5.3 Definition einer Extension für das HTTP-Protokoll

Damit Web-Clients und Server sich über die zusätzlich benötigten bzw. zur Verfügung stehenden Link-Informationen verständigen können, wurde im Rahmen dieser Arbeit das HTTP-Protokoll erweitert. Dies soll den unterschiedlichen Bedürfnissen und Vorlieben der Anwender Rechnung tragen und erlaubt es, gezielt die benötigten Meta-Informationen anzufordern. So wird vermieden, dass überflüssige Daten übertragen werden und die Performanz des Systems unnötig leidet.

Zu diesem Zwecke sind sowohl die Anfrage (*HTTP-Request*) des Clients als auch die Antwort (*HTTP-Response*) des Servers erweitert worden. Als Basis dient die aktuelle Version 1.1 des HTTP-Protokolls (Fielding, Gettys et al. 1999) sowie das *HTTP Extension Framework*[219] (Nielsen, Leach et al. 2000). Bei der Spezifikation wurden bereits existierende optionale HTTP-Parameter wie *Accept-Encoding*, *Accept-Language* und *Accept-Charset* berücksichtigt, mit deren Hilfe der Client dem Server mitteilen kann, welche Dateiformate er verarbeiten kann bzw. von ihm bevorzugt werden. In Analogie kann ein HyperScout-Agent angeben, welche Link-Informationen übertragen werden sollen und in welchem Format er akzeptiert. Wie im

[218] Dies ist der Stand des HTML5-Entwurfes vom 9. Dezember 2011 (Hickson 2011).

[219] Das HTTP Extension Framework bietet gegenüber diversen anderen Erweiterungen von HTTP den Vorteil, dass alle zusätzlichen Parameter im HTTP-Header übertragen werden und somit beispielsweise keine POST-Parameter notwendig sind. Dies gewährleistet unter Anderem, dass solche Anfragen auch in Caching Proxys (vergl. Abschnitt 8.4.2.1) zwischengespeichert werden. Andere Erweiterungen des Webs wie SOAP verwenden daher inzwischen optional auch das HTTP Extension Framework (vergl. Gudgin, Hadley et al. 2007, Abschnitt 7).

Web üblich ist aber dem Server die „Entscheidung“ vorbehalten, welche Daten er tatsächlich übermittelt. Dieses Vorgehen wird als „*Server-driven Negotiation*“ bezeichnet, da der Client lediglich seine Präferenzen angibt, der Server aber die entsprechende Auswahl trifft, wobei er versucht, die Anfrage des Clients möglichst optimal zu erfüllen, ohne aber dem Client explizit mehrere Optionen anzubieten.[220]

Der HTTP Extension Identifier

Zur Definition einer HTTP-Extension ist als Erstes ein URN (Uniform Resource Name) festzulegen, der die Protokollerweiterung identifiziert (siehe Abb. 125 und die Beispiele in Abb. 129 und Abb. 130).

```
http://www.hyperscout.org/HTTP/httpextension-1
```

Abb. 125: Der URN der HyperScout-HTTP-Extension

HTTP Request Extension Headers

Für die Anfrage des HyperScout-Agenten war lediglich ein zusätzlicher HTTP-Parameter zu definieren. Er führt die gewünschten Link-Attribute in Form einer Liste auf. Fehlt dieser Parameter, so sollen keine zusätzlichen Daten durch den Server bereitgestellt werden, da davon auszugehen ist, dass der Benutzer entweder keine ergänzenden Link-Hinweise wünscht oder er einen konventionellen Web-Client nutzt, der die Attribute nicht auswerten kann. So wird unnötiger zusätzlicher Datenverkehr durch die HyperScout-Erweiterung vermieden.

Der neue Parameter für HTTP-Requests heißt *Accept-Anchor-Attributes*. Die Definition des zusätzlichen HTTP-Kommandos (siehe Abb. 126) erfolgt mithilfe der erweiterten Backus-Naur-Form[221] (angelehnt an RFC 822, siehe: Crocker 1982), die auch zur Spezifikation von HTTP dient (Fielding, Gettys et al. 1999).

Die möglichen Werte für das Nichtterminalsymbol „`attribute-names`“ sind die im Abschnitt 7.5.1 definierten neuen XHTML-Attribute. Das Nichtterminalsymbol „`parameter`“ in Zeile 3 steht für mögliche Konkretisierungen einzelner Attribute, dessen optionale Werte für spätere Erweiterungen vorgesehen sind und im Rahmen dieser Arbeit nicht spezifiziert wurden.

---

[220] Wird stattdessen vom Server eine Auswahl angeboten und entscheidet der Client über die Parameter, so spricht man von „*Agent-driven Negotiation*“. Dieses Vorgehen wird im Web nicht favorisiert, da es ein zusätzliches *Request-Response-Paar* benötigt, wodurch sich die Übertragung verzögert.

[221] Als kurze Erläuterung der Syntax: Das Symbol # wird verwendet, um eine durch Kommata getrennte Liste von Elementen zu spezifizieren, wobei optional vor dem Hash-Zeichen die minimale und nach ihm die maximale Anzahl von Listenelementen aufgeführt werden kann. Wird keine Zahl angegeben, so bedeutet dies, dass beliebig viele Elemente möglich sind.

```
Accept-Anchor-Attributes = "Accept-Anchor-Attributes" ":"
                              #( attributes )
              attributes = ("*" | attribute-names ) *(";" parameter)
          attribute-names = ("linktype" | "linkdistribution" |
                                    "linkaction" | "hrefmimetypes" |
                                    "hrefmediatype" | "hreftopologicaltype" |
                                    "hreflocalinlinks" | "hreflocaloutlinks"|
                                    "hrefglobalinlinks" |
                                    "hrefglobaloutlinks" | "hreftitle" |
                                    "hrefauthor" | "hrefprovider" |
                                    "hrefcreationdate" | "hreflastmodified" |
                                    "hrefdescription" | "hrefabstract" |
                                    "hreflang" | "hrefstatus" |
                                    "hrefsize" | "hreftotalsize" |
                                    "hrefdelay" | "hrefpopularity" | "hrefreputation")
```

Abb. 126: Definition des *Accept-Anchor-Attributes* für die HyperScout-HTTP-Erweiterung.

Abb. 127 zeigt vier Beispiele für den neuen HTTP-Parameter, der als zusätzliches Attribut an den Server übermittelt wird. Das erste Beispiel weist den Server an, die drei zusätzlichen Attribute „`hreftitle`“, „`hrefstatus`“ und „`hrefcreationdate`“ zu jedem Anker-Element hinzuzufügen. In der zweiten Anfrage wird kein zusätzliches Attribut gewünscht, wogegen Zeile drei alle verfügbaren Link-Metadaten anfordert. Das letzte Beispiel stellt eine mögliche Nutzung ergänzender Parameter vor: Der Client möchte neben dem Titel eine Beschreibung des Zieldokuments von höchstens 100 Zeichen Länge erhalten.

```
1) Accept-Anchor-Attributes: hreftitle, hrefstatus, hrefcreationdate

2) Accept-Anchor-Attributes:

3) Accept-Anchor-Attributes: *

4) Accept-Anchor-Attributes: hreftitle, hrefdescription;maxlength=100
```

Abb. 127: Vier Beispiele für das neue HTTP-Request-Attribut *Accept-Anchor-Attributes*.

HTTP Response Extension Headers

Die Antwort des Servers führt auf, welche der angeforderten Attribute tatsächlich in das Dokument eingefügt wurden und welches Alter die zu Grunde liegenden Daten haben. Auf diese Weise kann der Metadaten-Dienst dem HyperScout-Agenten bereits im Protokoll mitteilen, in wieweit die Anforderungen des Benutzers erfüllt werden. Der Agent kann dann sofort bestimmen, ob er selbst weitere Informationen einholen soll oder nicht.

Es wurden zwei zusätzliche HTTP-Response-Parameter definiert (siehe Abb. 128): *Added-Anchor-Attributes* listet die zusätzlich im XHTML-Dokument eingefügten Link-Attribute auf, und *Last-Update* teilt dem Client mit, wann der Server letztmalig die Daten auf Konsistenz überprüft hat. Dieser Wert soll den Aktualisierungszeitpunkt der *ältesten* übermittelten Daten wiedergeben. Auf diese Weise lässt sich abschätzen, mit welcher Wahrscheinlichkeit die angebotenen zusätzlichen Informationen konsistent sind und ob sie den Anforderungen des Benutzers entsprechen.

```
Added-Anchor-Attributes = "Accept-Anchor-Attributes" ":"
                          #( attributes )
             attributes = ("*" | attribute-names ) *(";" parameter)
        attribute-names = ("linktype" | "linkdistribution" |
                                  "linkaction" | "hrefmimetypes" |
                                  "hrefmediatype" | "hreftopologicaltype" |
                                  "hreflocalinlinks" | "hreflocaloutlinks"|
                                  "hrefglobalinlinks" |
                                  "hrefglobaloutlinks" | "hreftitle" |
                                  "hrefauthor" | "hrefprovider" |
                                  "hrefcreationdate" | "hreflastmodified" |
                                  "hrefdescription" | "hrefabstract" |
                                  "hreflang" | "hrefstatus" |
                                  "hrefsize" | "hreftotalsize" |
                                  "hrefdelay" | "hrefpopularity" | "hrefreputation")

            Last-Update  = "Last-Update" ":" HTTP-date
```

Abb. 128: Die Definition der zwei neuen HTTP-Response-Attribute

Man und Opt Headers

Das HTTP Extension Framework unterscheidet zwischen *Man* und *Opt* Headers. Ersteres steht für *mandatory*, es handelt sich also um Pflichtangaben, die vom Server entsprechend zu berücksichtigen sind. *Opt Headers* sind hingegen *optionale* Angaben, die vom Server ignoriert werden können. Eine Spezifikation des Header-Typs ist für jede HTTP-Extension erforderlich.

Die Kommandos der HyperScout-HTTP-Extension sollen als ***Opt* Headers** verwendet werden, da die Link-Metadaten lediglich eine wünschenswerte Erweiterung (also optional) sind. Server, die die Anfrage nach XHTML-Dokumenten mit den gewünschten zusätzlichen Attributen nicht erfüllen können, sollen die Antwort keinesfalls ganz verweigern,[222] sondern das Originaldokument zur Verfügung stellen.

Ein Beispiel der HyperScout-HTTP-Erweiterung in der Praxis

Im folgenden Beispiel wird die Ressource „`index.html`“ von der Site „`www.informatik.uni-hamburg.de`“ abgefragt. Sowohl Client als auch Server verfügen im Beispiel über die entsprechenden Module für die HyperScout-Erweiterungen.

Der Client verlangt drei zusätzliche Link-Meta-Informationen: die Titel der Zieldokumente („`hreftitle`“), das Datum der jeweils letzten Änderung („`hreflastmodified`“) sowie die HTTP-Status („`hrefstatus`“) für alle Link-Ziele.

Eine Anfrage des Clients wird in Abb. 129 dargestellt. Die ersten beiden Zeilen sind gewöhnliche HTTP-Kommandos, die dem Webserver mitteilen, welches Objekt von welcher Web-

[222] Kann ein Server eine *mandatory* HTTP-Extension nicht interpretieren oder die geforderten Bedingungen nicht erfüllen, so ist die Anfrage mit dem HTTP-Fehlercode 510 „HTTP/1.1 510 Not Extended“ zu beantworten.

site[223] benötigt wird. Die dritte Zeile definiert, dass die optionale HyperScout-HTTP-Extension verwendet wird. Hierzu wird die URN der Erweiterung angegeben und für die Attribute der Extension ein dynamisch vergebener *Namespace* als *Header-Präfix*[224] spezifiziert („ns=13“). Die vierte Zeile enthält die eigentliche Anfrage mit dem Präfix. Sie führt drei zusätzliche Anker-Attribute auf.

```
GET /index.html HTTP/1.1
Host: www.informatik.uni-hamburg.de
Opt: "http://www.hyperscout.org/HTTP/httpextension-1"; ns=13
13-Accept-Anchor-Attributes: hreftitle, hreflastmodified, hrefstatus
```

Abb. 129: Der HTTP-Request des erweiterten Web-Clients.

Da es sich um eine optionale Erweiterung handelt, könnte eine gewöhnliche HTTP-Response des Servers erfolgen, selbst wenn der Server ein Dokument mit zusätzlichen Link-Metadaten zur Verfügung stellt. Der HyperScout-Dienst *soll* dem Client aber bereits im HTTP-Protokoll mitteilen, dass er die zusätzlichen Kommandos interpretieren konnte und berücksichtigt hat. Somit erhält der Client die Möglichkeit, den Benutzer unmittelbar auf die Verfügbarkeit zusätzlicher Attribute hinzuweisen oder selbst weitere Daten anderweitig, beispielsweise per *Prefetching* (s. Abschnitt 7.2.1), anzufordern. Die Antwort des Servers mit dem erweiterten HTTP-Protokoll wird in Abb. 130 dargestellt.

```
HTTP/1.1 200 OK
Content-Type: application/xhtml+xml
Date: Wed, 20 Apr 2011 08:12:31 GMT
Expires: Wed, 20 Apr 2011 08:22:31 GMT
Opt: "http://www.hyperscout.org/HTTP/httpextension-1"; ns=27
27-Added-Anchor-Attributes: hreftitle, hreflastmodified, hrefstatus
27-Last-Update: Wed, 20 Apr 2011 06:18:08 GMT
```

Abb. 130: Die HTTP-Response des erweiterten Webservers.

Die ersten vier Zeilen sind wiederum gewöhnliche HTTP-Kommandos. Der Server teilt als erstes mit, dass die Anfrage erfolgreich beantwortet werden konnte (HTTP-Code 200) und dem Web-Client nun eine XHTML-Datei[225] übermittelt wird. Die dritte und vierte Zeile geben die aktuelle Uhrzeit des Servers an und spezifizieren den Zeitpunkt, an dem das übergebene Objekt “ungültig” wird, d. h. wie lange es in einem Proxy oder dem Browser-Cache zwischengespeichert werden darf, bevor es bei wiederholter Darstellung neu übertragen werden muss. Dieser Wert soll ebenfalls durch den serverseitigen HyperScout-Dienst änderbar sein: Da die Link-Iinformationen in das XHTML-Dokument eingebettet sind, kann es

[223] Einzelne Web-Server können mehrere Websites unter einer IP-Adresse beherbergen. Da sich in der TCP-IP-Anfrage lediglich die IP-Adresse des Servers, nicht aber der DNS-Name befindet, wird im HTTP-Header dem Server mitgeteilt, auf welche der Sites sich eine Anfrage bezieht.

[224] Dies ermöglicht es, mehrere Erweiterungen mit denselben Attributnamen innerhalb eines Headers zu verwenden, ohne dass es zu Mehrdeutigkeiten kommt.

[225] De facto geben die meisten Web-Server heute auch bei XHTML-Dokumenten noch „text/html“ als Mime-Type an. Dies ist eigentlich inkorrekt, wird aber bevorzugt, da so auch ältere Browser die Dokumente anzeigen.

notwendig sein, diesen Wert anzupassen, um durch eine erneute Übertragung der Seite auch die Link-Hinweise zu aktualisieren.

Die drei letzten Zeilen sind Nachrichten des serverseitigen HyperScout-Dienstes: Zuerst wird die Protokollerweiterung identifiziert und eine Nummer für den Namespace vergeben. Die folgende Zeile führt die zusätzlichen Attribute auf, die der HyperScout-Dienst in das Dokument einfügt. Die letzte Zeile gibt an, wann die übermittelten Link-Metadaten das letzte Mal aktualisiert wurden. In diesem Falle sind alle bereitgestellten Link-Informationen weniger als zwei Stunden alt.

7.6 Diskussion des vorgestellten technischen Konzepts

Das in diesem Kapitel vorgestellte technische Konzept zur Bereitstellung zusätzlicher Link-Informationen im Web kombiniert die Stärken der drei in Abschnitt 7.2 klassifizierten grundlegenden Ansätze: Es weist eine gute *Skalierbarkeit* auf, da die Übertragung und Konsistenzsicherung der Link-Informationen auf alle Server des Webs verteilt wird. Die *Performanz* ist für die Anwender mit der des bisherigen Webs vergleichbar, weil nur wenige zusätzliche Daten zum Web-Client zu übertragen sind und keine zusätzlichen Verbindungen zu weiteren Servern aufgebaut werden müssen. Die serverseitige Aufbereitung der Link-Informationen *minimiert den Datenumfang* und die Erweiterung des HTTP-Protokolls gewährleistet, dass nur die gewünschten Informationen übermittelt werden. Dank der in Abschnitt 7.5 vorgestellten Protokoll- und Spracherweiterungen werden die Link-Daten gemeinsam mit den Anker-Elementen in den Webseiten übertragen, sodass es keine Adressierungs- oder Synchronisationsprobleme gibt.

Das vorgestellte Konzept ist mit dem gegenwärtigen Web technisch *kompatibel*. Die entwickelten Protokoll- und Spracherweiterungen sind konform mit den Standards des W3C; Webserver und Browser, die die neuen Parameter nicht kennen, ignorieren sie. Gleichzeitig ist das Konzept problemlos zu realisieren, wie im folgenden Kapitel 8 ausgeführt wird.

Die Trennung von Inhalten und Präsentation ist ein wichtiges Mittel, um die Link-Informationen für den Anwender auf individuelle Weise zugänglich zu machen: Der HyperScout-Dienst stellt die Informationen im Textformat bereit, der Client sorgt für ihre Präsentation, beispielsweise als Tooltip, Maus-Icon oder Overlay, wie in den Kapiteln 5.5.2 und 6.4ff ausgeführt. Somit kann der Anwender entscheiden, welche Informationen angezeigt werden und auf welche Weise beispielsweise mit kritischen Status-Informationen wie fehlerhaften Links umzugehen ist. Die hier vorgestellten technischen Konzepte sind daher unabhängig von der Benutzungsschnittstelle des Web-Clients und unter anderem auch für Screen-Reader und Mobilgeräte mit Touch-Screen geeignet. Implizit erleichtert das vorgestellte Konzept sogar die Gestaltung *barrierefreier Websites* (BITV2 2011), weil Benutzer für alle Links Zugang

zu ergänzenden, textlichen Informationen erhalten. Dies ist z. B. nützlich, wenn der Link-Anker eine Grafik ist und ein Screen-Reader zum Einsatz kommt (s. Abschnitt 9.3.3).

Die zu erwartende zusätzliche Netzwerkbelastung *zwischen den Servern* ist ebenfalls gering, da lediglich Objekte zu übertragen sind, auf die von lokalen Dokumenten direkt verwiesen wird. Änderungen können in der Regel mit kurzen HTTP-HEAD-Requests überprüft werden. Durch die regelmäßige Kontrolle aller externen Links einer Website kann der HyperScout-Dienst einen Zustand schwacher Konsistenz („*weak consistency*“, siehe: Kappe 1995) gewährleisten, ohne dass ein zusätzliches Server-Server-Protokoll benötigt wird.

Berücksichtigt man den gesamten Internet-Datenverkehr, so relativiert sich der Umfang der zusätzlich für das HyperScout-System zu übertragenden Daten, zumal es sich um komprimierbare, textliche Parameter handelt. Multimediale Objekte (Bilder, Videos) machen in der Regel den Großteil der von Webservern transferierten Daten aus, wobei insbesondere der Anteil an Videodaten stetig zunimmt (Cisco 2011). Zudem verursachen bereits seit einigen Jahren Peer-to-Peer-Netzwerke wie *BitTorrent* und *eDonkey* zum Austausch von Musik-Dateien, Filmen und Programmen je nach Land zwischen 40 und 80 Prozent des Internet-Traffics (Schulze & Mochalski 2009).

Da heutzutage in den Industrienationen Arbeitszeit der Regel wesentlich teurer als Rechenzeit oder Bandbreite ist, sollte die Effizienz und Benutzbarkeit des Webs weitaus mehr Bedeutung haben als der zusätzliche Datentransfer: Benutzer können mit HyperScout irreführende oder fehlerhafte Links erkennen, ohne ihnen folgen zu müssen. Dies kann dazu beitragen, überflüssige Navigationsschritte zu vermeiden, wertvolle (Arbeits-)Zeit einzusparen und die kognitive Belastung zu reduzieren.

Eine in der Praxis auftretende Schwierigkeit ist, dass für ein effizientes Funktionieren der hier vorgestellten Konzepte nicht nur die Webbrowser erweitert werden müssen, sondern auch die Webserver. Da eine Anpassung der Webserver recht lange Zeit in Anspruch nehmen könnte, würde sich als Zwischenlösung ein zentraler Dienst anbieten, der die Abschnitt 7.4.1 vorgestellte Sprache auf Basis von RDF und REST einsetzen könnte, um zusätzliche Link-Informationen zu Webseiten zentral abzurufen. Kapitel 9.2.2 geht auf Lösungsmöglichkeiten für diese Problematik ein.

Im folgenden Abschnitt wird die technische Grundlage für die Implementation des hier vorgestellten Konzepts präsentiert. Das vom Autor dieser Arbeit entwickelte *internediary-basierte* Konzept und Framework eignet sich aufgrund seiner generischen Komponenten für viele unterschiedliche Erweiterungen des Webs.

8 Ein generischer Ansatz zur Realisierung von Web-Erweiterungen

Die Entwicklung neuer Konzepte zur Vereinfachung und Erweiterung der Navigations- und Orientierungsmöglichkeiten im Web stellt zahlreiche Herausforderungen. So muss die Benutzungsschnittstelle mit aktuellen Browsern und Websites harmonieren und Anwender bei der Nutzung des Webs wie vorgesehen unterstützen. Des Weiteren ist die technische Kompatibilität mit gegenwärtigen Systemen und Websites sicherzustellen. Weil neue Softwarekonzepte in der Praxis meist Schwächen aufweisen, ist es sinnvoll, sie früh anhand von Prototypen zu evaluieren (Nielsen 1993b), sodass Defizite lokalisiert und behoben werden können, bevor das Produktivsystem realisiert wird. Die Erstellung lauffähiger Software-Prototypen mit funktionsfähiger Benutzungsoberfläche ist jedoch ein zeitaufwendiger und kostenträchtiger Prozess,[226] insbesondere wenn keine geeigneten Frameworks oder Bibliotheken zur Verfügung stehen.

Diese Herausforderungen bestanden auch für die in den vorhergehenden Kapiteln vorgestellten HyperScout-Konzepte zur Vereinfachung der Link-Navigation. Die Evaluation der erweiterten Link-Benutzungsschnittstelle sollte mithilfe von funktionsfähigen Prototypen durchgeführt werden, um möglichst praxisnahe Bedingungen zu gewährleisten (s. Kapitel 5 und 6). Die in Kapitel 7 präsentierten Erweiterungen von HTTP und XHTML waren zudem als „Proof-of-Concept"-Systeme zu implementieren, um ihre Eignung für das Web zu belegen.

Eine längere Recherche ergab, dass keine geeignete technische Basis für die Realisierung solcher Web-Erweiterungen zur Verfügung stand. Dieses Defizit und die Erfordernis, für HyperScout mehrere Prototypen programmieren zu müssen, waren Motivation für die Entwicklung eines Frameworks, das diese Lücke schließt und auch anderen Forschern zur Verfügung steht.

Dieses Kapitel stellt die softwaretechnische Konzeption und Entwicklung des Frameworks *Scone* vor, das sich zur Erstellung von Prototypen für die Erweiterung der Navigationsmöglichkeiten des Web eignet. Das Framework verfolgt einen generischen Ansatz, damit es gleichzeitig den Anforderungen möglichst vieler Projekte aus dem Bereich der Navigations- und Orientierungsunterstützung im Web erfüllt.

Abschnitt 8.1 beschreibt als Erstes am Beispiel von HyperScout einige grundlegende Anforderungen an Web-Erweiterungen zur Vereinfachung der Orientierung und Navigation und begründet die Notwendigkeit zur Entwicklung eines eigenen Frameworks. Die Konzeption des Frameworks beginnt mit einer Anforderungsanalyse, die 20 Systeme zur Unterstüt-

[226] Analysen haben ergeben, dass die Benutzungsoberfläche häufig die Hälfte und mehr vom Quellcode und der Entwicklungszeit interaktiver Systeme ausmachen (Myers & Rosson 1992). Es ist kaum zu erwarten, dass der Anteil in naher Zukunft abnehmen wird, da die Anforderungen an Benutzungsschnittstellen immer weiter steigen werden (Petrasch 2007).

zung der Orientierung und Navigation im Web berücksichtigt. Hiervon ausgehend werden die Komponenten und Funktionen spezifiziert, die ein entsprechendes Framework bieten muss, damit sich auf seiner Basis Projekte zur Navigationsunterstützung im Web umsetzen lassen (s. Abschnitt 8.2).

Im nächsten Schritt werden die vorhandenen Schnittstellen zur Erweiterung von Webbrowsern und Servern verglichen, um die geeignetste Technik für die Realisierung des Frameworks festzustellen. Eine Analyse der Vor- und Nachteile der Schnittstellen (s. Abschnitt 8.3) führte zu dem Schluss, dass ein intermediary-basiertes Framework den Anforderungen am besten gerecht wird und die größte Flexibilität bietet. Teil 8.4 befasst sich mit den Konzepten von Web-Intermediaries: Erst werden sie gegenüber den bekannteren Web-Proxies abgegrenzt, dann ihre Einsatzbereiche spezifiziert. Es folgt eine Untersuchung verfügbarer *programmierbarer Intermediaries* bezüglich ihrer Eignung zur Realisierung des geplanten Frameworks.

Abschnitt 8.5 ordnet das neu konzipierte Framework *Scone* ein: Es handelt sich um ein in Java entwickeltes *Gray-Box-Framework*, das ein *Plug-in-Konzept* für die Programmierung der Erweiterungen nutzt. Die Architektur und Funktionsweise des Scone-Frameworks wird ab Kapitel 8.6 vorgestellt: Es basiert auf einem *programmierbaren Intermediary* (Abschnitt 8.6.1), einem *persönlichen Web-Crawler* (Abschnitt 8.6.2), einer Komponente zur *Repräsentation und Persistierung der Daten* (Abschnitt 8.6.3) sowie einem Modul zum *Erfassen der Benutzeraktionen mit dem Browser* (Abschnitt 8.6.4). Darüber hinaus bietet es zahlreiche *Hilfskomponenten*, unter anderem zur Bearbeitung der übertragenen Daten, zur Browser-Steuerung und zur Evaluation neuer Prototypen (Abschnitt 8.6.5). Teil 8.6.6 zeigt exemplarisch die Programmierung von Plug-ins für das Framework.

Kapitel 8.7 stellt Einsatzmöglichkeiten des Framework anhand mehrerer von unterschiedlichen Forschern entwickelter Prototypen vor. Es folgt eine Diskussion der Potenziale und Grenzen des Frameworks (Teil 8.8). Abschnitt 8.9 befasst sich mit kürzlich entwickelten alternativen Techniken zur Erweiterung von Webbrowsern, und der letzte Teil (Kapitel 8.10) zieht eine kurze Bilanz des Scone-Projekts.

8.1 Voraussetzungen für die Realisierung von Navigationserweiterungen des Webs

Bereits bei der Konzeption der ersten HyperScout-Prototypen zeigte sich, dass solche Systeme zur Vereinfachung der Link-Navigation im Web eine umfangreiche Funktionalität aufweisen müssten. Drei der grundlegenden Anforderungen für die Realisierung von HyperScout waren:

- Die Erweiterung der Link-Benutzungsschnittstelle mit Tooltips, Maus-Icons und Link-Overlays erfordert eine dynamische *Modifikation aller im Browser dargestellten Dokumente* (siehe Kapitel 5 und 6).

- Für die erweiterte Link-Schnittstelle werden *zusätzliche Daten* benötigt, die *automatisch* aus den Link-Ankern der Ausgangsseite als auch aus allen Ressourcen, auf die die Links der Seite verweisen, zusammenzustellen sind.
- Die in Kapitel 7 vorgestellten Sprach- und Protokollerweiterungen erforderten Anpassungen des *HTTP-Protokolls* und der transferierten *HTML-Dokumente* sowohl auf *Client-* als auch auf *Serverseite*.

Diese Anforderungen stellten besondere Herausforderungen an die Implementation, die sich beispielsweise nicht mittels eines Browser-Plug-ins meistern ließen (s. Abschnitt 8.3.1). Darüber hinaus mussten sich die HyperScout-Prototypen für partizipative Benutzbarkeitstests ohne störende technische Probleme eignen, woraus eine hohe softwaretechnische Qualität und Praxistauglichkeit als wichtige Eigenschaft für die Prototypen folgte. Ferner sollten sie unabhängig von Plattform, Web-Client und Webserver sein, um systemtechnische Einschränkungen für die Tests zu minimieren und realitätsnahe Testbedingungen zu gewährleisten, bei denen die Teilnehmer ihren gewohnten Browser verwenden und auf beliebige existierende Websites zugreifen könnten.

Damit all diese Anforderungen und Ziele innerhalb eines vertretbaren Zeitraumes zu bewerkstelligen waren, sollte auf ein Framework oder Bibliotheken zurückgegriffen werden, die einen Großteil der benötigten Funktionalität bereitstellten. Eine längere Recherche ergab, dass kein passendes Framework existierte und die verfügbaren Bibliotheken nicht ausreichten, um als Basis für die HyperScout-Prototypen zu dienen (siehe Abschnitt 8.3).

Ein Vergleich mit zahlreichen verwandten Projekten, die ebenfalls eine Unterstützung der Navigation und Orientierung im Web zum Schwerpunkt haben, zeigte, dass deren Entwickler offensichtlich mit derselben Problematik konfrontiert waren: Alle untersuchten Systeme sind im Wesentlichen Eigenentwicklungen (siehe Abschnitt 8.2). Dieses Vorgehen weist den Nachteil auf, dass bereits die Realisierung der Systeme sehr aufwendig ist und daher häufig im Rahmen der Projektzeit keine Evaluation der entwickelten Konzepte mehr stattfinden kann. Da Projekte aus diesem Forschungsbereich aber in der Regel das Ziel verfolgen, die *Benutzbarkeit* des Webs zu verbessern, sollte auf Benutzbarkeitsstudien nicht verzichtet werden.

Diese Ausgangssituation wies auf das große Potenzial hin, das ein entsprechendes Framework auch für andere Forscher haben könnte. Das in diesem Kapitel vorgestellte Framework *Scone* sollte daher nicht nur zur Programmierung der *HyperScout-Prototypen* dienen, sondern sich auch für andere Projekte mit ähnlichen Anforderungen und Zielsetzungen eignen. Die folgende Anforderungsermittlung zur Funktionalität von Scone berücksichtigt daher neben HyperScout auch zahlreiche verwandte Projekte.

8.2 Anforderungen an ein Framework zur Erweiterung der Navigationsmöglichkeiten im Web

Um den Herausforderungen möglichst vieler Projekte zur Unterstützung der Navigation und Orientierung im Web gerecht zu werden, wurden die technischen Anforderungen von

zwanzig Projekten und Systemen aus den Bereichen der Navigations- und Orientierungsunterstützung im Web untersucht (s. Tabelle 4). Diese Analyse bildete die Grundlage für die Identifikation der wichtigsten Komponenten und Funktionen, die ein Software-Framework bieten muss, um für den vorgesehenen, recht umfangreichen Einsatzbereich geeignet zu sein. Die identifizierten Anforderungen wurden klassifiziert und unter Berücksichtigung anderer Projekte zur Erweiterung des Webs (Bouvin 1999; Cockburn & Greenberg 1999; Malandrino & Scarano 2005) in den folgenden 10 Kriterien zusammengefasst.

8.2.1 Filtern und Ändern von Ressourcen

Die erste technische Anforderung an die Funktionalität des Erweiterungs-Frameworks ist die Manipulation der im Browser dargestellten Objekte. Viele Projekte versuchen die Benutzbarkeit des Webs zu verbessern, indem sie Web-Ressourcen anpassen, optimieren oder ergänzen. Beispielsweise können auf Webseiten automatisch weiterführende Informationen und Interaktionsmöglichkeiten zur Vereinfachung der Orientierung und Navigation angeboten werden (siehe Tabelle 4, Spalte 1, Projekte: *CoIn*, *Foot*, *GrLa*, *HypS*, *SGA*, *TOC*, *WPH*) oder zusätzliche Links auf hilfreiche Dokumente oder Anmerkungen einbinden (Projekte: *CrLi*, *DLS*, *Vise*).

Zur Realisierung entsprechender Prototypen sollte das Framework über eine API[227] verfügen, die sowohl den lesenden als auch schreibenden Zugriff auf die Objekte im Browser-Fenster bzw. auf die zum Browser übertragenen Daten bietet (realisiert im *Scone Intermediary*, siehe Abschnitt 8.6.1). Eine einfache Bearbeitung der Objekte setzt voraus, dass der Zugriff nicht nur auf Bytecode-Ebene, sondern auch in interpretierten Datenformaten möglich ist (siehe Anforderung 8.2.3).

[227] Die Software-Schnittstelle eines Systems wird auch als *API* für *Application Programming Interface* bezeichnet.

Tabelle 4: Überblick über die technischen Anforderungen von 20 Projekten[228] zur Unterstützung der Navigation und Orientierung im Web

Projekt		Art[229]	1: Filtern und Ändern von Ressourcen	2: Zugriff auf zusätzliche Ressourcen	3: Interpretation und Analyse von Web-Ressourcen	4: Erfassen von Aktionen und Browserstatus	5: Steuern des Browsers	6: Mehrbenutzer-unterstützung	7: Aufzeichnen von Benutzeraktionen	8: Repräsentieren und Persistieren von Ressourcen	9: Erweiterung der Schnittstelle des Browsers
BaLi	Backlink Browser Extension (Chakrabarti, Gibson et al. 1999)	Client	Ø	Backlink-Listen von Such-maschinen	Extraktion des Seitentitels	Neue Seite	Seite laden	Ø	Besuchte Seiten	Metadaten zu Webseiten, Benutzeraktionen	Fenster neben dem Browser: 1. Hierarch. History 2. Back-Link-Liste
CoB	CoBrow (Sidler, Scott et al. 1997)	Work-group	Einbetten eines Applets zur Anzeige anderer Besucher und zum Chat	Ø	Seitentitel, Hyperlinks	Neue Seite	Ø	Benutzerprofile mit Foto. Zuordnen von Naviga-tionsaktionen zu Anwendern	Ø	Webseiten (URIs), Link-Struktur	Client zeigt andere Benutzer auf selber Webseite und bietet Online-Chat
CoIn	CoInternet-Projekt (Wollenweber 2002; Wollenweber 2004)	Work-group	Einbetten von zusätzlichen Tooltips, erweiterte Google-Suchausgabe	Daten zu allen referenzierten Objekten einer Seite, Google-Ergebnislisten	Metadaten zu Webseiten, Link-Struktur, Zugriffsdaten, HTTP-Attribute	Neue Seite	Seite laden, Browser-Fenster fokussieren	Zuordnen von Aktionen und Annotationen zu Benutzern	Besuchte Seiten	*s. HyperScout* Benutzerkonten, Bewertungen und Annotationen zu Webseiten	Fenster neben dem Browser zur Annotation von Webseiten und für Suchanfragen
Cop	Copernic Agent (Copernic Technologies Inc. 1996-2012) www.copernic.com	Client	Ø	Ausgaben diverser Such-maschinen	Interpretation von Suchergebnissen	Ø	Seite laden	Ø	Suchanfragen, Ergebnislisten, Gewählte Treffer	Titel, Beschreibung, Rang in Ergebnisliste Status einer Seite	Eigener Client der Ergebnisse von Suchmaschinen zusammenfasst
CrLi	CritLink Mediator (Yee 2002)	Work-group/ Server	Toolbar in Webseiten, Link-Icons für Annotationen	Ø	Seitentitel, Analyse des Textkörpers	Ø	Ø	Autor der Annotation	Anmerkungen, Bewertungen und Hyperlinks der Benutzer	Metadaten zu Webseiten, Benutzerkonten, Link-Anker, Anmerkungen etc.	Formular zur Eingabe von Annotationen im Webbrowser
DLS	DLS: Distributed Link Service / Microcosm, (s. Anhang B.13; Carr, DeRoure et al. 1995)	Client & Work-group	Einbetten zusätzlicher Links	Ø	Analyse des Textkörpers, Lokalisierung von Link-Ankern	Ø	Ø	Unterscheiden von Link-Autoren	Von Benutzern für das Web erstellte Links	Zusätzliche Link-Anker und bidirektionale, typisiert Links	Erstellen von Links, Auswählen von Link-Datenbanken
Foot	Footprints (Wexelblat 1999)	Server	Darstellung der Link-Popularität (als Text) und Pfade der Nutzer (Applet)	Alle Webseiten eines Servers	Seitentitel, Hyperlinks	Ø	Seite laden	Zuordnen von Benutzungspfaden	Besuchte Seiten aller Benutzer der Site	Titel, Hyperlinks, Navigationspfade	Treeview mit hierarchischer Ansicht populärer Pfade
GrLa	GroupLab Unified History System (Kaasten & Greenberg 2001)	Client	Ø	Ø	Seitentitel, Thumbnails	Neue Seite	Seite laden	Ø	Besuchte Seiten, Bookmark-Aktionen	Metadaten zu Webseiten, Domains, Thumbnails, Benutzeraktionen	Browser-Sidebar: Durchsuchbare grafische History, Bookmark-Funktion
HiTo	History Tool (Bumann & Bilinski 1998)	Client	Ø	Ø	Seitentitel	Neue Seite, Verlassen einer Seite	Seite laden	Ø	Besuchte Seiten, Verweildauer, Bookmark-Aktionen	Webseiten (URIs), hierarchische Struktur, Bookmarks, Benutzeraktionen	Fenster neben dem Browser mit History-Liste und Bookmark-Funktion
KSS	Kollaborative Filter System KSS (Paepcke, Garcia-Molina et al. 1998; Rodríguez-Mulà, García-Molina et al. 1998)	Work-group	Einbetten v. Nutzungs-angaben hinter Links, kombinieren der Ergebnisse mehrerer Suchmaschinen.	Trefferlisten von Suchmaschinen	Volltext-Analyse, Link-Anker	Ø	Ø	Suchanfragen der Benutzer	Besuchte Seiten, Ausgewählte Links, Suchbegriffe für Suchmaschinen	Webseiten (URIs), Volltext der Seite, Zugriffe auf Links, Suchbegriffe	Formular zum Stellen einer Suchanfrage an mehrere Benutzer
Lens	Link Lens (Stanyer & Procter 1999)	Client	Ø	Referenzierte Objekte der aktuell dargestellten Seite	Seitentitel, MIME-Types, Größe, Zugriffe, Medien, Volltext-Analsyse etc.	Mausposition im Dokument	Seite laden	Ø	Ø	Metadaten der Webseiten und eingebetteter Objekte, Links, Thumbnails etc.	Dialogboxen mit Tabs werden zu Links eingeblendet
Narc	Narcissus (Abb. 17 rechts; Hendley, Drew et al. 1995)	Client & Server	Ø	Alle Webseiten eines Servers	Link-Strukturen	Ø?	Ø?	Ø	Ø	Webseiten (URIs) Link-Struktur	Eigener Client zeigt 3D-Graph der Struktur einer Site
NVB	Navigational View Builder (Abb. 17 lints; Mukherjea & Foley 1995)	Client & Server	Ø	Alle Objekte eines Servers	Seitentitel, MIME-Types, Struktur-Infos etc.	Ø	Ø	Ø	Ø	Metadaten zu Webseiten und eingebetteten Medien	2D-Karte aller Objekte. Filter- und Cluster-Funktionen
Path	WebPath (Frécon & Smith 1998)	Client	Ø	Ø	Seitentitel, Farben, Hinter-grundgrafik, Link-Struktur	Ø	Seite laden	Ø	Besuchte Seiten	Titel, Hintergrundgrafik und Farbschema	3D-History-Browser neben Webbrowser
Robo	RoboForm (Siber Systems Inc. 1999-2012) www.roboform.com	Client	Passwortabfragen in Webseiten ermitteln	Ø	Formularfelder	Neue Seite	Formularfelder ausfüllen	Synchronisation der Daten über zentralen Server	Eingegebene Formulardaten und Passwörter	URI, Formularfelder, Formulardaten	Browser Toolbar, Fenster zur Konfiguration
SGA	SessionGraphs (Mayer & Bederson 2001)	Client	Ø	Ø	Seitentitel, Link-Struktur, Thumbnails	Neue Seite, Ver-lassen einer Seite	Seite laden	Ø	Ø	Titel, Hyperlinks, Thumbnails, Navigationspfade	2D-Darstellung von Web-Sitzungen neben dem Browser
TOC	WebTOC (Nation 1998)	Client	Einbetten eines Applet, das eine hierarchische Ansicht zeigt	Erfassen aller Ressourcen des lokalen Webservers	MIME-Type, eingebettete Objekte, Dateigröße	Neue Seite	Seite laden	Ø	Ø	Alle Ressourcen eines Servers mit Struktur, Medientyp und Struktur	Hierarchische Ansicht wird als Applet in Seite eingebunden
VIS	VISWIP (Cugini & Scholtz 1999)	Server	Ø	Alle Webseiten eines Servers	Seitentitel, Link-Struktur	Ø	Ø	Zuordnen von Benutzungspfaden	Besuchte Seiten aller Benutzer der Site	Metadaten zu Webseiten, Link-Struktur, Navigationspfade	Ø (Serverseitiges GUI)
Vise	Webvise (Grønbæk, Sloth et al. 1999)	Client	Einbetten zusätzlicher Links, Annotationen und Guided-Tour-Funktionen	Ø	Analyse des Textkörpers, Lokalisierung von Link-Ankern und nmerkungen	Markieren von Text, rechter Mausklick	Seite laden	Ø	Von Benutzern für das Web erstellte Links	Bidirektionale, typisierte Hyperlinks, Link-Anker, Annotationen, Guided Tours	Zusätzliches Menü für Link-Funktionen, Erweiterung des Pop-up-Menüs der rechten Maustaste
WPH	WBI Personal History (Barrett, Maglio et al. 1997)	Client	Einbetten von Elementen zum Zugriff auf History	Ø	Seitentitel, Volltext	Ø	Seite laden	Ø	Besuchte Seiten	Textliche Speicherung der Inhalte von Webseiten	Zusätzliche Funktion: Durchsuch-bare History
HypS	HyperScout (zum Vergleich)	Client	Einbetten von Tooltips, Maus-Icons und Link-Overlays	Metadaten über alle von der aktuellen Seite referen-zierten Objekte	Metadaten zu Webseiten, Link-Strukturen, Zugriffe, HTTP-Attribute etc.	Neue Seite	Ø	Ø	Besuchte Seiten Besuchte Server	Metadaten zu Webseiten & anderen Web-Objekten, Link-Struktur, Benutzeraktionen	Erweiterung des „Link-Interfaces“ von Webseiten, GUI zur System-Konfiguration
		Server	Einfügen zusätzlicher Anker-Attribute	Alle lokalen und alle direkt referenzierten externen Ressourcen	s.o.	Ø	Ø	Optional für Social-Navi-gation Funktionen	Besuchte Seiten anderer Benutzer	s.o. Benutzerkonten	Ø

[228] Die Analyse wurde vor der Konzeption des Frameworks durchgeführt, daher sind die meisten aufgeführten Systeme relativ alt. Eine stichprobenartige Untersuchung neuerer Systeme zeigte, dass diese Klassifikation auch bei ihnen angewendet werden kann (z.B. Millen, Feinberg et al. 2006; Tabard, Wackay et al. 2007; Yesilada, Lunn et al. 2007; Morris, Morris et al. 2008; Won, Jin et al. 2009; Liu & Tajima 2010; Teevan, Dumais et al. 2010).

[229] Die Spalte gibt an, ob es sich um eine Erweiterung des *Clients*, des *Servers* oder um ein System für „*Workgroups*“ handelt, bei denen es ein eingegrenzter Benutzerkreis gemeinsam nutzt.

8.2.2 Autonomer Zugriff auf Ressourcen im Internet

Einige Konzepte zur Vereinfachung der Benutzbarkeit des Webs benötigen *zusätzliche Daten* aus dem Internet, um die dargestellten Dokumente mit zusätzlichen Informationen anzureichern (s. o.) oder die Daten in neuer Form darzustellen (Tabelle 4, Spalte 2, Projekte *BaLi, CoIn, Cop, KSS, Narc, NVB,VIS*).

Zu diesem Zweck sollte das Framework über eine Komponente zum *Zugriff auf Ressourcen im Internet* verfügen. Da die spezifischen Anforderungen der Projekte an diese Komponente sehr unterschiedlich sind, muss sie flexibel programmierbar und steuerbar sein. Sie sollte zudem in der Lage sein, die an sie gestellten Aufgaben *selbstständig zu bearbeiten* und *auf Benutzeraktionen zu reagieren*.

Für den Zugriff auf das Web kann diese Anforderung mittels eines programmierbaren, regelbasierten persönlichen Web-Crawlers (siehe Miller & Bharat 1998) erfüllt werden, der die benötigten Daten gemäß seiner Aufträge zusammenstellt und dabei beispielsweise – entsprechend der Anforderungen und des Systemzustands – zwischen möglichst ressourcenschonend und die maximale Bandbreite nutzend vorgeht (realisiert im *Scone Robot*, siehe Abschnitt 8.6.2).

8.2.3 Interpretation und Analyse von Web-Ressourcen

Alle untersuchten Projekte benötigen die Daten in einer interpretierten Form: Oft werden von HTML-Dokumenten die Elemente und Attribute bearbeitet, oder es werden Meta-Informationen wie Seitentitel, Inhaltszusammenfassung, Zugreifbarkeit oder Verzögerung bei der Übertragung ausgewertet (Tabelle 4, Spalte 2, alle Projekte).

Jedes Dateiformat erfordert einen spezifischen Parser zur Interpretation der Daten. Da für das Web (X)HTML das wichtigste Format darstellt, soll ein entsprechende Parser bereits Teil des Frameworks sein (umgesetzt im *Scone HTMLParser* und im *DocumentParser*, siehe Abschnitte 8.6.5.1 und 8.6.5.3). Interpreter für weitere Formate müssen sich einfach integrieren lassen (siehe Abschnitt 8.6.1.1, die *Scone ObjectStreams*).

Eine weitere häufig genutzte Repräsentationsform von Webseiten sind *Thumbnails* (s. Projekte *GrLa, Lens, SGA*), also die verkleinerte grafische Darstellung der Seite (vergl. Abschnitt 4.5.2.4). Das Framework sollte sie daher ebenfalls bereitstellen können (realisiert im *Scone ThumbnailGenerator*, Abschnitt 8.6.5.5).

8.2.4 Ermittlung von Status und Ereignissen im Browser

Erweiterungen für Webbrowser bieten dem Anwender häufig kontextabhängige Informationen an. Dafür müssen sie über den Zustand des Browsers, das gegenwärtig dargestellte Dokument und die Benutzeraktionen informiert werden (Tabelle 4, Spalte 4, Projekte: *BaLi, CoB, CoIn, GrLa, HiTo, KSS, Lens, Robo, SGA, TOC, Vise*).

Dies setzt für das Framework eine Schnittstelle voraus, die Nachrichten über Browser-Ereignisse an die Erweiterungen senden kann und über die sich der Browser-Zustand abfragen lässt. Da alle Browser proprietäre APIs und Integrationsmöglichkeiten bieten (s. Abschnitt 8.3.1), ist ein entsprechender „Wrapper" wünschenswert, der die Schnittstellen der Browser unifiziert und so die Entwicklung von browserunabhängigen Systemen vereinfacht (realisiert im *Scone AccessTracking*, s. Abschnitt 8.6.4).

8.2.5 Steuerung des Browsers

Browser-Erweiterungen müssen häufig nicht nur auf Benutzeraktionen reagieren können, sondern auch in der Lage sein, den Browser zu steuern. Dies ist beispielsweise bei Navigationswerkzeugen gefordert, die dem Anwender eine Übersicht von Web-Ressourcen bieten, von denen eine bei Auswahl im Browser dargestellt wird (Tabelle 4, Spalte 5, Projekte: *BaLi, CoIn, Cop, Foot, GrLa, HiTo, Lens, Path, Robo, SGA, TOC, Visa, WPH*).

Die proprietären Schnittstellen der Browser machen hierfür ebenfalls eine einheitliche, kapselnde Schnittstelle im Framework wünschenswert. Neben dem Laden neuer Seiten im Browser sollte sie auch das Öffnen, Schließen und Anpassen von Browser-Fenstern gestatten (implementiert in den Hilfskomponenten *BrowserControl* und *Scone RAS*, s. Abschnitte 8.6.5.1 und 8.6.5.4).

8.2.6 Mehrbenutzerunterstützung

Nutzen mehrere Personen ein Softwaresystem gemeinsam, wird eine *Mehrbenutzerunterstützung* benötigt, damit sich Daten und Aktionen einzelnen Anwendern zuordnen lassen und Funktionen und Ausgaben personalisiert werden können (Tabelle 4, Spalte 6, Projekte: *CoB, CoIn, CrLi, DLS, Foot, KSS, VIS*). Eine Komponente zur Unterscheidung von Anwendern, zur Verwaltung von Benutzerkonten und zur Zuordnung von Ressourcen und Ereignissen ist daher auch notwendig, wenn eine Web-Erweiterung für Arbeitsgruppen eingesetzt werden soll (umgesetzt im *Scone UserHandling*, Abschnitt 8.6.4.3).

8.2.7 Aufzeichnen von Benutzeraktionen

Systeme zur Navigationsunterstützung im Web reagieren häufig nicht nur auf aktuelle Aktionen der Anwender, sondern werten auch ihre früheren Handlungen aus. Dies gilt für *History-Tools*, die sowohl das Zurückkehren zu Seiten vereinfachen sollen (Tabelle 4, Spalte 7, Projekte: *BaLi, GrLa, HiTo, SGA, WPH*) als auch für Orientierungshilfen, die dem Benutzer aufgabenabhängige Informationen bereitstellen (s. Projekte: *BaLi, CoIn, CrLi, HypS, KSS, Path, SGA*).

Das Framework sollte daher in der Lage sein, Benutzeraktionen aufzuzeichnen und den Zugriff auf diese Nutzungsdaten zu gewährleisten (realisiert im *Scone AccessTracking* und den *NetObjects*, siehe Abschnitte 8.6.4 und 8.6.1). Serverseitige Social-Navigation-Systeme (s. Abschnitt 3.1.6) benötigen zudem einen Zugriff auf die Nutzungsprotokolle von Webservern (s. Projekte: *CoIn, Foot, HypS, KSS*, realisiert im *LogParser*, s. Abschnitt 8.6.5).

8.2.8 Repräsentation und Persistierung von Ressourcen

Damit eine Navigationshilfe die vom Benutzer zugegriffenen Ressourcen und Links analysieren und bearbeiten kann, müssen sie als Datenobjekte im System *repräsentiert* werden. Sollen frühere Aktionen zugänglich sein, so sind diese Objekte zudem zu *persistieren*, sodass sie beispielsweise später durchsuchbar sind (Tabelle 4, Spalte 8, Projekte: *GrLa, HiTo, SGA, WPH*) als Übersicht visualisiert werden können (Projekte: *BaLi, Cop, Foot, HiTo, Narc, NVB, Path, SGA, VIS*) und sich ihre Verknüpfungsstrukturen auswerten lassen (Projekte: *BaLi, CoB, Foot, HypS, KSS, Narc, NVB, Path, VIS*).

Im Web sind zwar beliebige Dateitypen zu finden, dennoch weisen alle Web-Ressourcen einige gemeinsame Eigenschaften auf: Sie werden immer mittels einem *URI* adressiert, ihnen ist ein *MIME-Type* zugeordnet, und für die Übertragung wird HTTP mit den entsprechenden Statusparametern verwendet. Folglich sollte das Framework eine Komponente und ein Standard-Datenschema zur Repräsentation (und Persistierung) von Ressourcen bereitstellen, das grundlegende Daten für beliebige Web-Ressourcen verarbeiten kann und zudem den wichtigsten Dateityp (X)HTML explizit unterstützt. Das Schema muss dennoch erweiterbar sein und sich einfach an die spezifischen Erfordernisse des jeweiligen Projekts anpassen lassen (umgesetzt in den *Scone NetObjects*, s. Abschnitt 8.6.1).

8.2.9 Anpassung der Benutzungsschnittstelle des Browsers und der dargestellten Objekte

Die meisten Konzepte zur Unterstützung der Navigation und Orientierung im Web integrieren sich entweder in die Benutzungsschnittstelle des Browsers (Tabelle 4, Spalte 9, Projekte: *DLS, GrLa, HypS, Lens, TOC, Vise*) oder sie bieten ergänzende Schnittstellenelemente neben dem Browser an (Projekte: *BaLi, CoIn, Foot, HiTo, Path, SGA*).

Eine Erweiterung der Schnittstellenelemente des Browsers selbst wird dadurch erschwert, dass jeder Browser eine proprietäre API bietet, deren Programmierung vom verwendeten Betriebssystem und von der Browser-Version abhängt (s. Abschnitt 8.3.1).

Diese Problematik lässt sich auch mit einem Framework zur Erweiterung des Webs nicht grundsätzlich lösen, dennoch sollten sich auf seiner Grundlage möglichst viele Konzepte zur Erweiterung der Benutzungsschnittstelle des Webbrowsers zumindest *prototypisch* und *in großer Annäherung* an das Ideal realisieren lassen. Hierfür können beispielsweise zusätzliche Schnittstellenelemente *in den Webseiten* oder *neben dem Browser* dargestellt werden.[230] Die Kopplung mit dem Browser lässt sich dabei über die bereits in den Teilen 8.2.4 und 8.2.5 aufgeführten Komponenten realisieren.

[230] Damit das Framework eine grafische Schnittstelle auf dem Benutzerdesktop darstellen kann, muss es (oder eine Zusatzkomponente) auf dem Computer des Anwenders installiert sein.

8.2.10 Unterstützung der Systemevaluation

Die Unterstützung von Benutzbarkeitstests ist genau genommen keine *technische* Anforderung an das hier konzipierte Framework, sie ergibt sich aber aus seiner Zielsetzung: Da es zur Realisierung neuer Konzepte *und* zu deren Evaluation dienen soll, bietet es sich an, auch Werkzeuge zur Durchführung (partizipativer) Benutzbarkeitstests zur Verfügung zu stellen (s. Abschnitt 8.2).

Vielen Evaluationstechniken mit Benutzern werten neben den Kommentaren der Teilnehmer auch ihre Aktionen aus, um hieraus Rückschlüsse auf Stärken und Schwächen des Systems zu ziehen. Dies gilt nicht nur für partizipative Evaluationsverfahren mit eher kurzer Testdauer (wie bei Protokollanalysen (Dix, Finlay et al. 1993: 386ff; Shneiderman 1997) oder dem Thinking-Aloud-Verfahren (siehe Abschnitt 6.1.2ff; Lewis 1982; Jørgensen 1989), sondern auch für Langzeitstudien (Nielsen 1993b; Hilbert & Redmiles 2000; Weinreich, Obendorf et al. 2006b). Das Framework sollte daher die Benutzeraktionen mit dem Browser aufzeichnen können und in einem zur weiteren Analyse geeigneten Format zugänglich machen.

Benutzbarkeitsstudien lassen sich zudem häufig durch Evaluationssoftware unterstützen: Sie kann den Teilnehmern auf dem Bildschirm die für den Test notwendigen Informationen und Aufgaben präsentieren und ihnen die Möglichkeit bieten, Antworten und Feedback abzugeben (Ivory & Hearst 2001; Obendorf, Weinreich et al. 2004; Jerroudi, Ziegler et al. 2005). Bezogen auf das Web sollte ein entsprechendes Evaluationssystem zudem den Browser steuern können, um eine bestimmte Webseite als Ausgangsdokument für einen Test darzustellen und die Möglichkeit bieten, auf Ereignisse im Browser zu reagieren, sodass beispielsweise das Lösen von Aufgaben automatisch erkannt wird. Diese und weitere Anforderungen werden im Scone Plug-in *TEA UserTestTool* angeboten (s. Abschnitt 8.6.5.7).

8.3 Techniken zur Erweiterung des Webs

Zur Eingrenzung der möglichen Vorgehensweisen bei der Implementation eines neuen Frameworks zur Entwicklung von Web-Orientierungs- und Navigationshilfen wurden die verfügbaren Techniken zur Erweiterung von Webbrowsern und Webservern analysiert (Abschnitt 8.3.1) und bezüglich der Anforderungen an das Framework untersucht (Abschnitt 8.3.2).

8.3.1 Technische Möglichkeiten zur Erweiterung von Webbrowsern

Es stehen zahlreiche technische Möglichkeiten und Schnittstellen zur Erweiterung von Webbrowsern zur Verfügung, die jeweils spezifische Vor- und Nachteile bieten.

Beliebige Änderungen der Browser-Funktionalität sind möglich, wenn man Zugriff auf den **Quellcode** des Systems hat. In der Regel ist dies nur bei Open-Source-Browsern wie *Mozilla Firefox* oder *Google Chrome* gegeben – kommerzielle Browser wie der Microsoft Internet Explorer bleiben für ein solches Vorgehen ausgeschlossen. Eine Erweiterung des Codes von

grafischen Webbrowsern ist allerdings eine nicht zu unterschätzende Herausforderung, da sie zu den komplexesten Open-Source-Projekten gehören. Beispielsweise bestand bereits 2002 die Browser-Suite „Mozilla“ aus über zwei Millionen Zeilen Quellcode (Wheeler 2002). Die Anpassung solcher Applikationen erfordert daher einen erheblichen Einarbeitungsaufwand. Darüber hinaus ist man bei der Erweiterung des Quellcodes an eine bestimmte Browser-Version gebunden;[231] bei dem heutigen Entwicklungstempo veraltet der modifizierte Browser somit schnell.

Um dennoch Webbrowser erweitern zu können, verfügen sie über entsprechende *Programmschnittstellen (APIs)*. Leider sind die APIs populärer Browser recht unterschiedlich, und selbst die „standardisierten“ Schnittstellen sind bei einzelnen Browsern bzw. Browser-Versionen häufig im Detail inkompatibel zueinander.

Die erste Technik zur Erweiterung von Webbrowsern war das **Browser-Plug-in**. Diese Programmschnittstelle wurde ursprünglich für den *Netscape Navigator* entwickelt und von den meisten anderen Browser-Herstellern übernommen.[232] Plug-ins sind einfach zu installieren und bis heute gebräuchlich, z. B. als *Adobe Flash Player* und als *Apple Quicktime Viewer*. Ihr Anwendungsbereich und ihre Möglichkeiten sind aber begrenzt: Ein Plug-in ist in der Regel bestimmten MIME-Types zugeordnet, und die Ausgabe erfolgt in definierten Bereichen *innerhalb* eines HTML-Dokuments. Darüber hinaus sind die Interaktionsmöglichkeiten mit dem Browser auf wesentliche Operationen beschränkt, beispielsweise lässt sich zwar ein neues Dokument laden, aber nicht das bereits dargestellte Dokument manipulieren. Plug-ins müssen an die jeweilige Betriebssystemplattform und den Browser angepasst werden, zumal sich nicht alle Browser-Hersteller an den Plug-in-Standard von Netscape gehalten haben.[233]

Eine Ergänzung zum Plug-in-Konzept bietet der Microsoft Internet Explorer mit den **Browser Helper Objects (BHOs)**.[234] Ein BHO wird innerhalb des Browsers registriert und bei jedem Browserstart mitgeladen. Es ist daher durchgängig verfügbar. Mit einem BHO kann man die Bedienelemente des Browsers erweitern, z. B. um eine neue *Toolbar* zum Browser hinzuzufügen. Sie können auf vielfältige Ereignisse im Browser reagieren, wie Benutzeraktionen mit der Maus oder das Laden eines neuen Dokuments. Darüber hinaus lässt sich der Browser umfassend steuern, und die dargestellten Webseiten sind über das DOM (*Document Object Model*) zugreifbar und manipulierbar (Hégaret, Whitmer et al. 2005). Populäre Beispiele sind die *Google Toolbar* (http://toolbar.google.com/) und das BHO für den *Acrobat Reader*, mit dessen Hilfe PDF-Dokumente direkt im Browser-Fenster dargestellt

[231] Sofern die Modifikationen nicht in die offizielle Version einfließen, wären bei Browser-Updates die Anpassungen jeweils zu übertragen.

[232] Mehr technische Informationen zu Plug-ins findet man unter https://developer.mozilla.org/En/Plugins.

[233] Insbesondere die Plug-in-API des Internet Explorers weist einige entscheidende Abweichungen auf (Roberts 1999).

[234] BHOs werden im Alltagsgebrauch häufig auch als Plug-ins bezeichnet, da der Installationsprozess vergleichbar ist. Es handelt sich aber um eine unabhängige, proprietäre API von Microsoft (Roberts 1999).

werden. Die technischen Möglichkeiten von BHOs sind aber ebenfalls beschränkt: Unter anderem kann nicht ermittelt werden, welche Browser-Knöpfe oder Pulldown-Menüs der Benutzer ausgewählt hat (Haß 2004).

Eine weitere Schnittstelle zur Erweiterung des Internet Explorers ist **ActiveX**.[235] Dies bezeichnet eine auf dem **COM (Component Object Model)** aufbauende, recht lose Architektur von Microsoft, die den Zugriff auf viele Funktionen in Microsoft-Programmen und Bibliotheken gestattet. Die ActiveX-Schnittstelle des Internet Explorers gewährt Programmierern den Zugang zu allen wichtigen Funktionen des Browsers. Eine Integration solcher Erweiterungen in die Benutzungsoberfläche des Browsers (wie bei BHOs) ist zwar nicht vorgesehen, über das DOM kann aber auf die Dokumente im Browser zugegriffen werden, und Event-Listener gestatten es, auf nahezu beliebige Ereignisse im Browser zu reagieren. Beispiele für Systeme, die diese Schnittstelle nutzen, sind *WebVise*[236] und *Arakne (*Bouvin 2000).

Eine *plattformunabhängige* Technik zur Entwicklung von Erweiterungen für das Web bieten **Java Applets**. Dies sind (zumeist kleine) Programme in der objektorientierten Sprache Java, die in Webseiten eingebunden werden. Die Möglichkeiten von Applets sind aufgrund des *„Sandbox"* genannten Sicherheitskonzepts eingegrenzt: Einem Applet ist weder der Zugriff auf das Dateisystem oder andere kritische Systemressourcen möglich, und Netzwerkverbindungen lassen sich lediglich zu dem Server aufbauen, von dem das Applet geladen wurde (Middendorf, Singer et al. 2002). Mithilfe von Applets können beliebige interaktive Benutzungsschnittstellen in Webseiten integriert werden und es lassen sich Fenster mit grafischen Ein- und Ausgabeelementen öffnen. Die Kommunikationsmöglichkeiten mit dem Browser sind aber beschränkt, da der Zugriff nur über die Schnittstelle *„LiveConnect"* und die im Browser integrierte Skriptsprache *JavaScript* möglich ist. Zwei Beispiele für die Nutzung von Java Applets zur Vereinfachung der Navigation im Hypertext sind *WebTOC,* das eine interaktive hierarchische Übersicht für Websites anbietet (Nation 1998) und *Displets*, die multiple Links in Webseiten einbinden (Vitali, Chiu et al. 1997).

JavaScript ist im Gegensatz zu Java eine Skriptsprache und wurde ursprünglich von *Netscape Communications* entwickelt und später als **ECMAScript** standardisiert (ECMA 1999). JavaScript-Programme werden zumeist innerhalb von HTML-Seiten eingesetzt und eignen sich für vielfältige Erweiterungen der Schnittstelle des Webs. Die Sprache wurde lange Zeit aufgrund der schlechten Performanz, Stabilität und Standardkonformität[237] der Interpreter in

---

[235] Die Schnittstelle *ActiveX* ist nicht mit *ActiveX-Control* zu verwechseln, das trotz des ähnlichen Namens für eine andere Technik steht. ActiveX-Controls sind kleine Applikationen, die aus dem Web geladen und lokal ausgeführt werden. Sie wurden von Microsoft als Konkurrenz zu *Java Applets* entwickelt und können über die COM-Schnittstelle auf viele Funktionen des Browsers zugreifen. Aus Sicherheitsgründen sind sie in Verruf geraten (Chappell 1996).

[236] *WebVise* ist eine Erweiterung des Devise-Hypertext-Systems (s. Anhang C.1.5) für das World Wide Web (Grønbæk, Sloth et al. 1999).

[237] In der Praxis interpretiert jeder Browser JavaScript im Detail anders. Dies erschwert die Entwicklung plattformunabhängiger Erweiterungen (s. Abschnitt 4.4.6).

den Browsern nur für kleinere, interaktive Erweiterungen und Spiele eingesetzt worden. Erst seit etwa Ende 2005 erfährt es unter dem Namen **Ajax** auch Einsatz für umfangreichere Web-Anwendungen (s. Abschnitte 2.2.5 und 9.2). Ajax steht für „Asynchronous JavaScript and XML" und wird zumeist im Zusammenhang mit *browser-unabhängigen JavaScript-Bibliotheken* für *interaktive Webseiten* verwendet. Diese Bibliotheken vereinheitlichen die Programmierung der Browser und ermöglichen es, auch *nach* dem Laden einer Seite neue Elemente und Daten vom Webserver abzurufen und in die Seite einzubinden (Crane, Pascarello et al. 2005; Garrett 2005; O'Reilly 2005). JavaScript hat über das DOM vollen Zugriff auf das aktuelle Dokument und erlaubt eine weitgehende Steuerung des Browsers; so lassen sich Browser-Fenster öffnen, verschieben, skalieren oder schließen (Münz & Gull 2010). Da JavaScript von allen aktuellen grafischen Browsern unterstützt wird,[238] wird es inzwischen auf einem erheblichen Teil der Websites zur Erweiterung der Benutzungsschnittstelle eingesetzt.

Beim Browser *Firefox* spielt JavaScript eine weitere bedeutende Rolle: Alle wichtigen Elemente der Benutzungsschnittstelle sind in der Sprache **XUL (XML User Interface Language)** geschrieben,[239] einer auf XML und JavaScript basierenden Beschreibungssprache für grafische Schnittstellen. Dies macht Änderungen an der Benutzungsschnittstelle des Browsers vergleichsweise einfach und erlaubt es, nahezu beliebige plattformunabhängige Erweiterungen zu erstellen. Es gibt inzwischen tausende solcher **Browser-Extensions** (auch *Browser-Add-Ons* genannt)[240] für Firefox in der Sprache XUL. Diese Technik weist lediglich zwei Nachteile auf: Erstens sind die Extensions nicht für andere Browser nutzbar und zweitens müssen sie regelmäßig an die aktuelle Browser-Version angepasst werden, da die entsprechende XUL-API von Firefox ebenfalls fortlaufend weiterentwickelt wird (vergl. Abschnitt 8.9).

Eine plattform- und browserunabhängige Technik zur Erweiterung des Webs bieten **Intermediaries** (Barrett & Maglio 1998). Ein Intermediary bezeichnet ein Softwaresystem, das sich zwischen Client und Server befindet und so auf den gesamten Datenstrom zugreift (s. Abschnitt 8.4.1). Ein **clientseitiger Web-Intermediary** ist für den Browser als *Proxy* eingerichtet, sodass der Zugriff des Benutzers auf sämtliche Websites über diese Komponente läuft. So kann der Intermediary die Anfragen eines oder mehrerer Clients sowie die Antworten aller kontaktierten Server beliebig filtern, ändern und aufzeichnen. Ein **serverseitiger Web-Intermediary** (auch „Reverse Proxy" genannt) wird hingegen unter dem Domain-Name eines Webservers eingerichtet und er leitet die Anfragen sämtlicher Clients an den ursprünglichen Webserver weiter (siehe Abschnitt 8.3.2). Mit Intermediaries lassen sich neue Elemente und Links in Webseiten einfügen oder auch ganz neue Dokumente generieren. Allerdings sind die grafischen Steuerungselemente des Browsers selbst (wie Pulldown-

[238] Microsoft nennt die entsprechende Sprache JScript. Der Internet Explorer bietet darüber hinaus unter dem Titel *„Active Scripting"* ebenfalls die Möglichkeit, VisualBasic (bzw. *VBScript*) als Skriptsprache einzusetzen.

[239] Mehr Informationen zu XUL findet man unter: https://developer.mozilla.org/En/XUL.

[240] Eine Übersicht der Firefox-Erweiterungen bietet: https://addons.mozilla.org/de/firefox/.

Menüs oder Toolbars) nicht erweiterbar, da ein Intermediary nicht *in* den Browser integriert ist. Beispiele für die Ergänzung der Benutzungsschnittstelle des Webs mithilfe von Intermediaries sind die Projekte *Traffic Lights* (s. Abschnitt 5.3.2 und Abb. 51; Campbell & Maglio 1999) und *Microcosm/DLS* (s. Anhang B.13; Carr, DeRoure et al. 1995).

8.3.2 Technische Möglichkeiten zur Erweiterung von Webservern

Die für das technische HyperScout-Konzept entwickelten Protokoll- und Spracherweiterungen (s. Kapitel 7.5) erfordern ebenfalls eine Anpassung der Funktionalität des Webservers. Hierfür stehen mehrere Techniken zur Verfügung, die sich durch die Art der Integration und die verwendete Schnittstelle kennzeichnen.

Für eine Erweiterung der Funktionalität des populären Webservers *Apache* (Netcraft 2011) steht die **Apache Module API** zur Verfügung. Über diese Schnittstelle kann man an vielen Punkten innerhalb der Verarbeitung einer Anfrage eingreifen und so praktisch beliebige Erweiterungen des Servers realisieren. Objekte lassen sich filtern, ändern oder dynamisch generieren (Stein & MacEachern 1999). Es ist ebenfalls möglich, Anfragen an andere Systeme weiterzuleiten, um z. B. einen Lastausgleich in einem Serverpark zu koordinieren. Populäre Apache-Module sind *OpenSSL* für die verschlüsselte Datenübertragung per HTTPS und die serverseitige Skriptsprache *PHP*.

Der ähnlich weitverbreitete Webserver von Microsoft namens „IIS“ (*Internet Information Services*) bietet eigene Schnittstellen mit vergleichbarer Funktionalität. Die wichtigste API zur Erweiterung des IIS war bis ca. 2003 **ISAPI (Internet Server Application Programming Interface)**. Hiermit sind beliebige Programme und Filter in den Server integrierbar, allerdings steht die Schnittstelle unter dem Ruf, recht kompliziert und fehlerträchtig zu sein (Weissinger 2000). Die „**.**NET“-Softwareplattform brachte komfortablere Erweiterungsmöglichkeiten für den IIS: **HTTPExtensions** werden einem bestimmten MIME-Type zugeordnet, wogegen **HTTPModules** als beliebige Filter sowohl für Anfragen als auch Antworten des Servers agieren können (Lee & Jepson 2005).

Serverseitige Intermediaries sind ein alternatives Konzept, um von der Plattform und Server-Software einer Website unabhängig zu bleiben und sogar Systeme zu erweitern, für die man keine administrativen Rechte hat. Im Gegensatz zu clientseitigen Intermediaries (s. Ende von Abschnitt 8.3.1) sind serverseitige Intermediaries *einem* Webserver zugeordnet. Der Intermediary nimmt sämtliche für den Server bestimmten Anfragen entgegen, verarbeitet sie und leitet sie an ihn weiter. Die Antwort des Servers wird ebenfalls vom Intermediary entgegengenommen, entsprechend bearbeitet und an den Web-Client übermittelt (Barrett & Maglio 1998). Auf diese Weise lassen sich Erweiterungen von Websites und neue Dienste unabhängig von der Architektur eines Servers entwickeln (Malandrino & Scarano 2006).

8.3.3 Diskussion der verfügbaren Erweiterungsmöglichkeiten

Der vorhergehende Überblick hat gezeigt, dass zahlreiche Konzepte und Schnittstellen zur Erweiterung von Webbrowsern und -servern zur Verfügung stehen. Allerdings sind die meisten Lösungen an ein bestimmtes System und eine Plattform gebunden, und teilweise sind die Schnittstellen sogar von der jeweiligen Software-Version abhängig.

Das Defizit proprietärer Schnittstellen hat sich bereits beim *Distributed Link Service* (s. Anhang B.13) gezeigt: Um das Hypertext-System für Windows-, Macintosh- und Unix-Systeme zur Verfügung zu stellen, waren drei Programmversionen aufwendig zu entwickeln (Carr, DeRoure et al. 1998). Einen weiteren Nachteil haben (Cockburn & Jones 1997) beschrieben: Ihre Analyse zahlreicher Navigationshilfen für Webbrowser führte zu dem Ergebnis, dass die *Browser**un**abhängigkeit* eines Werkzeuges ein entscheidendes Designmerkmal für den *Projekterfolg* ist, da sich Browser stetig weiterentwickeln und auch nützliche Entwicklungen mit dem Browser an Aktualität verlieren, wenn sie nicht mehr unterstützt werden. Folglich konnten viele zukunftsweisende Werkzeuge wie die grafische Web-History „*PadPrints*“ (Hightower, Ring et al. 1998) und die integrierte History- und Bookmark-Verwaltung „*WebView*“ (Cockburn, Greenberg et al. 1999) nach kurzer Zeit nicht mehr genutzt werden, da sie nur für eine bestimmte Browser-Version geeignet waren.

Einige der aufgeführten Erweiterungstechniken bieten auch nicht die erforderlichen Voraussetzungen zur Realisierung der HyperScout-Prototypen. Beispielsweise sind *Plug-ins* in ihren Interaktionsmöglichkeiten mit dem Browser zu eingebeschränkt und sie werden erst *nach* dem Laden eines Dokuments aktiviert.

Das einzige Konzept, das sich sowohl zur Erweiterung der Funktionalität und Schnittstelle von Webbrowsern als auch -servern eignet und zudem von den Applikationen und Plattformen unabhängig ist, sind *Intermediaries*. Allerdings haben auch Intermediaries ihre Grenzen: Zum einen beschränken sich die Erweiterungen (in der Regel) auf die Schnittstelle der dargestellten Dokumente. Zudem sind die Änderungsmöglichkeiten der Dokumente von eher „statischer“ Natur, da die Bearbeitung zum Zeitpunkt der Übertragung stattfindet und der Intermediary auf die Dokumente keinen Zugriff mehr hat, sobald sie im Browser angezeigt werden. Erst wenn man Intermediaries, wie im Abschnitt 8.6.5.1 vorgestellt, mit Ajax oder Java Applets kombiniert, lassen sich auch interaktive Erweiterungen für Webbrowser realisieren.

Insbesondere die Flexibilität, die Plattformunabhängigkeit und die Einsetzbarkeit auf Client- und Serverseite waren ausschlaggebend dafür, dass das Intermediary-Konzept für die Entwicklung des Frameworks Scone und der darauf basierenden HyperScout-Prototypen gewählt wurde.

8.4 Intermediaries zur Erweiterung der Funktionalität des Webs

Dieses Teilkapitel gibt einen Überblick über die Eigenschaften und Möglichkeiten von Intermediaries. Im ersten Abschnitt 8.4.1 werden sie *charakterisiert* und gegenüber *Web-Proxies* abgegrenzt. Danach folgt eine *Klassifikation der Einsatzbereiche* von Intermediaries, die gleichzeitig zeigt, wie allgegenwärtig und unverzichtbar solche Systeme inzwischen für die Funktion des Internets geworden sind (Abschnitt 8.4.2). Der dritte Abschnitt 8.4.3 vergleicht die wichtigsten programmierbaren Intermediary-Systeme aus dem Forschungsbereich und analysiert sie bezüglich ihrer Eignung für die Entwicklung von Navigations- und Orientierungswerkzeugen für das Web. Der Abschnitt bietet damit gleichzeitig einen Überblick über die mit dem in dieser Arbeit entwickelten Framework *Scone* verwandten Projekte.

8.4.1 Charakteristika von Intermediaries

Es besteht ein enger Zusammenhang zwischen dem Konzept des *Web-Proxy* (engl. für *Stellvertreter*) und dem des *Intermediary* (engl. für *Vermittler, Mittelsmann*). Beide Systeme befinden sich im Datenstrom zwischen dem Client und dem Server und beeinflussen, ob und wie eine Anfrage des Clients an den Server weitergeleitet wird. Der Begriff *Proxy* ist älter und weiter verbreitet: Er geht auf die Idee zurück, dass ein System *statt* des Servers antwortet, um ihm Aufgaben abzunehmen. Der wichtigste Vertreter ist der *Caching Proxy*, der für eine Gruppe von Web-Nutzern angeboten wird und Server-Daten zwischenspeichert, um die im Internet übertragene Datenmenge zu reduzieren und die Benutzung des Webs zu beschleunigen (erste Arbeiten hierzu sind unter anderem von (Glassman 1994) und (Luotonen & Altis 1994). Ihre größte Bedeutung hatten *Caching Proxies* in den Anfangszeiten des Webs, da damals die verfügbaren Bandbreiten vergleichsweise beschränkt waren.

Der *Intermediary* ist im Gegensatz zum Proxy explizit eine *aktive Komponente,* dessen Möglichkeiten und Aufgaben weit über das Zwischenspeichern von Daten hinausgehen. Die Idee, das Web durch aktive Komponenten im Datenstrom zwischen Client und Server zu erweitern, kann bis zum Projekt *OreO* zurückgeführt werden. *OreO* wurde 1995 auf der vierten World-Wide-Web-Konferenz als *Toolkit* zur Erstellung von *Stream Transducers* vorgestellt (Brooks, Mazer et al. 1995). Stream Transducers sollten als aktive *Information Spaces* an beliebigen Stellen zwischen Client und Server agieren und so die Möglichkeiten des Webs erweitern. Den Anwendungsschwerpunkt sahen sie in einer Steigerung von Performanz und Zuverlässigkeit des Webs. Die Autoren erkannten aber bereits, dass ihr Konzept noch weiteres Potenzial bot: Als Beispiel stellten Sie *Group History* vor, das alle von einer Arbeitsgruppe zugegriffenen Webseiten speichert und über einen durchsuchbaren Textindex zugänglich macht.[241]

[241] Darüber hinaus wurde bereits ein *Transducer* vorgeschlagen, der die Benutzbarkeit des Webs verbessern sollte, indem alle Verweise mit dem „<BLINK>“ Element hervorgehoben wurden. Offensichtlich wurde dies aber nicht tatsächlich erprobt, da blinkende Texte auf Webseiten als „Benutzbarkeitskatastrophe“ gelten (Nielsen 1996; Spool, Scanlon et al. 1998).

Der heute im wissenschaftlichen Bereich gebräuchliche Begriff *Intermediary* für aktive Komponenten im Datenstrom zwischen Client und Server wurde von Wissenschaftlern des *IBM Almaden Research Center* geprägt. Ihre Definition lautet folgendermaßen:

> „Intermediaries are software programs or agents that meaningfully transform information as it flows from one computer to another. … [They are] located in the information stream between producer and consumer …[to] personalize raw information for individuals, devices and situations" (Maglio & Barrett 2000).

Demnach ist ein Intermediary eine aktive Komponente bzw. ein Software-Agent, der für den Anwender Informationen personalisiert. Dabei agiert er nach bestimmten Regeln und ist in der Lage, die übertragenen Daten zu beobachten, zu verändern sowie neue Objekte zu generieren (Barrett & Maglio 1998).

Ihr Anwendungsbereich beschränkt sich nicht auf das World Wide Web. Intermediaries sind konzeptionell für beliebige Netzwerkprotokolle und Dateiformate einsetzbar (Dikaiakos & Zeinalipour-Yazti 2001). Für die Nutzung mit dem HTTP-Protokoll hat die Forschungsgruppe vom IBM Almaden Forschungszentrum das *Web Intermediary Framework „WBI"* (sprich „Web-bie") entwickelt. Ein Anwendungsschwerpunkt von WBI ist die Umwandlung unterschiedlicher Protokolle und Formate, um neue Zugriffs- und Kommunikationsmöglichkeiten zu schaffen. Dies wird beispielsweise benötigt, wenn ein mobiles Endgerät bestimmte Anforderungen bezüglich des Dateiformates an den Webserver stellt (s. Abschnitt 8.4.2.4: „Transcoding") oder wenn zwei *Business Applications* miteinander kommunizieren müssen, die unterschiedliche XML-Dialekte beherrschen[242] (Ihde, Maglio et al. 2001). Die WBI-Entwickler sehen aber ebenfalls Potenziale, die Benutzbarkeit des Webs mithilfe von Intermediaries zu verbessern: Zwei von ihnen mit WBI realisierte Konzepte sind *WBI Traffic Lights* (s. Abschnitt 5.3.2 und Abb. 51) und *WBI Short Cuts*, das dem Benutzer im Browser-Fenster früher bereits verfolgte Pfade anzeigt (Maglio & Barrett 2000).

8.4.2 Einsatzbereiche von Intermediaries: Eine Klassifikation

Obgleich sicherlich wenige Endanwender eine Idee davon haben, was ein Intermediary ist, wäre das heutige Web ohne diese Systeme kaum denkbar. Ihre Einsatzbereiche sind sehr vielfältig, wobei sie zumeist für die Benutzer unbemerkt ihre Dienste im Internet verrichten. Die im Folgenden vorgestellte Klassifikation nach Einsatzbereichen unterscheidet zwischen den Kategorien *Zwischenspeichern, Filtern, Aggregieren, Transkodieren, Personalisieren* und *Privatisieren* von Ressourcen im Internet (vergl. Brooks, Mazer et al. 1995; Barrett & Maglio 1998; Maglio & Barrett 2000; Malandrino & Scarano 2005; Malandrino & Scarano 2006).

---

[242] Im Bereich des *Data Transcoding* hat sich WBI nach Aussagen der Entwicklercrew zu einem der finanziell erfolgreichsten Forschungsprojekte des IBM Almaden Research Center der letzten 10 Jahre entwickelt, da WBI ein Herzstück des *WebSphere Transcoding Publisher* ist, einer Infrastruktur zur automatischen Umwandlung unterschiedlicher XML- und Dateiformate.

8.4.2.1 Zwischenspeichern von Daten

Gegenüber den im vorhergehenden Abschnitt 8.4.1 erwähnten *Caching-Proxies*, die im Browser konfiguriert werden, um die Übertragung der Daten im Web zu beschleunigen, haben inzwischen weitaus raffiniertere Caching-Konzepte größere Bedeutung erlangt. Hierzu gehören globale *Caching-Infrastrukturen*, die Daten von populären Websites im Internet verteilen und für die Anwender auf transparente Weise im Netz ihrer Provider zwischenspeichern. Ein Anbieter eines solchen *Content Delivery Networks* ist *Akamai*.[243] Die meisten größeren Websites (z. B. *Amazon*, *Microsoft*, *Yahoo!* und *eBay*) nutzen einen solchen Dienst für statische Elemente wie eingebettete Bilder und Datei-Downloads, um die Last zu verteilen (Sivasubramanian, Szymaniak et al. 2004).

Im *YaCy-Projekt* wird ein solches Netz von Caching Intermediaries noch weitgehender genutzt.[244] YaCy ist ein lokal zu installierender Intermediary, der alle von einem Anwender zugegriffenen Seiten speichert und gleichzeitig eine *Suchfunktion* anbietet. Dabei kommunizieren die YaCy-Intermediaries mittels eines Peer-to-Peer-Protokolls untereinander, um bei einer Suchanfrage auch die Indexe anderer Anwender zu berücksichtigen. Ziel ist es, ein globales, von kommerziellen Anbietern wie Google unabhängiges Suchsystem aufzubauen (Christen 2005).

8.4.2.2 Filtern von Daten

Intermediaries können sowohl *Anfragen* von Clients als auch *Antworten* von Servern *filtern*. So lässt sich beispielsweise der Zugang zu bestimmten Ressourcen sperren und verhindern, dass Benutzer auf „unerwünschte" Inhalte zugreifen. Dies wird unter anderem von autokratischen Regimes genutzt, um der Bevölkerung systemkritische Inhalte vorzuenthalten.

Es gibt aber auch in Demokratien Gründe, gezielt unerwünschte Objekte mit Intermediaries zu blockieren. Beispielsweise kann man vermeiden, dass Minderjährige Zugang zu jugendgefährdenden Inhalten erlangen.[245] Häufig werden Intermediaries eingesetzt, um störende Elemente, wie Werbung oder Animationen, aus Webseiten herauszufiltern. Darüber hinaus kann überprüft werden, ob Objekte Schadcode (*Malicious Code*) enthalten oder auf problematische Objekte verwiesen wird.[246] Die automatische Filterung von Inhalten hat aber Grenzen: Zwar wurde bereits vor Jahren vom W3C mit *PICS* (*Platform for Internet Content Selection*) ein Metadaten-Standard zum Klassifizieren von Web-Ressourcen für Filtersysteme geschaffen (Resnick & Miller 1996), er wird aber fast gar nicht genutzt. Andere Verfahren sind zumeist unbefriedigend, da ihre heuristischen Algorithmen häufig versagen oder sie

[243] Mehr Informationen über die Firma „Akamai Technologies Inc." findet man unter http://www.akamai.com/.

[244] Die Homepage des YaCy-Projektes lautet http://www.yacy.net/.

[245] Zwei Systeme dieses Genres sind NetNanny (http://netnanny.com/) und CyberPatrol (http://cyberpatrol.com/).

[246] Zwei populäre Intermediaries zum Schutz der Anwender sind der *McAfee WebWasher* und das Open-Source-Projekt *Privoxy*, siehe: http://www.webwasher.de/classic/ und http://www.privoxy.org/.

sich auf unterschiedliche Art umgehen lassen, wodurch sie eine trügerische Sicherheit aufweisen (Neumann & Weinstein 1999).

8.4.2.3 Aggregieren von Informationen

Intermediaries können nicht nur die übertragenen Informationen einschränken, es ist auch möglich, Informationen zu *aggregieren*, indem Daten aus unterschiedlichen Quellen zusammengestellt werden. Unter anderem wird dies im Bereich der *Web-Services* eingesetzt: Vermittelnde Dienste stellen automatisch aus den Daten mehrerer Anbieter eine Übersicht unterschiedlicher Leistungen und Preise zusammen; ein konkretes Beispiel sind Internet-Reiseagenturen.

Im Bereich der Hypertext-Forschung werden aggregierende Intermediaries eher genutzt, um Dokumente mit zusätzlichen Interaktionselementen oder Navigationshinweisen anzureichern. Beispiele sind kollaborative Systeme wie die *CritLink* (Yee 2002; s. Abschnitt 5.2.3 und Abb. 42 aud Seite 134), das Anmerkungen zu beliebigen Web-Dokumenten hinzufügt, und das *KSS-System*, das den Benutzern die Popularität von Links in Webseiten darstellt (Rodríguez-Mulà, García-Molina et al. 1998).

8.4.2.4 Transkodieren von Daten und Protokollen

Das Transformieren von Daten unterschiedlicher Formate und das Vermitteln zwischen verschiedenen Protokollen ist ein häufiger Anwendungsbereich von Intermediaries im Internet (Maglio & Barrett 2000). Die Einsatzbereiche solcher *transkodierender Intermediaries* sind vielfältig. Sie werden unter anderem benötigt, wenn der Zugriff mit bestimmten Geräten oder unter besonderen Einsatzbedingungen auf vorhandene Dienste ermöglicht oder optimiert werden soll. Beispielsweise erlaubt das System *Kamel* den Benutzern von WAP-Mobiltelefonen den Zugang zu Inhalten des Webs (Müller-Wilken 2000). Dienste wie *Opera Turbo*[247], *Amazon Silk*[248] oder auch das Projekt *RabbIT*[249] reduzieren die Übertragungsdauer von Webseiten erheblich, indem sie Dokumente komprimieren und Medieninhalte neu, mit verminderter Qualität, recodieren und an den Client übertragen. Eine solche Optimierung kann für alle Anwender ohne Breitbandzugang eine wesentliche Zeitersparnis bedeuten.

Intermediaries zum Transkodieren von Daten findet man ebenfalls im Business-to-Business-Bereich, wenn eine Anpassung von Protokoll oder Datenformat für den Nachrichtenaustausch zwischen den Anwendungssystemen von Firmen erforderlich ist. Ein Beispiel sind *Business Adapter*, die zur Kopplung unterschiedlicher Geschäftssysteme eingesetzt werden

---

[247] Der Dienst Opera Turbo wird seit Version 10 für Nutzer der Opera-Browser angeboten. Mehr Informationen findet man unter: http://www.opera.com/browser/turbo/.

[248] Bei Amazon Silk sollen die Seiten komplett auf den Cloud-Servern des eigenen Rechenzentrum gerendert werden, um die so vorbereiteten Seiten komprimiert auf den Tablet PC „Amazon Kindle“ übertragen zu können. Der Dienst soll voraussichtlich ab 2012 verfügbar sein. Weitere Informationen erhält man bei Amazon unter: http://amazonsilk.wordpress.com/.

[249] RabbIT 3 ist ein Open-Source-Projekt in der Sprache Java: http://rabbit-proxy.sourceforge.net/.

(Merz 2002; Medjahed, Benatallah et al. 2003), obgleich der Begriff Intermediary in diesem Zusammenhang kaum Verwendung findet.[250]

8.4.2.5 Personalisieren von Inhalten

Das Anpassen der übertragenen Daten kann auch für die *Personalisierung* von Ressourcen an die Bedürfnisse der Benutzer benötigt werden. Beispiele sind die Übersetzungsdienste für Webseiten von *Google* und *Yahoo!*[251] und das Apache-Modul *mod_accessibility*, das nicht nur die (X)HTML-Dokumente des lokalen Webservers, sondern auch (als Intermediary) die Seiten externer Sites automatisch in unterschiedliche Formate aufbereiten kann, um den barrierefreien Zugriff auf das Web zu gewährleisten.[252]

8.4.2.6 Privatisieren und Anonymisieren

Intermediaries finden ebenfalls zur *Privatisierung* und *Anonymisierung* bei der Benutzung von Diensten im Internet Anwendung. Dabei werden alle Anfragen über einen oder mehrere Intermediaries vermittelt, sodass der Benutzer bzw. seine IP-Adresse nachträglich nicht mehr ermittelbar ist. Systeme wie der *Anonymizer*[253] und *AN.ON*[254] bieten Netze von Intermediaries, die jeweils dynamisch ausgewählt und kaskadiert werden, um die Rückverfolgung einzelner Datenpakete nahezu unmöglich zu machen. Zudem lassen sich individuelle Parameter wie Cookies aus den HTTP-Anfragen entfernen, damit aufeinanderfolgende Aktionen eines Benutzers nicht zuzuordnen sind. Diese Techniken erlauben auch das Umgehen von filternden Intermediaries (siehe 8.4.2.2).

8.4.2.7 Evaluation der Benutzung und Benutzbarkeit von Online-Systemen

Ein weiterer Einsatzbereich von Intermediaries sind Studien des Benutzerverhaltens und der Benutzbarkeit von Online-Systemen. Dabei können mithilfe des Intermediary nicht nur die zugegriffenen Objekte protokolliert werden, es lassen sich durch das Einbetten von JavaScript-Code in die Webseiten auch genauere Informationen über die Benutzeraktionen

---

[250] Anwendungsunabhängige Systeme, die die Kommunikation zwischen unterschiedlichen Plattformen und Server-Applikationen ermöglichen, sind ein klassischer Anwendungsbereich von *Middleware*. Solche Intermediaries sind daher auch Gegenstand dieses Forschungsbereiches (Dikaiakos & Zeinalipour-Yazti 2001). Oft werden unter *Middleware* aber ganze Architekturmodelle und Systemplattformen verstanden, die die Kommunikation mittels einer eigenen Schicht oberhalb des Betriebssystems kapseln, wie man dies bei *DCE*, *CORBA* oder in Ansätzen auch bei *UDDI* und *SOAP* findet (Tanenbaum & Steen 2002; Coulouris, Dollimore et al. 2005).

[251] *Google* bietet seinen Übersetzungsdienst für Webseite in über 60 Sprachen unter http://translate.google.com/ an. „*Yahoo! Babel Fish*“ unterstützt 12 Sprachen: http://babelfish.yahoo.com/.

[252] Dieses Modul reduziert beispielsweise Webseiten auf die wesentlichen Informationen und kann automatisch „`title`“-Attribute zu Hyperlinks hinzufügen, die es aus den Titeln der Link-Ziele extrahiert. Mehr Informationen findet man unter: http://apache.webthing.com/mod_accessibility/.

[253] Anonymizer ist ein Dienst der Firma Anonymizer Inc., der neben dem anonymen Zugriff auf das Web noch weitere Dienste zur Anonymisierung anbietet, wie Wegwerf-E-Mail-Adressen: http://www.anonymizer.com/.

[254] AN.ON ist ein Forschungsprojekt der TU Dresden und der Universität Regensburg zum Datenschutz und zur anonymen Kommunikation im Internet. Die Website des Projekts lautet: http://anon.inf.tu-dresden.de/.

aufzeichnen und zusätzliche Informationen und Schnittstellenelemente (z. B. Fragebögen) einbinden (s. Abschnitt 8.6.4.1 zum Thema „Remote Usability Tests“).

(Rajamony & Elnozahy 2001) verwendeten ein serverseitiges System, das JavaScript-Code in alle Seiten einer Website einbindet und mithilfe dieses Codes die *Ladezeiten* der Webseiten direkt im Browser misst. (Atterer, Wnuk et al. 2006) haben den clientseitigen Intermediary *UsaProxy* entwickelt, der mittels JavaScript die Benutzeraktionen mit dem Browser *innerhalb* von Webseiten registriert und so auch Zugang zum Umgang mit Ajax-Applikationen erhält. Das System *AjaxScope* (Kıcıman & Livshits 2007) dienen zur Optimierung von JavaScript-Programmen. Dabei modifiziert ein Intermediary den übertragenen JavaScript-Code und ergänzt eigene Funktionen, die die Performanz von „Rich Internet Applications“ beim Benutzer analysieren. Es werden die aufgerufenen Funktionen und Ausführungszeiten aufgezeichnet, um den Umgang mit dem System zu optimieren. Das System *Doloto* von Microsoft Research ist darüber hinaus in der Lage, JavaScript-Code automatisch zu optimieren (Livshits & Kıcıman 2008).

Das in diesem Kapitel präsentierte System *Scone* bietet mit der *AccessTracking*-Komponente umfangreiche Möglichkeiten zur Analyse von Benutzeraktionen mit dem Webbrowser (s. Abschnitt 8.6.4). Die mit diesem System durchgeführte Langzeitstudie zur Benutzung des Webs wird in Abschnitt 8.7 kurz vorgestellt (s. auch Weinreich, Obendorf et al. 2006b; Weinreich, Obendorf et al. 2008).

8.4.3 Programmierbare Intermediaries: Ein Überblick

Die vorhergehende Klassifikation zeigt die Vielseitigkeit und fortwährende Aktualität des Intermediary-Konzepts. Die aufgeführten Beispiele nutzen jeweils speziell für den jeweiligen Zweck entwickelte und optimierte Systeme. Sie beruhen aber alle auf ähnlichen technischen Prinzipien, daher bietet es sich an, zumindest zu Entwicklungs- und Forschungszwecken *programmierbare Intermediaries* einzusetzen, die die Realisierung entsprechender Prototypen vereinfachen.

Das Potenzial programmierbarer Intermediaries wurde bereits in den Anfangsjahren des Webs erkannt. Das erste Projekt, in dessen Rahmen ein solches System für das Web beschrieben wurde, ist *OreO* (s. Abschnitt 8.4.1). Das PRojekt lief aber bereits Mitte der 1990er Jahre aus und technische Details sind kaum noch verfügbar. Zahlreiche folgende Projekte präsentierten weitere programmierbare Intermediaries mit jeweils eigenem Anwendungsschwerpunkt (s. auch Malandrino & Scarano 2005).

Im Folgenden werden die Potenziale und Grenzen der bedeutendsten programmierbaren Intermediaries analysiert. Es wird darauf eingegangen, inwieweit sie sich als Basis für die Realisierung der HyperScout-Prototypen eignen und welches Potenzial sie bieten, um als allgemeinere Grundlage für die in Abschnitt 8.2 aufgeführten Anforderungen für Systeme zur Vereinfachung der Orientierung und Navigation im Web zu dienen. Die untersuchten

programmierbaren Intermediaries lieferten zudem Konzepte und Erkenntnisse, die zum Teil für das Scone-Framework berücksichtigt werden konnten und bei seiner Konzeption halfen.

Neben den hier behandelten *programmierbaren* Intermediaries gibt es noch eine ganze Reihe *konfigurierbarer Filtersysteme* für das Web, die mithilfe von Filtersprachen (z. B. auf Basis regulärer Ausdrücke) übertragene Webseiten ändern. Sie dienen in der Regel zur Entfernung von Werbung und von eingebettetem Code, zur Anonymisierung bei der Benutzung des Webs oder zur Analyse der übertragenen Daten (Malandrino & Scarano 2005). Beispiele sind *Proxomitron*, *Proximodo* und *Privoxy*.[255] Da sie aber keine APIs zur programmatischen Erweiterung bieten, wurden solche Systeme ausgeklammert.

8.4.3.1 V6 Web Engine

Die „*V6 Web Engine*" war einer der ersten programmierbaren persönlichen Intermediaries für das Web (Lang & Rouaix 1996b; Lang & Rouaix 1996a).[256] Er bietet ein System von *Filtern* und *Diensten*, die zur Erweiterung der Funktionalität des Webs vorgesehen sind:

- *Filter* setzen sich aus einem *Query-Reply-Filterpaar* zusammen. Der *Query-Filter* kann die Anfrage des Browsers manipulieren und gleichzeitig den *Reply-Filter* für die Antwort des Servers konfigurieren, der die Antwort dementsprechend bearbeitet.
- Zu den Diensten der *V6 Engine* gehört der *HTTP-Client* für die Weiterleitung der Anfragen ins Web, der *Caching Service* zum Zwischenspeichern von Daten und ein *Authentication Service* für die Zugriffskontrolle auf lokal gespeicherte Dokumente. Die *Dienste* der *V6 Engine* werden vom *V6 Distributor* gesteuert.
- Ein *Meta-Service* ist unter einem „virtuellen URI" erreichbar und erlaubt die Konfiguration des Systems über Web-Formulare.[257]

Entwicklungsziel der *V6 Engine* war die Trennung von Präsentation und Funktionalität im Web. Nach Ansicht der Autoren bestand das dringende Problem, dass Webbrowser immer komplexere Applikationen werden, deren Entwicklung bald kaum noch zu handhaben sei, zumal sich Browser zur universellen Benutzungsschnittstelle für alle Arten von Applikationen im Internet entwickelten. Um Kompatibilitätsprobleme zwischen Browsern und Web-Anwendungen zu vermeiden, sollte sich die Entwicklung der Browser auf die grundlegenden Funktionen zur Präsentation von Informationen beschränken, wogegen weitergehende Funktionalität in spezialisierte Intermediaries auszulagern sei. Beispielanwendungen der *V6 Engine* waren ein Dienst zur Verwaltung von Web-Passwörtern und zur Vereinfachung von Log-in-Prozessen, ein Filter zum Anpassen der Dokumente an die Möglichkeiten des

[255] Die Systeme sind unter folgenden Adressen zu finden: http://**proxomitron**.info/, http://**proximodo**.sf.net/ und http://www.**privoxy**.org/.

[256] Die Projekt-Homepage war bis 2008 unter folgender Adresse zu finden: http://pauillac.inria.fr/~rouaix/V6/.

[257] Virtuelle Adressen beinhalten Domain-Namen, die nicht gültig sind, zum Beispiel http://service.v6/services/. Solche Anfragen werden nicht weitergeleitet, sondern vom Intermediary selbst beantwortet. Dieses Konzept wurde auch in Scone realisiert (s. *Scone Robot* im Abschnitt 8.6.2.4 und den *RessourceGenerator* im Abschnitt 8.6.6).

Browsers und einige experimentelle Erweiterungen von HTML und HTTP. Das System ist inzwischen nicht mehr erhältlich und wurde in der kaum verbreiteten funktional-objektorientierten Programmiersprache *Objective Caml* geschrieben,[258] daher kam es für die Entwicklung des geplanten Frameworks nicht infrage.

8.4.3.2 Muffin

Das Open-Source-Projekt *Muffin*[259] ist ein programmierbarer Intermediary in der Sprache Java, der an der San Diego State University entwickelt wurde (Boyns 1998). *Muffin* bietet ein konfigurier- und erweiterbares *Filtersystem*, das Endnutzern mehr Kontrolle über die zum Browser transferierten Daten geben soll. Diverse anpassbare Filter stehen zur Verfügung, mit denen Webseiten analysiert, gespeichert und verändert werden können. Die meisten dienen dazu, unerwünschte Elemente wie Werbung, Animationen und eingebetteten Skript-Code aus Webseiten zu entfernen. Einige einfache Beispiele zeigen, wie *Muffin* die Benutzung des Webs vereinfachen soll: Hierzu gehört ein Filter, der HTML-Formulare bereits bei der Übertragung zum Browser mit persönlichen Standardwerten ausfüllt und so ihre Bearbeitung erleichtert (Boyns 1998).

Muffin ist sehr flexibel, erreicht aber nicht die Performanz und Funktionalität von WBI (s. Abschnitt 8.4.3.5). Ein Defizit stellt die interne Repräsentation von HTML-Dokumenten als Baumstruktur dar. Einerseits lassen sich so die Daten gut bearbeiten und das Kaskadieren mehrerer Filter bereitet keine Probleme.[260] Hingegen ist das Erstellen des Objektbaumes relativ zeitaufwendig und es verzögert die Übertragung, bis das gesamte Dokument in Muffin vorliegt. Zudem fehlen Konzepte zur Persistierung der Daten, zur Verwaltung von Benutzern und zum Zugriff auf weitere Ressourcen im Internet. Insbesondere aufgrund der Performanzdefizite wurde es *nicht* für die Entwicklung des Framworks gewählt.

8.4.3.3 eRACE

Die *extensible Retrieval Annotation Caching Engine (eRACE)* wird von seinen Entwicklern als „Middleware-System" beschrieben, das „die Entwicklung von verteilten Intermediary-Diensten" unterstützt (Dikaiakos & Zeinalipour-Yazti 2001). *eRACE* bietet eine modulare, erweiterbare und verteilte Java-Infrastruktur, die Daten von unterschiedlichen Quellen im Internet sammeln, speichern und verarbeiten kann.[261] Sie unterstützt neben *HTTP* auch die Protokolle *NNTP* und *POP3* (ibid.).

Die *eRACE*-Infrastruktur besteht aus zahlreichen Komponenten, die jeweils über TCP/IP miteinander kommunizieren, wodurch sie verteilt ausgeführt werden können. Die Komponenten für das HTTP-Protokoll werden unter dem Namen *WebRACE* zusammengefasst.

---

[258] *Objective Caml* wurde ebenso wie die *V6 Engine* am INRIA in Frankreich entwickelt: http://caml.inria.fr/ocaml/.

[259] Mehr Informationen zu *Muffin* sind unter folgender Adresse erhältlich: http://muffin.doit.org/.

[260] Die grundlegende Idee, einmal interpretierte Daten zwischen den Filtern weiterzuleiten, wurde auch in dieser Arbeit übernommen, allerdings wurde dabei ein generischer, nicht auf HTML zugeschnittener, Ansatz gewählt.

[261] Die Homepage des Projektes eRACE lautet: http://www.cs.ucy.ac.cy/Projects/eRACE/.

Neben einem *Intermediary* bietet es drei weitere Teilkomponenten: Ein *Crawler* lädt Objekte aus dem Web, der *Object Cache* dient zum Speichern von Daten und die *Annotation Engine*[262] klassifiziert die übertragenen Objekte und versieht sie mit Attributen. Auf diese Weise kann *WebRACE* für Benutzer interessante Dokumente aus dem Internet zusammenstellen und sie z. B. auf geänderte Seiten hinweisen. *WebRACE* wurde als *Software-Agent* konzipiert, der seine Aufgaben unter Berücksichtigung von Anwenderprofilen selbstständig bearbeitet.

Der praktische Schwerpunkt des *eRACE*-Projektes ist die Optimierung der Performanz des Webs. Anstatt Daten in lokalen *Caching Proxies* für Benutzergruppen zwischenzuspeichern, zeigt die *eRACE*-Architektur, wie zukünftig verteilte Intermediaries immer aktuelle Dokumente für alle Benutzer des Webs bereitstellen könnten. Aufgrund dieser Zielsetzung wurde besonderes Augenmerk auf die *Stabilität* und *Performanz* des Systems gelegt.[263] Da *eRACE* für diesen Anwendungsbereich optimiert wurde, sind die verfügbaren Schnittstellen jedoch entsprechend beschränkt.

8.4.3.4 Pluxy

Ein weiterer in Java implementierter modularer Web-Intermediary heißt *Pluxy, the **Plu**ggable Pro**xy***. Pluxy wurde mit dem Ziel entwickelt, „typische Intermediary-Filtermechanismen" zu realisieren und die Implementation von „Browser-Assistenten" zu unterstützen.

Die Programmierung von Pluxy erfolgte in Form von Filter-Plug-ins. Das Plug-in-Konzept von Pluxy stellt eine seiner Stärken dar: Die sogenannten *Pluxins* können dynamisch während der Laufzeit über eine grafische Schnittstelle aktiviert und konfiguriert werden. Für jedes *Pluxin* wird in einer Beschreibungsdatei die Abhängigkeit von anderen Pluxins definiert. Diese Datei dient auch zur automatischen Generierung einer grafischen Konfigurationsschnittstelle für das *Pluxin*.[264]

Das wesentliche Defizit von *Pluxy* ist die beschränkte Funktionalität des Frameworks. Neben der Intermediary-Komponente stehen kaum weitere Werkzeuge zur Verfügung: Es fehlen ein Parser für Dokumente, Persistenz- und Verteilungskonzepte und eine Benutzerverwaltung. Die Beispielanwendung von *Pluxy* war „*Pharos*", ein Empfehlungssystem und Forum für Internet-Gemeinschaften, das das gemeinschaftliche Annotieren von Webseiten erlaubte (Bouthors & Dedieu 1999). *Pluxy* ist inzwischen nicht mehr erhältlich.

---

[262] Die *Annotation Engine* dient nicht zum Hinzufügen von „Anmerkungen" durch die Benutzer, sondern fügt automatisch klassifizierende Daten hinzu, die festlegen, wie mit einem Dokument weiter verfahren werden soll.

[263] In (Zeinalipour-Yazti & Dikaiakos 2001) beschreiben die Autoren, welche Tests und Systemoptimierungen durchgeführt wurden, um die Leistungsfähigkeit von *eRACE* zu optimieren. Diese Ergebnisse wurden für die Entwicklung des Persistenzmodules *NetObjects* (s. Abschnitt 8.6.3) und des Crawlers von Scone (s. Abschnitt 8.6.2) berücksichtigt.

[264] Das Konzept einer Metasprache zur Generierung einer grafischen Schnittstelle zur Konfiguration der Plug-ins wurde ebenfalls in Scone realisiert (s. Abschnitt 8.6.6).

8.4.3.5 WBI

WBI (***W**eb **B**ased **I**ntermediary*) wurde vom *IBM Almaden Research Centre* in San Jose entwickelt (s. Abschnitt 8.4.1). Es dient als Framework zur Erstellung verschiedenartiger Intermediary-Anwendungen in Java. Zum Einsatz von WBI muss ein sogenanntes *WBI-Plug-in* programmiert und im System registriert werden. WBI-Plug-ins setzen sich aus *MEGs* zusammen, die auf unterschiedliche Aufgaben im HTTP-Datenstrom spezialisiert sind (Barrett, Maglio et al. 2000). Ein MEG ist entweder ein ***M**onitor*, ***E**ditor* oder ***G**enerator*: Monitore beobachten die übertragenen Daten, Editoren können in die Übertragung eingreifen und beliebige Änderungen vornehmen, wogegen Generatoren selbst eine Antwort auf die Anfrage des Clients geben. Jeder MEG ist mit einer Aktivierungsregel und einer Priorität versehen, die angeben, unter welchen Bedingungen und in welcher Reihenfolge die MEGs ausgeführt werden sollen (s. Abschnitte 8.6.1 und 8.6.6).

Stärken von WBI sind die einwandfreie Stabilität und hohe Performanz des Intermediary. Die ausführliche Dokumentation aller Klassen und Methoden des Frameworks erleichtert zudem die Entwicklung von WBI-Plug-ins. Eine grafische Schnittstelle visualisiert die Datenverarbeitung in WBI und unterstützt das Debugging.

Allerdings weist WBI als Basis für die HyperScout-Prototypen und ähnliche Projekte erhebliche Defizite auf: Es gibt keine Konzepte zur Identifikation und Verwaltung von Benutzern und es fehlen Komponenten zur Persistierung und Verwaltung von Objekten. WBI enthält zwar einen Parser für HTML-Dokumente, dieser ist aber nur eingeschränkt nutzbar, da er vergleichsweise langsam ist und Probleme mit komplexen und mit syntaktisch fehlerhaften HTML-Dokumenten hat (s. Abschnitt 8.6.5.1). Zudem sind die Funktionen von WBI hauptsächlich auf die Transformation unterschiedlicher Datenformate ausgerichtet; andere Klassen, wie beispielsweise zum Laden von Ressourcen aus dem Internet, sind nur wenig entwickelt oder wurden spezifisch auf die mitgelieferten Beispielanwendungen zugeschnitten.

8.4.4 Bewertung der existierenden Systeme

Jedes der vorgestellten Intermediary-Systeme bietet nur einen Teil der für die Implementation von Web-Navigationserweiterungen benötigten Funktionalität und keines genügte den Anforderungen für die Realisierung der HyperScout-Konzepte. Als Konsequenz erwies es sich als sinnvoll und notwendig, ein neues Framework zu entwickeln, das die in Abschnitt 8.2 definierten Anforderungen erfüllt.

Als Ausgangsbasis für das Framework wurde der Intermediary WBI von IBM gewählt, da er von den untersuchten Systemen insgesamt den ausgereiftesten Eindruck machte (siehe auch Abschnitt 8.6.1) und sich auch gut für den Ausbau zum benötigten Framework eignete.

8.5 Ein Framework zur Erweiterung des Webs: Scone

Zur Unterstützung der in Abschnitt 8.2 spezifizierten Anforderungen für die Entwicklung von Systemen zur Orientierung und Navigation im Web wurde im Rahmen dieser Arbeit das objektorientierte Framework *Scone* konzipiert und entwickelt. Dem Umfang und der Vielgestaltigkeit der Anforderungen wird das Framework durch eine komponentenbasierte Architektur (Griffel 1998) gerecht. Die Komponenten von Scone sind kombinierbar, programmierbar und erweiterbar. Die Schnittstellen der Architektur erlauben das Einbinden weiterer Bibliotheken und Werkzeuge.

Scone ist in der *plattformunabhängigen* Sprache *Java* programmiert, sodass sich das System auf allen gängigen Betriebssystemplattformen einsetzen lässt. Viele grundlegende Eigenschaften von Java – wie die strenge Typisierung und die Objektorientierung – sind für die Entwicklung von Software-Frameworks nahezu unverzichtbar, da sie die Wiederverwendbarkeit und Erweiterbarkeit des Programmcodes wesentlich vereinfachen. Zudem liegen in Java aufgrund seiner Popularität für fast alle Anwendungsbereiche Open-Source-Bibliotheken vor.

Das Scone-Framework besteht aus *vier Kernkomponenten* und einer Reihe von Hilfskomponenten, die je nach benötigter Funktionalität zumeist durch Komposition, zum Teil aber auch durch Erweiterung genutzt werden. Das Framework verwendet ein Plug-in-Konzept, wodurch sich die mit dem System entwickelten Prototypen einfach aktivieren, kombinieren und konfigurieren lassen.

In den folgenden beiden Abschnitten wird erst charakterisiert, wodurch sich *Frameworks* und *Plug-ins* auszeichnen, welche Varianten dieser Konzepte es gibt und wie sich Scone jeweils einordnen lässt (s. Abschnitte 8.5.1 und 8.5.2). Das nächste Teilkapitel 8.6 stellt dann die Architektur des Scone-Frameworks und die einzelnen Komponenten vor.

8.5.1 Charakteristika von Software-Frameworks und Einordnung von Scone

Unter dem Begriff *Framework* wird im Sinne der objektorientierten Systementwicklung[265] eine zusammenarbeitende Menge von Klassen verstanden, die eine funktionale Grundlage zur Realisierung einer bestimmten Kategorie von Software-Systemen bietet. (Gamma, Helm et al. 1994: 26) nehmen in ihrer Spezifikation eine klare Abgrenzung von Frameworks gegenüber der *Klassenbibliothek* bzw. dem *Toolkit* vor:

> „A Framework is a set of cooperating classes that make up a reusable design for a specific class of software… Frameworks thus emphasize design reuse over code reuse, though a framework will usually include concrete subclasses you can put to work immediately. When you use a toolkit, … you write the main body of the application and call the code you want to reuse. When you use a framework, you reuse the main body and write the code it calls."

Entscheidend für ein Framework ist demnach, dass es sich um ein *wiederverwendbares* Design handelt, bei dem alle Bestandteile aufeinander abgestimmt sind. Es ist nicht dafür vor-

[265] Teilweise bezeichnet man in der Informatik mit dem Begriff *Framework* auch *Referenzmodelle* und *Referenzarchitekturen*. Dieses Begriffsverständnis wird hier nicht berücksichtigt.

gesehen, in andere Applikationen integriert zu werden, sondern dient selbst als *einfach anpassbares Skelett* für die Entwicklung von Systemen einer vordefinierten Kategorie. Das Framework gibt die Architektur und den Kontrollfluss der späteren Anwendungen vor (s. auch (Johnson 1997b; Mattsson, Bosch et al. 1999). Damit sich die Komplexität von Frameworks handhaben lässt, sollten bekannte Entwurfsmuster (Design-Patterns) eingesetzt werden (Gamma, Helm et al. 1994; Mattsson, Bosch et al. 1999).

Bei Frameworks wird zwischen dem *White-Box-* und dem *Black-Box-Prinzip* unterschieden (Gamma, Helm et al. 1994: 19ff). Ein White-Box-Framework stellt die Wiederverwendung durch „*Subclassing*" in den Mittelpunkt, also das Erben von Oberklassen und Interfaces. Um ein System mit dem Framework zu realisieren, müssen die Klassen und Interfaces entweder implementiert oder überschrieben werden. Dafür müssen Interna der Oberklassen sichtbar und bekannt sein. Dies macht den Einarbeitungsaufwand in White-Box-Frameworks relativ hoch, sie bieten dem Entwickler aber größere Flexibilität.

Ein Framework nach dem Black-Box-Prinzip setzt hingegen auf die *Komposition* der Klassen. Somit müssen vor allem die Schnittstellen der Klassen präzise beschrieben sein. Die Entwicklung eines Black-Box-Frameworks ist im Allgemeinen aufwendiger, da es bereits einen Großteil der Funktionalität mitbringt. Dem Programmierer bietet es den Vorteil, dass er nur geringe Kenntnisse interner Details der Framework-Klassen benötigt. Dies verkürzt zwar die Einarbeitungszeit, reduziert aber die Flexibilität. Da Black-Box-Frameworks für Entwickler meist schneller zum Erfolg führen und einfacher zu erlernen sind, werden sie häufig bevorzugt. In diesem Sinne definieren Gamma et al. ein Prinzip guten objektorientierten Designs als „Favor object composition over class inheritance" (Gamma, Helm et al. 1994: 20).

In der Praxis wird zumeist keines der beiden Ideale eingehalten. Der Begriff *Gray-Box-Framework* beschreibt den Sachverhalt, dass viele Systeme Aspekte beider Prinzipien in sich vereinen. So wird versucht, sowohl eine schnelle Einsetzbarkeit durch Komposition zu erreichen als auch in bestimmten Bereichen eine hohe Flexibilität zu bieten. Mit steigendem Reifegrad tendieren Frameworks häufig zum Black-Box-Ansatz.

Das im Folgenden vorgestellte Framework Scone lässt sich ebenfalls als *Gray-Box-Framework* einstufen, da sich fast alle Komponenten durch Komposition nutzen lassen, aber gleichzeitig Klassen und Interfaces zur Erweiterung zur Verfügung stehen, um die Anpassbarkeit zu erhöhen. Die wichtigsten *Hotspots* des Scone-Frameworks – die Teile, die bei der Realisierung einer Anwendung im Mittelpunkt stehen (Pree 1994) – werden im Zusammenhang mit den einzelnen Komponenten von Scone beschrieben (s. Abschnitt 8.6 ff).

Die Entwicklung eines Frameworks ist ein anspruchsvoller, recht langwieriger iterativer Prozess (Johnson 1997b; Mattsson, Bosch et al. 1999). Dies trifft auch auf das Scone-Framework zu, dessen Entwicklungsprozess insgesamt (mit Unterbrechungen) mehrere Jahre in Anspruch genommen hat und an dem fünf Personen aktiv beteiligt waren. Durch *Refactoring* wurde Scone schrittweise verbessert und erweitert. Dabei sind die konkreten An-

forderungen und Erfahrungen unterschiedlicher Projekte und Prototypen eingeflossen, bei denen das Framework zum Einsatz kam (s. Abschnitt 8.7).

8.5.2 Plug-ins: Architekturen zur Erweiterung und Konfiguration von Software-Systemen

Der Begriff Plug-in bezeichnet ein Konzept für Programmschnittstellen, das die *dynamische* Integration modularer Software-Komponenten erlaubt. Im Gegensatz zu normalen Software-Bibliotheken lassen sich Plug-ins in der Regel auch von Endanwendern über eine grafische Schnittstelle installieren und konfigurieren. Das Scone-Framework bietet eine *Plug-in-Architektur*. Das Plug-in-Konzept wurde für Scone gewählt, da es den Einsatz und die Konfiguration der mit Scone entwickelten Prototypen vereinfacht.

Bei Beginn der Entwicklung von Scone im Jahr 1999 war das softwaretechnische Konzept von Plug-ins noch nicht derart populär wie heute; inzwischen stellen unterschiedlichste Anwendungen Plug-in-Schnittstellen zur Verfügung, über die Dritte Erweiterungen entwickeln und integrieren können.

Es lassen sich zwei grundlegende Varianten des Plug-in-Konzepts unterscheiden:

- Eine Kategorie von Plug-ins dient dazu, eine eigenständige Applikation um *optionale* Komponenten oder Funktionen zu erweitern. Das Hauptprogramm ist auch ohne Plug-ins nutzbar, lässt sich aber auf diese Weise für persönliche Anforderungen anpassen. Beispiele für den Einsatz dieses Konzept sind Webbrowser (s. Abschnitt 8.3.1) und Programme zum Abspielen von Mediendateien wie *Winamp*; Plug-ins dienen dabei zur Wiedergabe oder Darstellung zusätzlicher Objekttypen. Bei Bildbearbeitungsprogrammen wie *Adobe Photoshop* werden sie zum Einbinden neuer Filter und Effekte eingesetzt.
- Der zweite Ansatz für Plug-ins ist jünger und wesentlich weitgehender. Dabei ist die gesamte Systemarchitektur auf die Nutzung von Plug-ins ausgelegt und *angewiesen*. Solche Systeme bestehen nur aus einem kleinen Kernel, der die Schnittstellen zur Integration und Kommunikation für die Plug-ins bereitstellt und sie verwaltet. Die Funktionalität des Systems wird durch die Plug-ins realisiert. Ein Beispiel hierfür ist die populäre Software-Entwicklungsumgebung *Eclipse:*[266] Sämtliche Funktionen sind über Plug-ins realisiert, sogar die grafische Schnittstelle und Komponenten zum Zugriff auf das Dateisystem, obgleich diese Plug-ins Bestandteil der Standard-Distribution sind. Dieses Konzept macht Eclipse ausgesprochen flexibel und erweiterbar: Durch entsprechende Plug-ins kann es für die Software-Entwicklung in beliebigen Programmiersprachen eingesetzt werden. Es existiert aber beispielsweise auch ein Plug-in, das Eclipse zu einem vollständigen LDAP-Client[267] macht.

---

[266] Das Eclipse-Projekt mit einer Übersicht der Plug-ins ist unter der Adresse http://www.eclipse.org/ zu finden.

[267] LDAP (*Lightweight Directory Access Protocol*) ist ein Protokoll zur Abfrage und Modifikation von Daten eines verteilten Verzeichnisdienstes. Es wird in vielen Bereichen eingesetzt; typische Beispiele sind Adressbücher und Benutzerverzeichnisse.

Das in Scone gewählte Konzept ist eher mit dem von *Eclipse* vergleichbar: Obwohl Scone auch ohne Plug-ins gestartet werden kann, führt es dann keine Operationen durch, außer dass es als Intermediary alle Daten ungefiltert weiterleitet. Alle Erweiterungen auf der Basis des Frameworks werden als Plug-ins implementiert (s. Abschnitt 8.6.6); im Gegensatz zu Eclipse sind bei Scone die Framework-Komponenten selbst nicht als Plug-ins realisiert, sondern über sogenannte *Plug-in Requirements* vom Entwickler zu aktivieren und zu konfigurieren.

8.6 Die Architektur des Frameworks Scone

Dieser Abschnitt gibt einen Überblick über die komponentenbasierte Architektur des Frameworks Scone. Es besteht aus vier Basiskomponenten (s. Abb. 131) und einer Reihe von Hilfskomponenten. Jede der Komponenten ist in einem Java-Paket zusammengefasst und besteht aus zahlreichen Klassen und Interfaces. Einige der Basiskomponenten sind auch unabhängig voneinander nutzbar, allerdings erschließt sich das Potenzial des Frameworks erst bei gemeinsamer Nutzung mehrerer Komponenten.

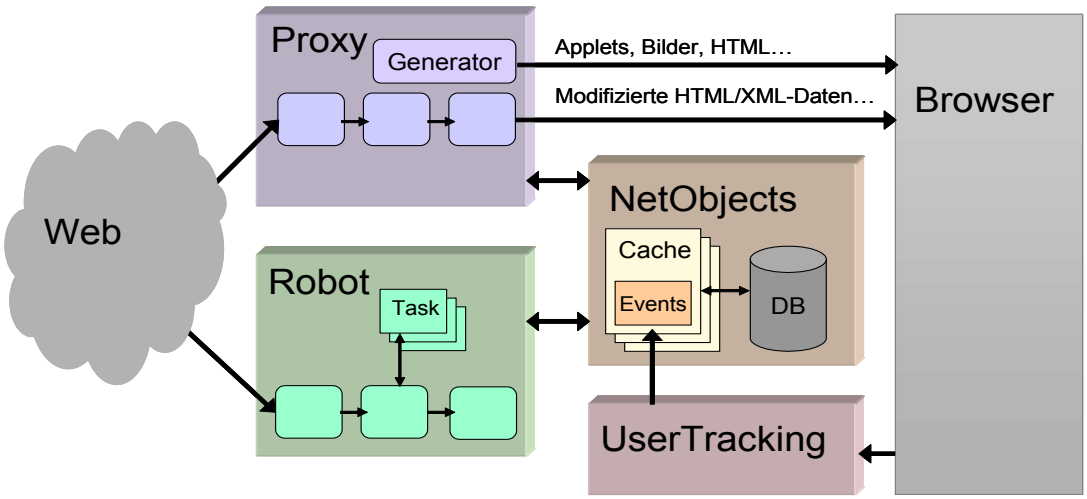


Abb. 131: Die vier Basiskomponenten von Scone

Die erste der vier Basiskomponenten (s. Abb. 131) ist der programmierbare Intermediary *Scone Proxy*. Er kann auf die zwischen Browsern und Webservern transferierten Daten zugreifen, sie modifizieren und neue Objekte generieren (s. Abschnitt 8.6.1). Zweitens ermöglicht ein flexibler, regelbasierter, semi-autonomer und nebenläufiger Web-Crawler, der *Scone Robot*, das Zusammenstellen von Ressourcen aus dem Web (s. Abschnitt 8.6.2). Die dritte Komponente *Scone NetObjects* realisiert ein objektorientiertes Datenmodell in Form persistierbarer Java-Objekte zur Repräsentation der wichtigsten „Elemente“ des Webs, wie URIs, Web-Ressourcen und Hyperlinks (s. Abschnitt 8.6.3). Die vierte Basiskomponente ist das *Scone UserTracking*, das die Benutzeraktionen mit dem Browser erfassen und aufzeichnen kann (s. Abschnitt 8.6.4). Hinzu kommen mehrere Hilfskomponenten, die grundlegende, häufig benötigte Funktionen bereitstellen (s. Abschnitte 8.6.5ff).

Alle Komponenten bieten aufeinander abgestimmte APIs und sind in eine Plug-in-Architektur integriert (siehe Abschnitte 8.5.2 und 8.6.6). Die Kombination der Komponenten erlaubt die schnelle Entwicklung vielfältiger Arten von Erweiterungen für das Web, wobei der Schwerpunkt auf Werkzeugen zur Vereinfachung der Navigation und Orientierung liegt (s. Abschnitt 8.2).

8.6.1 Eine erweiterte Intermediary-Architektur: *Scone Proxy*

Die erste Kernkomponente von Scone ist der programmierbare Web Intermediary *Scone Proxy* (s. Abschnitt 8.4.1). Scone Plug-ins können sich mit seiner Hilfe in den Datenstrom zwischen Browser und Server „einklinken" und erhalten Zugriff auf alle übertragenen Objekte. Die im Browser dargestellten Dokumente können so analysiert, angepasst und ergänzt werden (s. Anforderungsanalyse, Abschnitte 8.2.1 und 8.2.3).

Als Basis für die Intermediary-Komponente von Scone dient das ***WBI Development Kit*** (Barrett & Maglio 1998; Barrett, Maglio et al. 2000) des *IBM Almaden Research Centers* (San José, Kalifornien), das um einige Konzepte erweitert wurde, die teilweise auch Eingang in das Original-Framework von IBM fanden (s. Abschnitt 8.4.3.5). WBI ist aus den im Abschnitt 8.4.3 vorgestellten Intermediaries für das Scone-Framework ausgewählt worden, da es zahlreiche Stärken aufweist:

- Analysen vor Beginn der Entwicklung von Scone zeigten, dass es sich um den mit Abstand performantesten Intermediary in Java handelt.
- Die API ist ausführlich dokumentiert.
- WBI lief bei Tests im Gegensatz zu den meisten anderen Projekten auch bei längerer Nutzung stabil.
- WBI bietet eine grafische Debugging-Schnittstelle, die die Datenübertragung und die Aktivitäten der installierten Plug-ins detailliert darstellt (s. Abb. 132).

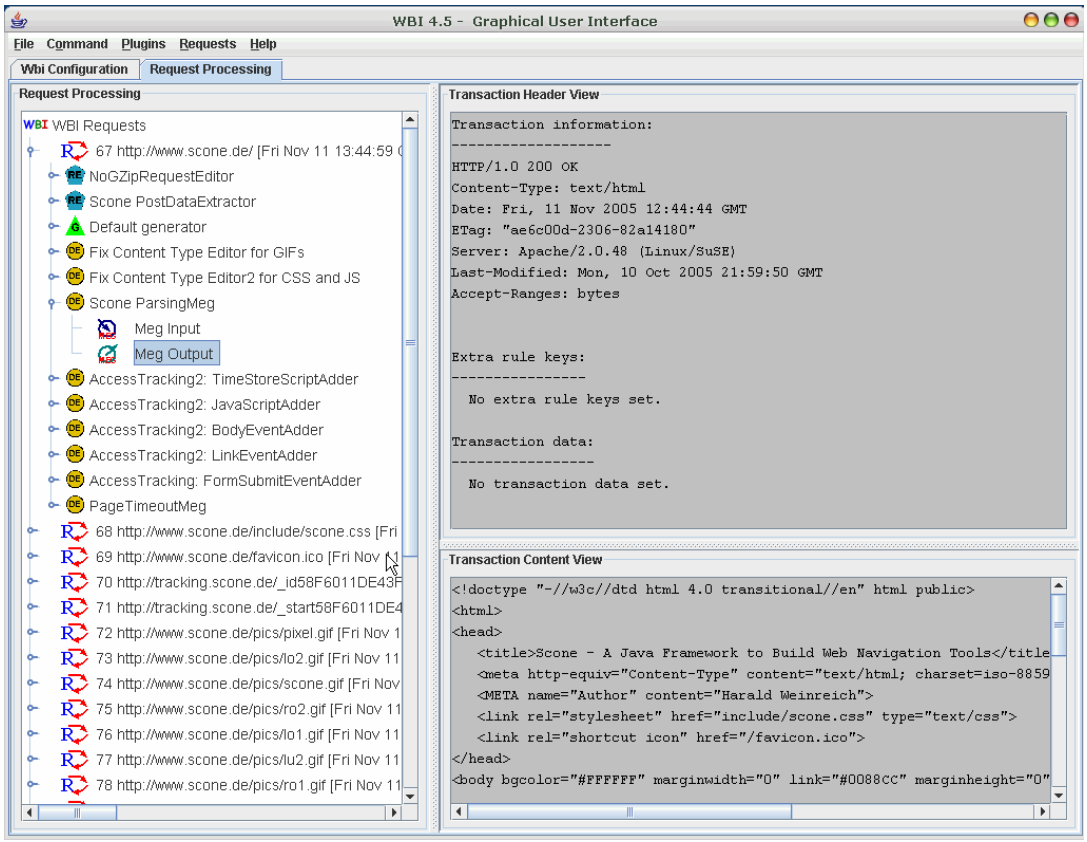


Abb. 132: WBIs grafische Schnittstelle für die Verarbeitung einzelner Anfragen.

Konzeptionell basiert WBI auf der Programmierung sogenannter MEGs, ein Akronym für **M**onitor/**E**ditor/**G**enerator. Diese von WBI angebotenen Klassen sind jeweils auf charakteristische Aufgaben spezialisiert:



Ein *Monitor* beobachtet die Übertragung und hat Zugriff auf alle Daten, sowohl den „Request" des Browsers als auch die „Response" des Servers. Er kann aber keine Änderungen vornehmen.



Der *RequestEditor* dient zum Bearbeiten von Anfragen. So lassen sich beispielsweise die HTTP-Parameter eines Browser-Requests manipulieren.



Der *DocumentEditor* dient zur Manipulation der Antwort von Servern. Dabei kann nicht nur auf die Parameter des HTTP-Protokolls zugegriffen werden, sondern auch auf den Datenstrom mit dem vom Server übertragenen Objekt.



Ein *Generator* kann eine Antwort auf die Anfrage eines Browsers erzeugen. Entsprechende „Requests" werden nicht an den Webserver weitergeleitet, sondern bereits durch den Intermediary beantwortet.

Aus der Kombination dieser MEGs lassen sich beliebige Funktionen innerhalb des Intermediary realisieren. Abb. 133 zeigt exemplarisch, wie MEGs eingesetzt werden: Die Anfrage des Browsers wird vom *Request-Editor* bearbeitet und zum Server weitergeleitet. Alternativ kann eine Anfrage aber auch direkt durch einen *Generator* beantwortet werden. Die Server-Antwort wird in der Illustration zuerst durch einen *Monitor* überwacht und dann durch zwei MEGs vom Typ *Document Editor* bearbeitet.

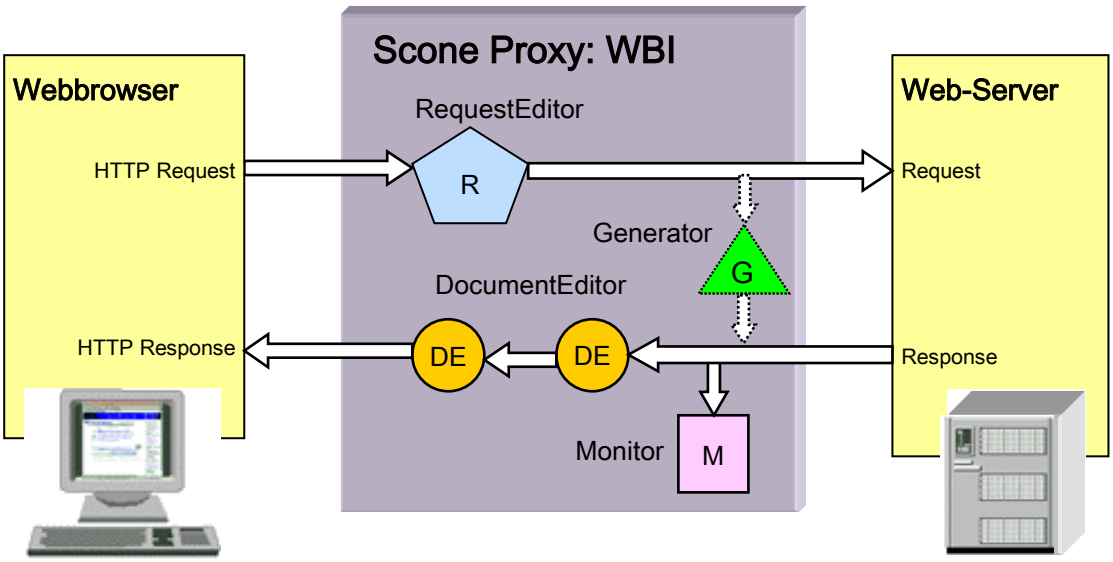


Abb. 133: Schematischer Einsatz von MEGs innerhalb der Proxy-Komponente von Scone.

Die Bearbeitungsreihenfolge der Komponenten (z. B. der zwei *DocumentEditors* in Abb. 133) und ihre Aktivierung lassen sich über *Regeln* und *Prioritäten* festlegen. Eine Regel kann beispielsweise bewirken, dass ein MEG nur für bestimmte *Mime Types* aktiviert wird.

Der Code in Abb. 134 zeigt, wie man innerhalb von *Scone* den MEG der Klasse *MyDocumentEditor* installiert, der nur Objekte des Typs „`text/html`“ bearbeiten soll. Es lassen sich auch beliebige andere Attribute des HTTP-Protokolls berücksichtigen, wie das Ziel der Anfrage oder die Dateigröße des Objekts. Die Priorität (in diesem Falle „50“) gibt die Ausführungsreihenfolge der MEGs an, wobei MEGs mit höherem Prioritätswert zuerst ausgeführt werden. Ein ausführlicheres Beispiel findet man im Abschnitt 8.6.6, der auf die Programmierung von Scone Plug-ins eingeht.

```
MyDocumentEditor myDocumentEditor = new MyDocumentEditor();
myDocumentEditor.setup("My first Editor", "content-type=text/html", 50);
addMeg(myDocumentEditor);
```

Abb. 134: Registrieren eines WBI-MEGs innerhalb eines Scone Plug-ins.

Mehr Details zu WBI und seiner Programmierung findet man unter anderem in (Barrett & Maglio 1998; Barrett, Maglio et al. 2000; Maglio & Barrett 2000; Ihde, Maglio et al. 2001).

8.6.1.1 Integration und Erweiterung des Intermediary WBI

WBI wurde nicht nur als eine Kernkomponente in das Framework Scone integriert, sondern auch in wesentlichen Punkten angepasst und erweitert. Mehrere der Modifikationen sind in Kooperation mit den WBI-Entwicklern von IBM in Almaden (USA) vorgenommen worden und in die offizielle WBI-Version eingeflossen.

Die tiefgreifendste Änderung war die Umstellung der Schnittstellen zur Datenübertragung zwischen den MEGs. Die bei Beginn des Scone-Projekts verfügbare WBI-Version bot ausschließlich Schnittstellen für binäre Datenströme zwischen den MEGs an. Das führte dazu,

dass jeder MEG den Datenstrom selbst parsen und dann zur Weiterleitung zum nächsten MEG wieder in einen Bytestrom umwandelten musste. Dies erwies sich in der Praxis als schwerfällig und ressourcenhungrig, sobald mehrere MEGs verwendet wurden, da sie jeweils den Datenstrom wieder und wieder parsten.

Zur Lösung dieses Problems sind im Rahmen dieser Arbeit die *ObjectStreams* für Intermediaries entwickelt worden, die die Übergabe beliebiger Objekttypen zwischen MEGs ermöglichen. Wenn mehrere MEGs beispielsweise auf einem DOM arbeiten, muss der DOM-Objektbaum nur einmal erstellt werden und kann dann direkt an weitere MEGs übergeben werden. Die beteiligten MEGs verwenden dann DOM Input Stream (DIS) und DOM Output Stream (DOS); erst *nach* dem letzten auf dem DOM arbeitenden MEG wird der Datenstrom automatisch wieder in einen Bytestrom umgewandelt (s. Abb. 135). Die Entwicklung fand im Sommer 2000 vom Autor dieser Dissertation zusammen mit Volkert Buchmann statt, die Integration erfolgte danach in Zusammenarbeit mit dem IBM Almaden Research Center (s. auch: Buchmann 2002).

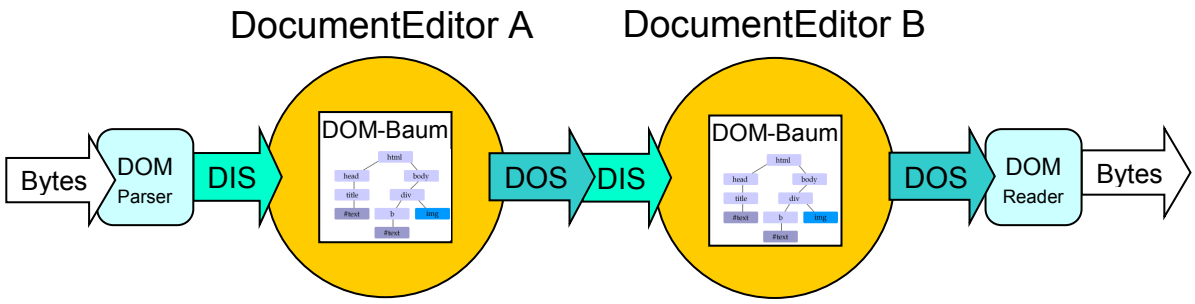


Abb. 135: Schematische Darstellung der WBI-ObjectStreams. Im Beispiel sind die Objekte DOM-Bäume, es werden daher DOM-InputStreams und DOM-OutputStreams verwendet.

Basierend auf den Objektstreams ist das Konzept der *TokenStreams* für die Bearbeitung von HTML-Dokumenten in Intermediaries entwickelt worden.[268] Anlass war der Mangel an Java-Techniken zur Bearbeitung von HTML-Dokumenten, die den Anforderungen an Scone genügten.

Die beiden etabliertesten Verfahren für die Bearbeitung von XHTML- und XML-Dokumenten in Java sind *DOM*, das *Document Object Model* (Hégaret, Whitmer et al. 2005), und SAX, die *Simple API for XML* (Brownell 2002). Beim DOM wird aus der XML-Datei ein hierarchisch zu navigierender Objektgraph erstellt. Ein Schwachpunkt des DOM-Konzepts für den Einsatz in Scone ist, dass zur Erstellung des Objektbaumes das gesamte Dokument benötigt wird. Es führt dazu, dass die Datei erst vollständig vom Intermediary geladen werden muss, bevor sie bearbeitet und an den Browser weitergeleitet werden kann; als Folge kommt es zu störenden Verzögerungen bei der Übertragung. Dieses Problem tritt bei SAX nicht auf, das

[268] Der Intermediary *Muffin* (Boyns 1998) verwendet ebenfalls *Tokenstreams* zwischen den einzelnen Filtern (s. Abschnitt 8.4.3.2). Sie dienten als Inspiration für die *Scone HtmlTokenStreams*, erfordern aber vor der Bearbeitung, dass das gesamte Dokument im Intermediary vorliegt und sind daher weniger performant.

eine event-basierte API bietet und die Daten Elementweise an den Browser weiterreichen kann. Allerdings wurden sowohl DOM als auch SAX für XML-Dateien konzipiert, wodurch es mit (X)HTML-Dokumenten im Web regelmäßig zu Problemen kam, da sie sehr häufig syntaktische Fehler im HTML-Code aufweisen. Dieses Problem ließ sich auch nicht durch den Einsatz von XHTML-Konvertern oder HTML-Tidy[269] lösen.

Für Scone wurde daher das Open-Source-Projekt *HtmlStreamTokenizer*[270] zum Parsen von HTML-Dokumenten angepasst und integriert. Dieser Parser verhält sich ähnlich zur Standard-Java-Klasse *StreamTokenizer*, zerlegt aber HTML-Dokumente in *HTML-Token*. Aufgrund der Modifikationen für Scone ist er erheblich toleranter gegenüber Syntaxfehlern im HTML-Code als die zuvor aufgeführten Techniken (siehe Abschnitt 8.6.5.1).

Das Konzept der *TokenStreams* nutzt die Ausgabe des *HTMLStreamTokenizers* und sorgt dafür, dass die zu übertragenden Daten nur *einmal* in (X)HTML-Token zerlegt werden müssen. Gemäß dem Konzept der Object-Streams wird zwischen den MEGs das *TokenStream-Objekt* übergeben. Darüber hinaus bieten die *TokenStreams* von Scone Zugriff auf alle Daten des HTTP-Protokolls. Sie entsprechen damit den spezifischen Anforderungen zur Bearbeitung von Webseiten innerhalb von Intermediaries.

Zusätzlich zu den beiden aufgeführten Erweiterungen in WBI bietet Scone mehrere Standard-MEGs, die häufig benötigte Funktionen bereitstellen. Hierzu gehören der *DocumentParser* zur Extraktion von Metadaten aus Webseiten (s. Abschnitt 8.6.5.3), das *ServerSide-Plug-in* zum Einsatz von Scone als „reverse Proxy“ auf Serverseite (s. Abschnitt 8.6.5.6), mehrere MEGs zur Korrektur von fehlerhaften Angaben im HTTP-Protokoll[271] und ein *Generator*, mit dessen Hilfe Scone selbst zum Webserver wird.

8.6.2 Ein persönlicher, programmierbarer Web-Crawler: *Scone Robot*

Die zweite Basiskomponente von Scone ist ein *Web-Crawler* bzw. *Web-Robot*.[272] Er ist – ähnlich den Crawlern von Suchmaschinen – in der Lage, ganze Websites zu indexieren, indem er den Links in den Seiten folgt. Der Crawler mit dem Namen „*Scone Robot*“ wurde aber für die Entwicklung von Navigationswerkzeugen entwickelt und für den Einsatz als persönlicher

---

[269] HTML Tidy ist ein Programm, das viele Fehler in HTML-Code erkennt und korrigiert. Es wurde ursprünglich von Dave Ragget für das W3C entwickelt (http://www.w3.org/People/Raggett/tidy/) und wird nun als Open-Source-Projekt weitergeführt: http://tidy.sourceforge.net/.

[270] Der HtmlStreamTokenizer stammt von Arthur Do: http://sourceforge.net/projects/htmltok/.

[271] Ein häufig auftretendes Beispiel sind fehlerhafte MimeType-Angaben der Web-Server. Ursache sind zumeist Fehlkonfiguration der Server. Beispielsweise werden oft GIF-Bilder oder JavaScript- und CSS-Dateien als „TEXT/HTML“ klassifiziert. Browser korrigieren diese Fehler automatisch, indem sie den Kontext berücksichtigen (z.B. wenn es sich um eine eingebundene Grafik handelt), Scone verwendet dafür MEGs, die den Inhalt der Dokumente analysieren (siehe: http://de.wikipedia.org/wiki/Magische_Zahl_%28Informatik%29).

[272] Die Begriffe *Web-Crawler* und *Web-Robot* werden hier synonym verwendet. Oft findet man in der Literatur auch noch die Begriffe *Web-Spider*, *Web-Agent* und *Gatherer* (Koster 1993; Koster 1994b; Bowman, Danzig et al. 1995; Eichmann 1995; Miller & Bharat 1998).

Web-Crawler optimiert. Er sollte aufgrund der Anforderungsanalyse (s. Abschnitt 8.2.2) folgende Eigenschaften aufweisen:

- Der Crawler soll selbsttätig Informationen zusammenstellen können.
- Die Aufträge an den Crawler müssen über zahlreiche Parameter steuerbar sein.
- Aufträge sind nach Prioritäten zu bearbeiten.
- Hinfällig gewordene Aufträge müssen identifiziert und abgebrochen werden können.
- Der Crawler ist für den Einsatz auf Clientseite zu optimieren, beispielsweise soll er die zur Verfügung stehenden Ressourcen berücksichtigen.
- Er soll dennoch einfach zu programmieren sein.

Im nächsten Abschnitt 8.6.2.1 werden erst verwandte Lösungsansätze vorgestellt. Dann wird auf die Architektur und die Programmierung des „*Scone Robot*“ eingegangen (Abschnitte 8.6.2.2 und 8.6.2.3). Abschnitt 8.6.2.4 beschreibt die grafische Schnittstelle zur Steuerung und zum Debugging des Systems.

8.6.2.1 Persönliche Web-Crawler

Bereits auf den ersten World Wide Web-Konferenzen wurden unterschiedliche Systeme vorgestellt, die das Web anhand der Links durchsuchen können, um die gefundenen Seiten lokal bereitzustellen. Beispiele sind *WebCrawler* (Pinkerton 1994), *ALIWEB* (Koster 1994a), der *RBSE-Spider* (Eichmann 1994) und der *MOMspider* (Fielding 1994). Ein weiteres wegbereitendes Projekt war das verteilte Suchsystem *Harvest*, das bereits die ressourcenschonende dezentrale Indexierung von Dokumenten unterstützte – eine Technik, die sich bis heute leider im Web nicht durchsetzen konnte (Bowman, Danzig et al. 1995). Alle globalen Suchmaschinen nutzen einen lokalen Index, der mithilfe solcher Web-Crawler erstellt wird (vergl. Abschnitte 3.1.1 und 3.3.4).

Persönliche Crawler, die für einzelne Benutzer oder Benutzergruppen arbeiten, wurden erst später zum Forschungsgegenstand. Dies liegt vor allem darin begründet, dass in der Anfangszeit des Webs die Bandbreite des Internets und die Belastbarkeit von Webservern sehr beschränkt waren. Aus diesem Grund wurden auch Richtlinien für die Entwicklung von Web-Crawlern erstellt, die unter anderem ihre Nutzung für persönliche Zwecke stark einschränkten (Koster 1993; Eichmann 1995).[273]

Im Gegensatz zu den recht gut erforschten Suchmaschinen-Crawlern, die möglichst viele Dokumente des Webs im Suchindex aktuell halten sollen (vergl. Abschnitt 3.3.4), ergeben sich für persönliche Cralwer je nach Einsatzgebiet recht spezifische und davon abweichende Anforderungen. Der HyperScout-Client benötigt beispielsweise Daten, die möglichst zeitnah

[273] Diese Richtlinien müssen heute relativiert betrachtet werden, da sich die Rahmenbedingungen bezüglich der verfügbaren Bandbreiten und der Leistungsfähigkeit der Webserver erheblich geändert haben.

und unter Berücksichtigung der beschränkten Bandbreite des Anwenders zusammengestellt werden. Da Benutzer oft schnell im Web navigieren, kommen nicht nur häufig neue Aufträge hinzu, sie können sich auch rasch ändern oder kurz nach Beginn schon wieder veralten.

Zu den ersten persönlichen Web-Crawler, die Daten aus dem Web „exklusiv" für einzelne Personen aufbereiteten, gehörten Agenten-Systeme wie der personalisierbare Nachrichtendienst *Krakatoa Chronicle* (Kamba, Bharat et al. 1995) und der *ShopBot* zum automatischen Preisvergleich (Doorenbos, Etzioni et al. 1997). Inzwischen gibt es zahlreiche kommerzielle, persönliche Agenten, die für Anwender Informationen bereitstellen, wie die lokale Meta-Suchmaschine *Copernic Agent*[274] und der Offline-Reader *WebReaper*.[275]

Das Java-Framework *WebSphinx*[276] der Carnegie Mellon University war als einer der ersten persönlichen Web-Crawler nicht für eine bestimmte Anwendung spezialisiert, sondern erlaubte die einfache Entwicklung von site-spezifischen, personalisierbaren und migrationsfähigen Crawlern (Miller & Bharat 1998). *WebSphinx* führte das Konzept der *Classifier-Methode* ein, die zum Klassifizieren von Web-Dokumenten dienten und den Crawler steuerten. Für den „*Scone Robot*" wurde dieses Konzept zum *Classifier-Filter-Konzept* verfeinert (s. Abschnitt 8.6.2.2).

WebSphinx bot zudem eine interaktive Entwicklungsumgebung namens *Crawler Workbench*, in der Aufträge für den Cralwer erstellt und visualisiert werden konnten. *Scone* bietet in Anlehnung daran den *Robot-Monitor*, über den sich der Crawler steuern lässt (s. Abschnitt 8.6.2.4).

8.6.2.2 Architektur und Funktionsweise des Crawlers

Bei der Konzeption des „*Scone Robot*" wurden die in Abschnitt 8.6.2 aufgeführten Anforderungen zugrunde gelegt und die Erfahrungen der im vorherigen Abschnitt aufgeführten Projekte berücksichtigt. Als Ergebnis sollte der Cralwer den Anforderungen der meisten Web-Orientierungs- und Navigationssysteme bezüglich des Ladens von Ressourcen aus dem Internet genügen.

Der Crawler arbeitet *nebenläufig* und ist in der Lage, *mehrere Aufträge gleichzeitig* zu bearbeiten sowie mehrere Objekte *parallel herunterzuladen*. Dennoch kann die Komponente als leichtgewichtig bezeichnet werden, zumal sie aus weniger als 30 Klassen besteht. Der Crawler weist eine modulare Architektur auf und für die Funktionalität bedeutende Klassen und Interfaces – wie die Warteschlange oder die Steuerung des Crawl-Vorgangs – lassen sich anpassen oder ersetzen.

---

[274] Copernic Agent wird von Copernic Technologies Inc. hergestellt: http://www.copernic.com/.

[275] Mark Otways Projekt WebReaper ist hier beheimatet: http://www.webreaper.net/.

[276] WebSphinx steht für **Web**site-**S**pecific **P**rocessor for **H**TML **IN**formation e**X**traction. Die Homepage des Projektes lautet: http://www.cs.cmu.edu/~rcm/websphinx/.

Die Komponente „*Scone Robot*“ arbeitet *regelbasiert* und versucht für den Benutzer auf die jeweils optimale Weise die benötigten Informationen zusammenzustellen. Er weist die Eigenschaften eines *reaktiven, semi-autonomen Software-Agenten* (Wooldridge 2002) auf, da er auf bestimmte Ereignisse – beispielsweise die Übertragung einer Seite – reagiert und daraufhin bestimmten Regeln folgt, und diese dann weitgehend *unabhängig* bearbeitet. Dabei handelt er *adaptiv,* indem er je nach den Möglichkeiten des Servers mehr oder weniger Informationen bereitstellt und auf die jeweilige Rechner- und Netzwerkbelastung berücksichtigt.

Abb. 136 zeigt einen schematisierten Überblick der Architektur des „*Scone Robot*“ als UML-Klassendiagramm: Aufträge werden an die Komponente gestellt, indem der Klasse *Robot* ein Objekt vom Typ *RobotTask* übergeben wird (s. Abb. 136, oben). Die Klasse *Robot* ist nach dem Singleton-Pattern implementiert, da dies die zentrale Vergabe von Ressourcen vereinfacht (Gamma, Helm et al. 1994).

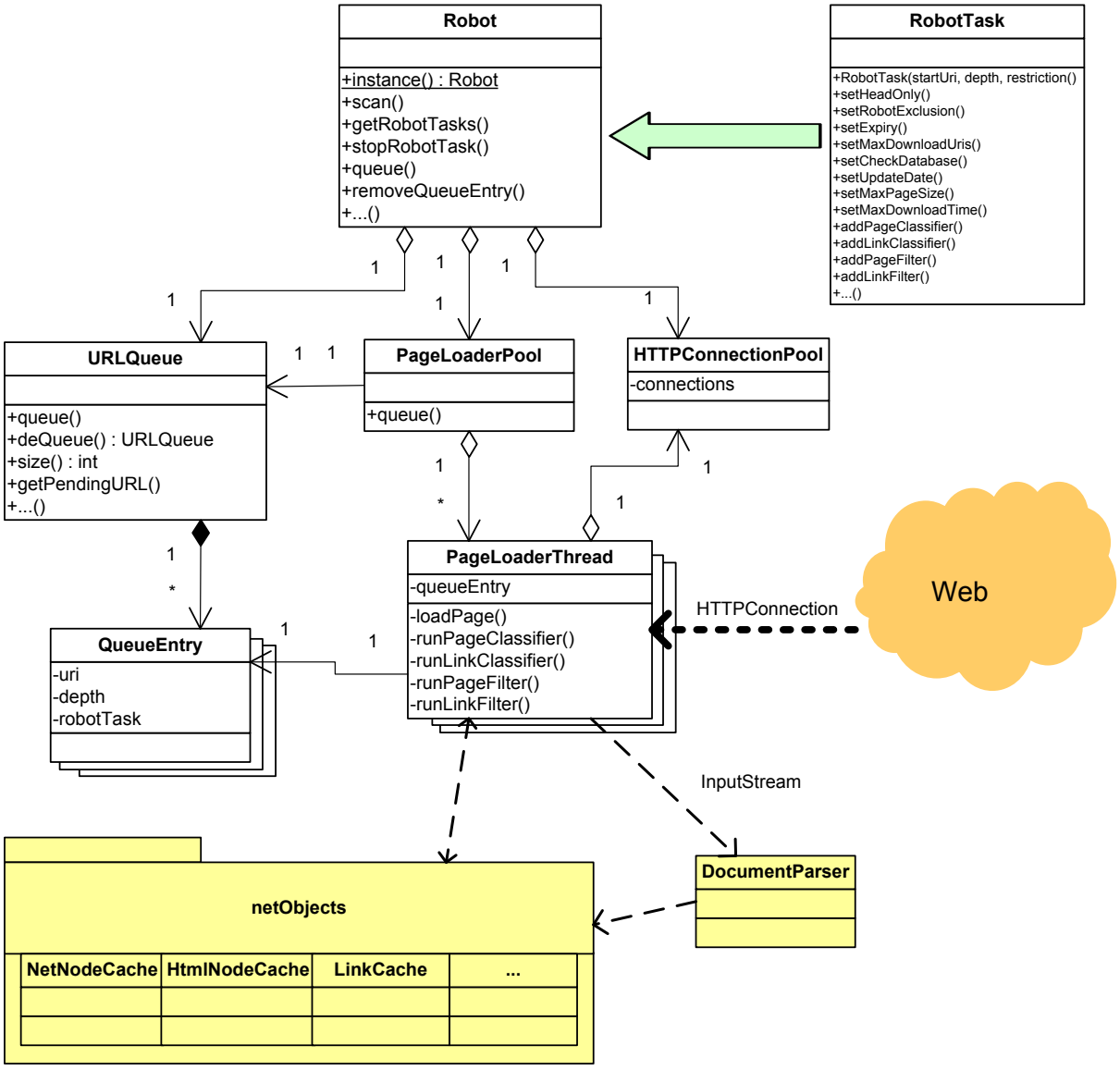


Abb. 136: Ein schematischer Überblick über die Architektur des „*Scone Robot*“

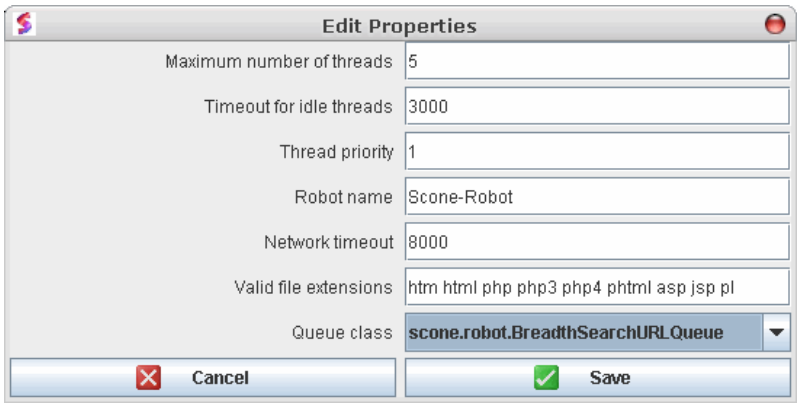


Abb. 137: Das grafische Interface zur Basis-Konfiguration des „*Scone Robot*“

8.6.2.3 Programmierung und Steuerung des Crawlers

Die Konfiguration und Steuerung des Crawlers basiert auf mehreren Mechanismen. Grundlegende Parameter lassen sich über die grafische Schnittstelle zur Konfiguration von Scone festlegen (s. Abb. 137), wie die Anzahl der insgesamt parallel arbeitenden *PageLoaderThreads*, Standard-Timeouts für die Übertragung, gültige Dateitypen[277] und die verwendete Suchstrategie. Eine neue Suchstrategie ist durch das Ersetzen der entsprechenden Klasse realisierbar, wobei Scone bereits drei Klassen bietet, die unterschiedliche Warteschlangen implementieren. Hierzu gehören die Breiten- und die Tiefensuche[278] (s. Wollenweber 2002).

Einzelne Aufträge an den Cralwer werden hingegen über die Parameter der Klasse *RobotTask* spezifiziert. Sie legen fest, *welche* Ressourcen *wie* bereitgestellt werden sollen. Obligatorische Parameter sind *StartURI*, die Adresse mit der ein Crawl-Vorgang begonnen wird, die *Tiefe*, bis zu der – ausgehend von dem *StartURI* – rekursiv Links verfolgt werden sollen und eine *Bedingung* („*Restriction*“), die beispielsweise angibt, ob nur interne oder auch externe Links zu berücksichtigen sind (s. Abb. 138, Zeile 2). Es gibt noch über zehn weitere optionale Parameter, wie die Anweisung, dass nur *HTTP-Header* übertragen werden sollen und unterschiedliche Beschränkungen für den Umfang der zu ladenden Daten.

Auf noch präzisere Weise lässt sich der Crawler mittels des *Classifier-Filter*-Konzepts steuern (s. Abschnitt 8.6.2.1). Die *Classifier-* und *Filter-Klassen* dienen dazu, sowohl alle *Links* als auch alle *Dokumente* aufgrund ihrer Eigenschaften zu *klassifizieren* und dann gemäß der *Filterbedingungen* weiter zu behandeln. Diese Trennung hilft bei der strukturierten Definition komplexer Regeln zur Erfassung von Ressourcen im Web. Das Beispiel in Abb. 138 zeigt, wie mithilfe dieses Konzepts der Crawler nur den Links folgt, die einen bestimmten Text als Teil des Link-URI aufweisen.

---

[277] Der Dateityp zu verfolgender Links wird bei Web-Robots über die Endung des Dateinamens im URI spezifiziert, da der MimeType erst nach der Übertragung des HTTP-Headers des Link-Zieles zur Verfügung steht.

[278] Die Breitensuche ist in der Regel zu bevorzugen, da sie mit weniger Problemen behaftet ist (Pinkerton 1994). *Scone* bietet gleich zwei unterschiedliche Breitensuch-Strategien an, die sich darin unterscheiden, wie mit konkurrierenden Aufträgen an den Robot umgegangen wird. Genauere Informationen findet man in der API-Dokumentation und in (Wollenweber 2002).

```
/* Aufgabe: Erfasse die Website www.scone.de bis zur Tiefe drei. Folge nur internen Links. */
rt = new RobotTask(new SimpleUri("http://www.scone.de/"), 3, RobotTask.INTERNAL, this);
rt.setCheckDatabase(true); // Überprüfe, ob die Seite bereits in der DB ist.
rt.setMaxPageSize(10000);  // Es sollen nur die ersten 10kB von jedem Dokument geladen werden.
// Verwende einen Link-Classifier und einen Link-Filter.
rt.addLinkClassifier(new MyLinkClassifier());
rt.addLinkFilter(new MyLinkFilter());
robot.scan(rt);
...
/* Markiere alle Links, die zur Scone Java API Dokumentation führen. */
private class MyLinkClassifier implements LinkClassifier {
    public void classify(RobotLink robotLink, RobotHtmlNode robotHtmlNode, QueueEntry qe) {
        String urlText = robotLink.getLink().getToNode().getUri();
        if (urlText.indexOf("javadoc") != -1) {
            robotLink.setAttribute("Java API", "Yes");
        }
    }
}
...
/* Folge nicht den Links zur Java API Dokumentation. */
private class MyLinkFilter implements LinkFilter {
    public boolean filter(RobotLink robotLink, RobotHtmlNode robotHtmlNode, QueueEntry qe) {
        if (robotLink.getAttribute("Java API") == "Yes") {
            return false;
        }
    }
}
```

Abb. 138: Ein einfaches Beispiel für die Programmierung des „*Scone Robot*“ (Code-Auszug).

Der *Scone Robot* bietet vielfältige Wege, um die beschränkten Ressourcen auf Clientseite effizient zu nutzen: Neben der Definition von *Time-outs* und der Begrenzung der zu übertragenden Datenmenge für einzelne Dokumente ist es auch möglich, auf in Bearbeitung befindliche Aufträge zuzugreifen und sie zu ändern oder sie abzubrechen.

Mit all diesen Mechanismen lässt sich präzise definieren, welche Informationen der Crawler erfassen soll. Er kann so beispielsweise lediglich Dokumenttypen ermitteln, Antwortzeiten eines Servers abschätzen oder auch Titel und Größe von Webseiten bestimmen. Weitere Details zur Architektur und Funktionsweise des „*Scone Robot*“ werden in (Wollenweber 2002: 48ff) beschrieben.

8.6.2.4 Ein grafisches Interface für persönliche Crawler: *RobotMonitor*

Für Entwickler wird ergänzend das Plug-in *RobotMonitor* angeboten, das eine Benutzungsschnittstelle zur Kontrolle und Steuerung des *Scone Robot* aktiviert. Mit dem Webbrowser wird unter dem virtuellen URI `http://_robot.scone.de/` auf sie zugegriffen (s. Abb. 139). Das Plug-in dient zu Testzwecken und vereinfacht das Debugging anderer Plug-ins. Es gibt einen Überblick über die in Bearbeitung befindlichen Aufträge, listet die Anzahl bereits bearbeiteter Objekte auf und zeigt die noch in der Queue anstehenden Sub-Tasks (s. Abb. 139, oben). Darüber hinaus lassen sich Aufträge abbrechen und neue Aufträge spezifizieren (s. Abb. 139, unten).

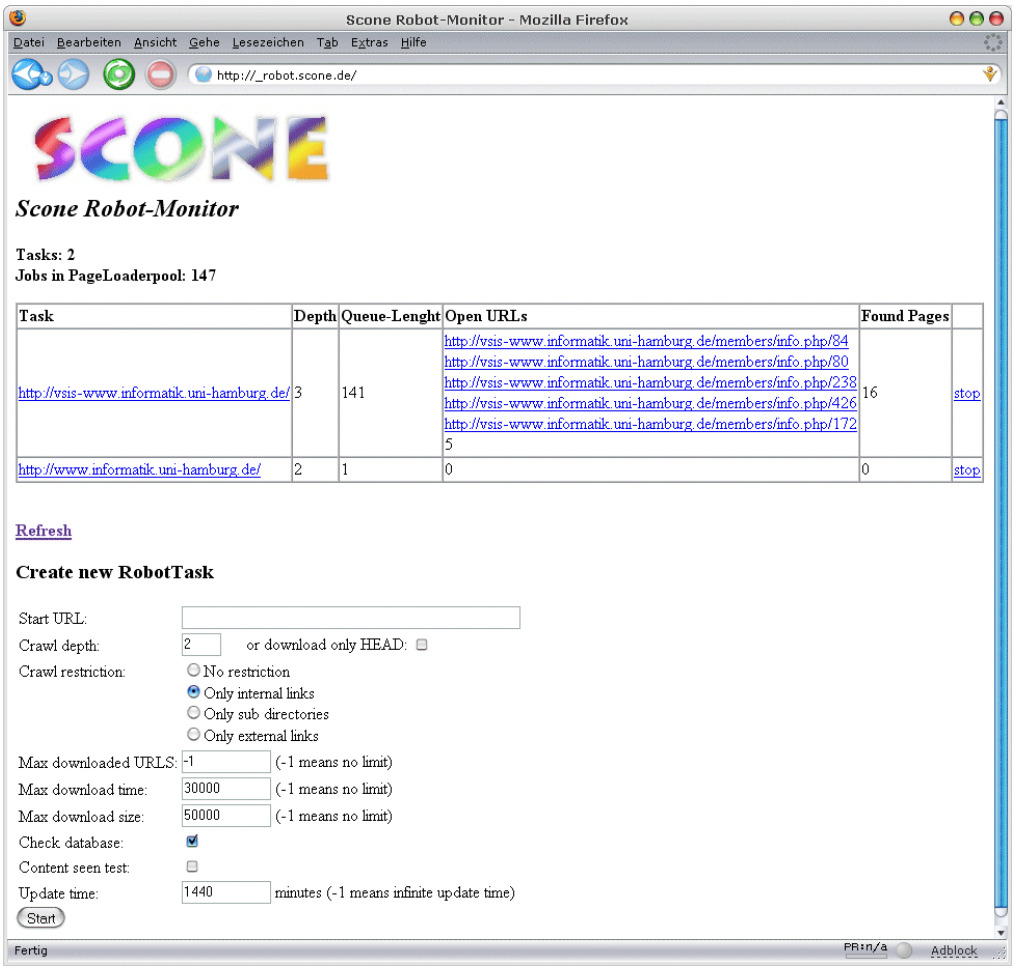


Abb. 139: Der *RobotMonitor* visualisiert die Aktivität des Crawlers und erlaubt seine Steuerung.

8.6.3 Repräsentation und Persistierung von Web-Ressourcen: *Scone NetObjects*

Die Anforderungsermittlung in Abschnitt 8.2.8 hat ergeben, dass Systeme zur Erweiterung der Orientierungs- und Navigationsmöglichkeiten im Web in der Regel eine Komponente benötigen, mit denen sich die verwendeten Ressourcen programmatisch als Objekte *repräsentieren* lassen. Darüber hinaus ist es häufig notwendig, diese Objekte zu *persistieren*, damit sie auch nach längerer Zeit zugreifbar sind.

NetObjects ist die Komponente von Scone, die diese Funktionalität bereitstellen. Sie verfügt über Klassen zur Repräsentation und Verwaltung der wichtigsten Objekttypen des Webs und bietet Mechanismen zur transparenten und performanten Persistierung dieser Objekte. Im Folgenden werden erst die technischen Anforderungen an die Komponente genauer spezifiziert (Abschnitt 8.6.3.1), dann werden die verfügbaren Klassen für Datenobjekte vorgestellt (Abschnitt 8.6.3.2). Teil 8.6.3.3 geht auf die Konzepte zur Identifikation und Adressierung dieser Objekte ein. Danach werden verschiedene Konzepte zur Persistierung der Objekte in Java verglichen, auf ihre Eignung für Scone hin untersucht und Erfahrungen mit einer objektorientierten Datenbank-Lösung zusammengefasst (Abschnitt 8.6.3.4). Teil 8.6.3.6 stellt den für die Scone NetObjects neu entwickelten objektrelationalen Datenbankadapter vor, und Abschnitt 8.6.3.7 beschreibt die entwickelten Lösungen zur Konsistenz der

Datenobjekte in Scone. Der letzte Teil 8.6.3.8 zieht ein kurzes Fazit der Eignung des objekt-relationalen Datenbank-Konzepts von Scone für große, verteilte Hypertext-Informationssysteme.

8.6.3.1 Anforderungen an Persistenzmechanismen für Web-Erweiterungen

Damit sich das Scone Framework als Basis für unterschiedliche Werkzeuge zur Erweiterung der Orientierung und Navigation im Web eignet, benötigt es eine flexible und einfache anpassbare Komponente, die die wichtigsten Objekte beim Umgang mit dem Web repräsentieren kann. Die Klassen zur Datenrepräsentation müssen einfach anzupassen sein und sollten sich gleichzeitig möglichst transparent für den Entwickler persistieren lassen.

Eine weitere Anforderung ist das verzögerungsfreie Schreiben und Wiederherstellen der Objekte: Wenn ein Benutzer beispielsweise mit mehreren Browser-Fenstern gleichzeitig arbeitet oder Webseiten mit vielen Hyperlinks öffnet, so können innerhalb von Sekunden Hunderte neuer Datenobjekte (beispielsweise *Links*) erstellt werden. Das System muss entsprechend schnell solche Datenstrukturen erzeugen, abspeichern und wieder einlesen können, ohne dass für den Anwender Wartezeiten auftreten. Eine Verlangsamung beim Browsen im Web würde die Benutzbarkeit verschlechtern und könnte die Evaluationsergebnisse eines Prototyps negativ beeinflussen.

Darüber hinaus sollte es möglich sein, komplexe Anfragen an das System zu stellen, die Attribute und Beziehungen zwischen Objekten beinhaltet. Beispielsweise kann es für einen Prototyp erforderlich sein, bestimmte Verhaltensmuster eines Anwenders auszuwerten oder die Strukturen eines Webservers zu analysieren. Die Schnittstelle zur Abfrage der Datenobjekte soll dennoch unkompliziert sein und vertretbare Anforderungen an die Entwickler stellen.

8.6.3.2 Konzepte zur Objektrepräsentation von Web-Ressourcen

Scone NetObjects bietet Klassen, um alle wichtigen Konzepte bei der Benutzung des Webs als Datenobjekte zu repräsentieren.

Eine grundlegende Kategorie von Objekten im Internet sind *alle per URI adressierbaren Ressourcen*. In den Scone NetObjects wurde hierfür die Klasse *NetNode* entworfen, die neben der Adresse der Ressource weitere elementare Attribute bietet (s. Abb. 140, Mitte oben). Hierzu gehört der *technische Typ* der Ressource (repräsentiert als *Mime-Type*), der *Domain-Name* und der Zeitpunkt der *letzten Aktualisierung* auf dem Server.[279] Hinzu kommen grundlegende technische Eigenschaften, wie der *HTTP-Status* beim letzten Zugriff und die *Dateigröße* (s. Abschnitte 4.5.4.1 und 4.5.4.2).

[279] Diese Informationen lassen sich zwar für die meisten per URI adressierbare Ressourcen angeben (beispielsweise wenn HTTP oder FTP als Protokoll genutzt wird), nicht aber für alle: E-Mail-Adressen sind ebenfalls als URI ausdrückbar, nicht verfügbare Attribute (z.B. die Größe) bleiben in solchen Fällen im NetNode-Objekt leer.

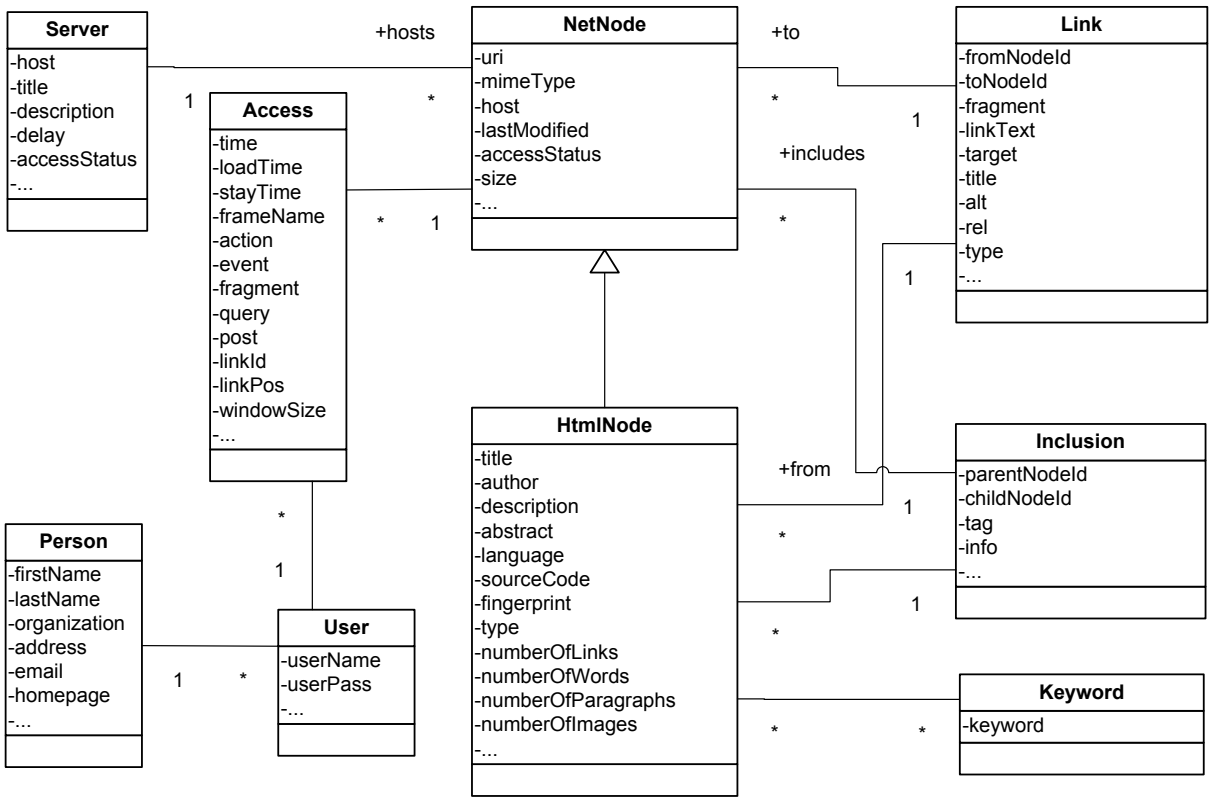


Abb. 140: Schematische Darstellung der Klassen zur Datenrepräsentation in den *NetObjects*.

Für (X)HTML-Dokumente gibt es die Unterklasse *HtmlNode*, die weitere spezifische Attribute für Web-Dokumente bietet (s. Abb. 140, Mitte unten). Dazu gehören sowohl inhaltliche Informationen wie der *Seitentitel*, der *Autor*, die *Sprache* und eine *Zusammenfassung* der Seite (s. Abschnitt 4.5.2ff), als auch zahlreiche eher technische Attribute wie ein *Fingerprint* (ein Hash-Wert, der den Seiteninhalt identifiziert, s. Abschnitt 8.6.5.3), der Quellcode der Seite sowie Informationen zur Art und Anzahl eingebetteter Objekte.

Mithilfe weiterer Klassen werden die Beziehungen zwischen den Ressourcen des Webs repräsentiert. Hierzu gehört die Klasse *Inclusion* für Objekte die in HTML-Seiten eingebunden sind, z. B. Grafiken und iFrames. Daten zu Websites werden in der Klasse *Server* zusammengefasst (s. Abb. 140). Sie berücksichtigt den Anbieter der Site, den Titel der Homepage (als „Site-Titel“) und serverspezifische Daten wie die typische Zeitverzögerung beim Zugriff auf seine Objekte (die „*Client-Perceived Response Time*“, siehe: Rajamony & Elnozahy 2001).

Zur Unterstützung von Orientierung und Navigation müssen auch die *Link-Strukturen* des Webs repräsentierbar sein. Dies geschieht durch die Klasse *Link*. Links sind gerichtete, typisierte Verweise – wie in HTML beschrieben (Raggett, Hors et al. 1999) –, aber anders als in HTML handelt es sich in Scone um *eigenständige* Objekte, die auf zwei *NetNodes* verweisen (s. Abb. 140, rechts) und somit in beide Richtungen navigierbar und auszuwerten sind (vergl. „XLink“ im Anhang C.3). *Links* bieten zahlreiche Attribute, die von den Eigenschaften der Anker-Elemente in HTML abgeleitet sind (s. Abschnitt 4.4.1). Zu den Attributen gehören die charakteristischen *HTML-Anker-Attribute* wie „`title`“, „`rel`“, „`rev`“ (s. Abschnitt 4.4.2), ein

Fragment-Identifier, der zur Adressierung von Positionen innerhalb von Dokumenten dient (Berners-Lee, Fielding et al. 2005), sowie ein „*Linktext*“, der den Inhalt des Link-Markers enthält (s. Abschnitt 2.2.1). Zudem weist Scone allen Link-Objekten Typinformationen zu, mit denen sich Links einfach filtern lassen, z. B. nach site-externen Links, oder auch strukturelle Links, die in der Struktur einer Website auf- oder abwärts führen (s. Abschnitte 4.5.1.1 und 4.5.3.3).

Eine weitere Gruppe von Klassen dient zur Repräsentation der Benutzer und ihrer Aktionen. Die entsprechenden Objekte werden durch die *AccessTracking*-Komponente von Scone erstellt (s. Abschnitt 8.6.4), die die Navigationsaktionen der Benutzer erfasst. Für einzelne Navigationsaktionen stehen die Objekte der Klasse *Access*, für die Darstellung von Benutzern dienen die Klassen *User* und *Person*[280] zur Verfügung (s. Abb. 140, links).

8.6.3.3 Herausforderungen der technischen Identifikation und Adressierung von Web-Ressourcen

Eine der Herausforderungen bei der Entwicklung der NetObjects war die schnelle und eindeutige Adressierung der Datenobjekte sowohl im Speicher als auch in der Datenbank. Im Web werden Ressourcen per URI identifiziert, in objektorientierten Sprachen werden Objekte und die Beziehungen zwischen ihnen aber mittels Referenzen (Verweise auf Speicherbereiche) hergestellt. Dieser sogenannte *Impedance Mismatch* (s. auch Abschnitt 8.6.3.4) erfordert ein entsprechendes Adressierungsschema zur Identifikation von Web-Ressourcen in den NetObjects.

Das ist komplizierter, als es erscheinen mag: So ist zwar ein naheliegendes Attribut zur Adressierung eines *NetNode* der URI, die Zuordnung ist aber oft nicht eindeutig, da sich einerseits der Inhalt vieler Seiten häufig ändert und andererseits mehrere URIs auf dieselbe Ressource verweisen können.

Scone bietet mehrere Mechanismen, um dieser Problematik Rechnung zu tragen. Zum einen kann der „Fingerprint“ einer Seite, der seinen (veränderlichen) Inhalt kennzeichnet (s. o.), zur Berechnung des Identifikators herangezogen werden. Darüber hinaus werden gleichbedeutende Adressen mittels einer „*Normalisierung*“ des URI erkannt. Hierfür wird ein für Scone entwickeltes heuristisches Verfahren verwendet,[281] das charakteristische Mechanismen von Webservern nutzt, um äquivalente URIs zu bestimmen. Dies geschieht innerhalb der Scone-Klasse *SimpleURI*,[282] die URIs nach der RFC 3986 (Berners-Lee, Fielding et al. 2005) re-

[280] In Scone können Personen mehrere User-Accounts anlegen; daher werden User und Person als getrennte Klassen repräsentiert.

[281] Ein ähnliches Verfahren wurde bereits von (Takano & Winograd 1998) beschrieben.

[282] Die in Java vorhandene Klasse „URL“ ist hierfür ungeeignet, da sie solche Verfahren nicht unterstützt und – was noch gravierender ist – sich in mehreren Punkten nicht an den RFC 3986 für URIs hält (Berners-Lee, Fielding et al. 2005).

präsentiert und sie „normalisieren“ kann. Beispielsweise werden die folgenden drei Adressen automatisch als dieselbe *SimpleURI* interpretiert:

- `http://www.scone.de/`
- `http://www.scone.de/index.html`
- `http://www.scone.de/data/../`

Eine Anpassung der Normalisierung ist durch eine Modifikation dieser Klasse zu erreichen. Je nach Bedarf lassen sich auch POST-und GET-Parameter (der „Query“-Teil eines URI, siehe: Berners-Lee, Fielding et al. 2005) berücksichtigen, da sie je nach Website ebenfalls für den Inhalt der Seite bestimmend sein können.

Ausgehend von *SimpleURI* (und gegebenenfalls weiteren Attributen) wird ein 16-Byte langer Hash-Wert berechnet,[283] der zur Identifikation von *NetNodes* im Cache und in der Datenbank dient. Diese Lösung bietet den Vorteil, dass geänderte Anforderungen an die Identifikation der Ressourcen im Web lediglich eine Anpassung des Algorithmus zur Berechnung der ID erfordern.

Für die Adressierung der anderen Objekttypen in den NetObjects werden ebenfalls Hash-Werte verwendet, die jeweils auf charakteristischen Objekteigenschaften basieren. Beispielsweise wird der Identifikator eines *Link-Objekts* aus der Start- und Zieladresse des Links berechnet. Dieses Konzept gewährleistet eine konsistente Schnittstelle zum Zugriff auf alle Datentypen in den Scone NetObjects.

Die hier aufgeführten Klassen zur Repräsentation von Ressourcen und Konzepten im Web sind in eine Struktur eingebunden, die den Zugriff auf die Datenobjekte über diese Identifikatoren erlaubt und sie (wenn gewünscht) automatisch persistiert. Diese Architektur wird im übernächsten Abschnitt 8.6.3.6 beschrieben.

8.6.3.4 Erfahrungen mit unterschiedlichen Konzepten zur Persistierung von Web-Ressourcen in Java

Um die Anforderungen bezüglich der Persistierbarkeit der Datenobjekte in den Scone NetObjects zu erfüllen (s. Abschnitt 8.6.3.1), wurden zahlreiche Konzepte zur Speicherung von Daten in Java untersucht und zum Teil auch praktisch erprobt. Im Folgenden wird ein kurzer Überblick über die in Erwägung gezogenen Konzepte gegeben, und es wird dargelegt, welche Vor- und Nachteile die Techniken aufweisen. Da es zahlreiche „Standards“ zur Speicherung von Objekten in Datenbanken für die Sprache Java gibt, wurden nur die wichtigsten relevanten Techniken berücksichtigt.

---

[283] Das in Scone verwendete Verfahren erstellt ähnlich dem MD5-Hash-Algorithmus (*Message Digest Algorithm)* einen vom Ausgangsstring ausgehenden, möglichst eindeutigen Wert fester Länge. Es wurde ein im Vergleich zu MD5 schnellerer Algorithmus verwendet, da es nicht auf die Stärke der Verschlüsselung ankommt, sondern lediglich auf die Vermeidung von *zufälligen* Kollisionen (siehe z. B. Menezes, Oorschot et al. 1996).

Ein zentraler Standard zur Persistierung von Daten mit Java ist *JDBC (**J**ava **D**atab**b**ase **C**onnectivity)*. Er normt die API zum Zugriff auf relationale Datenbanksysteme (Reese 2000). Der Entwickler muss dabei die konkrete Abbildung der Daten auf die Datenbank selbst realisieren, JDBC ist für ihn somit keinesfalls transparent. Die Implementation ist darüber hinaus vom eingesetzten Datenbank-Managementsystem abhängig. In Bezug auf die Objektorientiertierung von Java stellt sich das Problem des *Impedance Mismatch* (Date 2003): Hierzu gehört, dass die Objekte im Programm über ihre Adresse identifiziert werden, die Daten innerhalb der Objekte gekapselt sind, sie häufig komplexe, zusammengesetzte Strukturen aufweisen und zumeist über eine eigene Funktionalität verfügen. Dagegen werden die Datensätze in relationalen Datenbanken über ihre Attribute identifiziert und man operiert auf Mengen von normalisierten Tupeln (siehe auch Abschnitt 8.6.3.3). Änderungen an der Struktur der Datenobjekte erfordern daher meist auch Anpassungen der Datenbank-Schnittstelle.

Ein verbreiteter Standard zur *transparenten* Speicherung von Daten in Java, der im Zusammenhang mit J2EE[284] entwickelt wurde, sind Enterprise *Java Beans (EJB)* mit *Container Managed Persistence (CMP)*. Diese API erlaubt es, Komponenten ohne spezielle Anpassung an den darunter liegenden Persistenzmechanismus abzuspeichern, indem der entsprechende Container die Persistierung vornimmt. Anwendungsserver (*Application Server*) auf Basis von J2EE wenden diesen Mechanismus häufig im Rahmen ihrer mehrschichtigen Architektur an.[285] EJB CMP ist für die Nutzung in Scone kaum geeignet, da die entsprechende mehrschichtige Enterprise-Architektur – die durchaus als Komplex bezeichnet werden kann – einen in diesem Rahmen kaum zu rechtfertigenden Overhead darstellt. Ziel bei Scone war es, ein überschaubares, einfach zu erlernendes und leichtgewichtiges Framework anzubieten, das sich auch durch Endanwender auf Client-Systemen installieren lässt und nicht die aufwendige Installation eines Anwendungsservers voraussetzt. Zudem sind die Anforderungen an Scone bezüglich der Performanz – insbesondere beim Speichern von Objekten – mit EJB CMP auf normalen Arbeitsplatzrechnern kaum zu erreichen.

Ein weiterer Standard zur transparenten Speicherung von Daten in Java ist das 2002 von mehreren Firmen verabschiedete *JDO* (*Java Data Objects*). JDO ist ein Framework für die hersteller- und systemunabhängige Persistierung von Java-Objekten, wobei die Spezifikation des Mappings der Objekte auf das Datenbank-System in XML-Metadateien definiert wird. Es gibt zahlreiche Implementationen von JDO, die zur Speicherung der Daten relationale oder objektorientierte Datenbanken verwenden, beispielsweise *DataNucleus*.[286] JDO wurde für den

---

[284] *J2EE*, die *Java 2 Enterprise Edition*, bezeichnet die Spezifikation eines Architekturstandards für Softwarekomponenten und Dienste. Er dient als Rahmen für modulare, verteilte und mehrschichtige Anwendungssysteme in Java.

[285] Im Rahmen von *EJBs* kann auch *Bean Managed Persistence* (*BMP*) eingesetzt werden. Bei diesem Modell muss sich allerdings die „*Java Bean*" selbst um das Abspeichern der Attribute kümmern.

[286] *DataNucleus* ist ein Open-Source-Projekt und gilt als Referenz-Implementation von JDO. Mehr Informationen findet man unter: http://www.datanucleus.org/.

Einsatz innerhalb von mehrschichtigen 2-Tier- oder 3-Tier-Architekturen entworfen und wird ebenfalls für J2EE genutzt. Es solle einige der Defizite von EJB vermeiden (Schwerfälligkeit, potenzielle Dateninkonsistenzen bei parallelem schreibenden Zugriff, zu hohe Systemabhängigkeit (Reese 2003). JDO genügt schon eher den Anforderungen von Scone: Es ist relativ leichtgewichtig, bietet mit JDOQL eine eigene Anfragesprache und bietet dem Entwickler eine gewisse Transparenz bei der Datenpersistierung. Allerdings gilt es immer noch als relativ kompliziert in der Programmierung (s. Jordan & Russell 2005).

Das Open-Source-Framework *Hibernate* nimmt das Mapping von Objekten zur relationalen Datenbank mithilfe von XML-Spezifikationsdateien oder Java Annotationen vor. Das Datenbankschema wird dabei automatisch vom System verwaltet, Anfragen werden über eine eigene Anfragesprache HQL gestellt. Obgleich *Hibernate* für den Programmierer als komfortabel gilt, ist es ebenfalls für Anwendungsserver vorgesehen und entsprechend komplex.

Die *Java Persistence API* (auch *JPA* genannt) ist ein offizieller Java-Standard, der die Kritik an den obigen Standards aufgreift und für die schnelle relationale Persistenz relativ einfacher Objekte entwickelt wurde. Es ist erstmals im Mai 2006 veröffentlicht worden und konnte somit für Scone nicht mehr berücksichtigt werden. Den gestellten Anforderungen kommt es insgesamt am nächsten.

Eine anderer Ansatz zur Persistierung von Objekten sind *objektorientierte Datenbanken (OODBs)*. Sie bieten sich zur Speicherung von Daten in objektorientierten Sprachen an, da sie dasselbe Paradigma verfolgen und Objekte ohne zusätzliche Abbildung direkt persistieren. Das Angebot an objektorientierten Datenbank-Managementsystemen für Java ist im Vergleich zu den relationalen und objektrelationalen Datenbank-Systemen überschaubar. Zu Beginn des Scone-Projekts waren unter anderem *Objectivity/DB*,[287] *ObjectStore*[288] und *Poet FastObjects*[289] verfügbar.

Da eine objektorientierte Datenbank ohne eine Abbildung auf ein objektrelationales Schema auskommt, wurde für eine der ersten Versionen von Scone ein Persistenzmechanismus auf Basis von *Poet FastObjects* entwickelt. Die OODB galt zu der Zeit als schnellstes der verfügbaren Systeme (Brumen, Domajnko et al. 1999), und die Implementation erschien einfach, zumal der normale Java-Compiler weiterverwendet werden konnte[290] und das System die Abfragesprache OQL unterstützte.[291]

---

[287] Das System wird von der Firma *Objectivity Inc.* angeboten: http://www.objectivity.com/solutions/.

[288] *ObjectStore* wird von der Firma *Progress Software Corporation* angeboten: http://www.progress.com/objectstore/.

[289] Die Firma *Poet* ist inzwischen mit der Firma *Versant* fusioniert, und das System wird nun als *Versant FastObjects* angeboten: http://www.versant.com/products/FastObjects.aspx.

[290] Klassen werden gemäß dem Standard der *ODMG 3.0* (Cattell, Barry et al. 2000) von „außen" in einer Metadatei als persistenzfähig deklariert und dann durch einen Post-Compiler für die Verwendung mit dem Datenbanksystem angepasst.

[291] OQL ist eine SQL-ähnliche Anfragesprache, die ebenfalls von der ODMG standardisiert wurde (Cattell, Barry et al. 2000).

8.6.3.5 Erfahrungen mit OODBs zur Persistierung von Web-Ressourcen

Im Zusammenspiel mit den Konzepten des objektorientierten Datenbanksystems traten allerdings einige grundlegende Probleme auf, die voraussichtlich auch für andere Projekte im Bereich großer verteilter Informationssysteme relevant sind: Der Zugriff auf Objekte in der Datenbank geschieht bei Poet FastObjects *immer* innerhalb von Transaktionen. Erst nach dem Öffnen einer Transaktion ist der Zugriff auf persistierbare Objekte freigegeben, und das Schließen der Transaktion führt automatisch zum Abspeichern der Objekte. Im Zusammenhang mit dem Intermediary und dem Robot von Scone stellte die *zwingende* Verwendung von Transaktionen in FastObjects eine Einschränkung dar, weil alle entsprechenden Datenobjekte auch nur innerhalb von Transaktionen nutzbar waren (POET 2000). Da in einem großen verteilten System wie dem Web oft nicht absehbar ist, wie lange einzelne Operationen dauern, also *wie lange* die Übertragung einer Ressource in Anspruch nimmt und *ob* der Transfer überhaupt korrekt abgeschlossen wird, dauerten einzelne Transaktionen häufig sehr lange. Dabei wurden aber alle beteiligten Objekte gesperrt, sodass Threads,[292] die parallel auf dieselben Datenobjekte zugriffen, blockiert wurden.

Verschärft wurde dieses Problem durch das Konzept *transitiv persistenter Objekte*, dads dazu führt, dass *alle referenzierten* Objekte innerhalb von Transaktionen ebenfalls gesperrt wurden. Für das Web als Hypertext-System bedeutet dies aufgrund der Vernetzung durch die Links, dass der Zugriff auf *eine* Webseite gleichzeitig alle weiteren Ressourcen blockiert, sofern sie (auch entfernt) über Hyperlinks mit ihr verknüpft sind.[293]

Der stetig wachsende Hypertext-Graph hatte überdies zur Folge, dass Datenbankzugriffe immer länger dauerten. Insbesondere Schreiboperationen führten bereits bei einigen Tausend Objekten zu blockierenden[294] Zeitverzögerungen von mehreren Sekunden. Hierfür ließ sich trotz diverser Optimierungsversuche leider keine praktikable Lösung finden. Als Fazit scheinen objektorientierte Datenbanken für die Persistierung von Dokumenten in umfangreich verknüpften Informationsmengen bei zeitkritischen Anwendungen Defizite aufzuweisen, sofern häufig Schreiboperationen auftreten.[295] Daher musste die Implementation der Scone NetObjects mit *Poet FastObjects* aufgegeben werden. Diese Erfahrungen halfen aber bei der Entwicklung eines objektrelationalen Datenbankadapters für Scone, der im Folgenden vorgestellt wird.

---

[292] Eine Nebenläufigkeit beim Browsen im Web kann bereits auftreten, wenn ein Benutzer mehrere Browser-Fenster oder Tabs verwendet oder „Reload“ aufruft, bevor die Seite vollständig übertragen wurde.

[293] Dieses Problem wurde für Scone mithilfe des Konzepts der sog. Schattenobjekte (gemäß "Shadow Objects" in (Xiao 1995: S. 77ff; Bacon 1998) als transiente Kopien der Datenbank-Objekte gelöst, die die Dauer von Datenbank-Transaktionen minimierten (Stephan 2002: S. 31 ff).

[294] Diese Schreiboperationen nahmen im Datenbank-Managementsystem derart viel Rechenzeit in Anspruch, dass währenddessen sogar die Arbeit mit dem Browser bzw. Computer insgesamt stockte.

[295] Solche Probleme zeigten sich bereits vor vielen Jahren bei Netzwerk-Datenbanken, die eine gewisse Verwandtschaft mit OODBs aufwiesen, da sie direkte Referenzen zwischen den Datensätzen verwendeten.

8.6.3.6 Ein objektrelationaler Datenbankadapter zur leichtgewichtigen Datenpersistierung

Insbesondere die Komplexität und mangelnde Performanz existierender Techniken zur objektrelationalen Persistierung von Daten in Java war Motivation für die Entwicklung einer neuen Datenbank-Schnittstelle für Scone. Sie sollte wesentlich leichtgewichtiger sein als die erwähnten Standards EJB CMP und JDO und dennoch für den Entwickler eine möglichst hohe Transparenz und einfache Anpassbarkeit bieten.

Der in Scone realisierte Mechanismus verwendet zur Verwaltung der Datenobjekte spezielle *Container-Klassen*. Jeder Objekttyp verfügt über einen eigenen Container, der die Zugriffe auf die Datenbank kapselt, ein automatisches *Caching* der Daten durchführt und die Konsistenz zwischen Primärspeicher und Datenbank sicherstellt. Als Datenbanksystem wurde die etablierte Open-Source-Software MySQL[296] gewählt, da sie frei erhältlich ist und eine sehr gute Performanz aufweist.[297]

Abb. 141 skizziert die Architektur des objektrelationalen Persistenzkonzepts der *Scone NetObjects* am Beispiel des Datenobjekts *NetNode* (s. Abb. 141, Bildmitte unten, vergleiche auch mit Abschnitt 8.6.3.2 und Abb. 140, Bildmitte oben). Die Container-Klassen heißen wie die Klasse ihres Datenobjekts mit dem Zusatz *Cache*[298] (z. B. *NetNodeCache* für die Klasse der *NetNodes*), da die Container die Objekte im Primärspeicher zwischenspeichern („Cachen“) können. Der Entwickler muss für jeden Datentyp jeweils nur mit den beiden farbig hinterlegten Klassen vertraut sein: der Objektklasse (*NetNode*) und seiner Container-Klasse (*NetNodeCache*); alle weiteren Datentypen nutzen dasselbe Prinzip.

---

[296] MySQL ist das quelloffene relationale Datenbanksystem der Firma *Oracle*: http://www.mysql.com/.

[297] Ein unabhängiger Vergleich der Gesamtperformanz von Datenbanken ist der *Barcelona Circuit* des *PolePosition-Projekts:* http://www.polepos.org/.

[298] Die Persistierung in der Datenbank ist innerhalb der Cache-Klassen soweit abstrahiert, dass sich die *Scone NetObjects* auch ohne Datenbank-Managementsystem verwenden lassen, wobei die Objekte lediglich im Cache gehalten werden. Für viele Prototypen erwies sich dies als ausreichend. Die Verwendung des DBMS lässt sich in der grafischen Schnittstelle von Scone konfigurieren (s. Abschnitt 8.6.6).

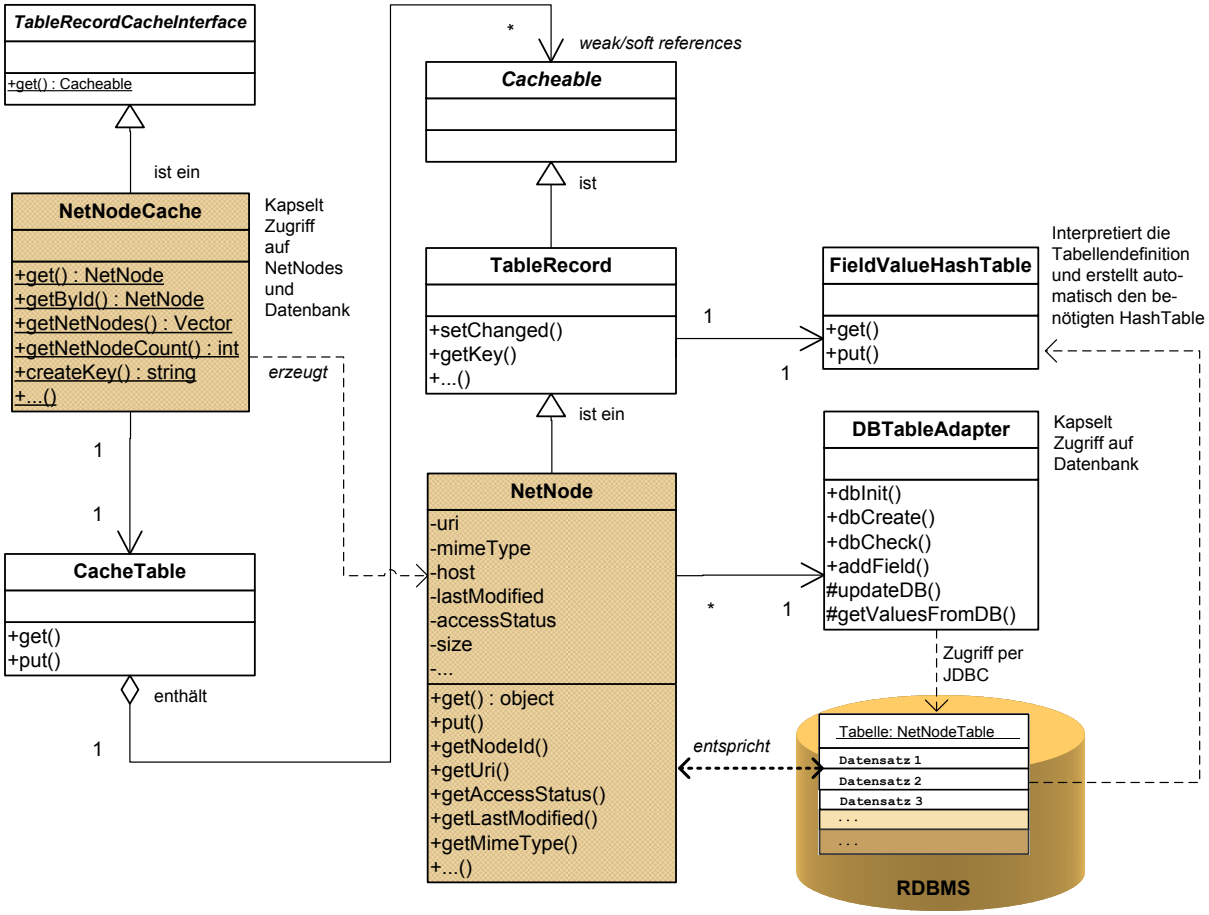


Abb. 141: Struktur des Scone Datenbankadapters, gezeigt am Beispiel der *NetNodes*.

Von jeder Container-Klasse gibt es gemäß dem *Singleton-Pattern* (Gamma, Helm et al. 1994) nur eine Instanz; sie sichert die Synchronisation des Zugriffs auf den Cache und die Datenbank. Jeder Container bietet *Factory-Methoden* (Gamma, Helm et al. 1994) zum Bereitstellen der Datenobjekte. Zur Adressierung der Objekte dient als Identifikator eine 16-Byte lange Hexadezimalzahl. Dieser Wert wird aus kennzeichnenden Attributen des jeweiligen Objekttyps mittels einer Hash-Funktion berechnet (s. Abschnitt 8.6.3.3). Die Methode zur Berechnung des Identifikators (*createKey*) wird von der Container-Klasse bereitgestellt (s. Abb. 141, links).

Je nach Typ des Objekts können zum Zugriff auch andere Attribute verwendet werden, z. B. lassen sich *NetNodes* sowohl über den Identifikator „*NodeID*“ als auch über die Adresse der Ressource (den URI) zugreifen (s. Abb. 141, *NetNodeCache*). Nach Aufruf der Factory-Methode wird als Erstes kontrolliert, ob das Objekt noch im Primärspeicher – also im *CacheTable* (s. Abb. 141, links unten) des Containers – vorhanden ist. Ist es im Cache, wird das Objekt direkt zurückgegeben, sonst wird ein neues Datenobjekt erstellt, und es wird ihm „mitgeteilt“, dass es in der Datenbank nach einen bereits vorhandenen Eintrag suchen soll. Dafür ruft es den *DBTableAdapter* auf, der per JDBC auf das Datenbanksystem zugreift. Wird ein Eintrag gefunden, werden die Werte aus der Datenbank übernommen, andernfalls liefert die Methode ein initialisiertes Objekt zurück.

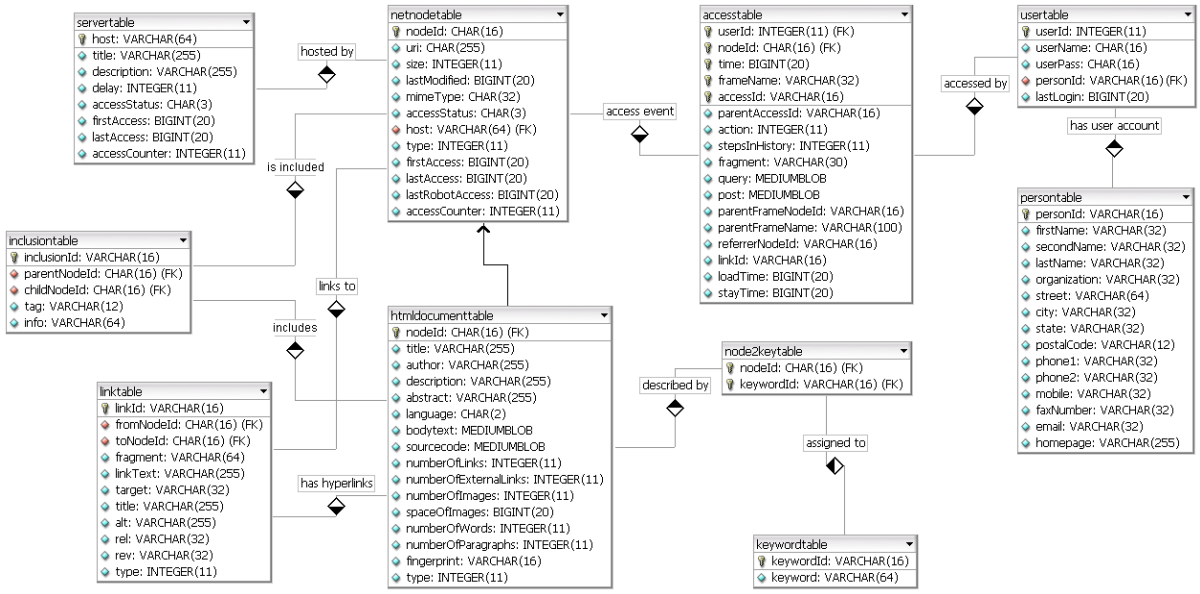


Abb. 142: EER-Modell Version 1.2 der Scone-Datenbankstruktur.

Die Datenobjekte der NetObjects (wie NetNode, HtmlNode und Link) verwenden Hash-Tabellen zur Speicherung der *Attribute*. Dies geschieht mithilfe des *FieldValueHashTable*, der bei der Erstellung eines Datenobjekts automatisch initialisiert wird, indem sein Konstruktor die Attributnamen und -typen aus der Strukturdefinition der zugehörigen Tabelle ausliest.[299] Es ist daher lediglich notwendig, die Tabellendefinition eines Datenobjekts anzupassen, wenn die Attribute verändert werden sollen. Die Datenbanktabellen (s. Abb. 142) sind nach den Klassen der NetObjects mit dem Zusatz „Table" benannt (z. B. *NetNodeTable* für die Klasse der *NetNodes*, vergleiche Abb. 140); die Tabellenfelder entsprechen den Objektattributen.

Jede Container-Klasse kann darüber hinaus nach dem *Observable-Pattern* (Gamma, Helm et al. 1994) Nachrichten an angemeldete Objekte senden, wenn ein Datenobjekt erstellt wird, es geändert wird oder wenn ein lesender Zugriff erfolgt. Auf diese Weise lassen sich Plug-ins über Ereignisse informieren, beispielsweise wenn der Benutzer eine neue Seite aufruft (s. Abschnitt 8.6.4) oder ein Dokument durch den Robot bereitgestellt wurde (s. Abschnitt 8.6.2.3).

Der Persistenzmechanismus bietet zusätzlich eine Schnittstelle, über die auch strukturierte Anfragen an die Datenbank möglich sind und die entsprechende Objektstrukturen zurückliefert. So lassen sich beispielsweise alle Link-Objekte mit einer bestimmten Eigenschaft abrufen. Zu diesem Zweck wurden *Datenbank-Wrapper* in den Container-Klassen realisiert, die solche strukturierten Anfragen verarbeiten und einen Iterator mit den gewünschten Objekten zurückliefern. Der Wrapper stellt zudem sicher, dass vor jeder Datenbankanfrage

[299] Konkret werden diese Daten nicht aus der Datenbank selbst entnommen, sondern der SQL-Definitions-Datei, mittels derer die Datenbank erstellt wurde. Zur Interpretation dieser Schema-Datei wurde ein eigener Parser entwickelt, der das Mapping der Feldtypen der Datenbank auf die entsprechenden Variablentypen in Java vornimmt. Der Vorteil dieser Lösung liegt wiederum darin, dass die Komponente so auch ohne Verbindung zur Datenbank verwendet werden kann.

alle Änderungen der beteiligten Caches mit der Datenbank synchronisiert werden. Der Wrapper-Methode wird in der Anfrage mitgeteilt, welche Objekttypen beteiligt sind und welche Bedingungen sie erfüllen sollen. Die Bedingung entspricht dabei dem WHERE-Ausdruck einer SQL-Anfrage. Aus den Parametern wird ein SQL-Ausdruck generiert, die Datenbankausgabe wird interpretiert und als Iterator mit den entsprechenden Objekten zurückgegeben. So liefert die Methode *getNetNodes()* des *NetNodeCache* eine Menge von *NetNodes* zurück, die die angegebene Bedingung erfüllen; *getNetNodeCount()* ermittelt die Anzahl der Objekte, die ein Kriterium erfüllen (s. Abb. 141).

8.6.3.7 Caching und Konsistenzsicherung von Daten in Web-Erweiterungen

Das Zwischenspeichern der Daten in den *CacheTables* der Container beschleunigt den lesenden Zugriff auf die Datenobjekte erheblich, da beim Browsen im Web häufig wiederholt auf dieselben Ressourcen innerhalb kurzer Zeit zugegriffen wird. Um das Speichern der Daten ebenfalls zu optimieren, wurde ein Mechanismus entwickelt, der das Einfügen und Aktualisieren der Objekte in die Datenbank (sofern möglich) verzögert, bis der Computer über freie Ressourcen verfügt. Dies leistet ein mit niedriger Priorität laufender Thread, der für jeden *CacheTable* das regelmäßige Übertragen der Datenobjekte aus dem Speicher in die Datenbank durchführt.

Die *CacheTables* der Container sind als dynamisch erweiterbare Hash-Tabellen realisiert und können theoretisch eine unbegrenzte Menge von Objekten zwischenspeichern, praktisch ist dies jedoch durch den verfügbaren Primärspeicher des Computers begrenzt. Aus diesem Grund werden die Referenztypen „`java.lang.ref.WeakReference`“ und „`java.lang.ref.SoftReference`“ eingesetzt, die dafür sorgen, dass bereits persistierte und aktuell nicht benutzte Objekte bei Bedarf automatisch[300] vom *Garbage Collector* aus dem Hauptspeicher entfernt werden. Noch nicht persistierte Objekte werden im *CacheTable* durch zusätzliche „harte“ Referenzen vor dem automatischen Löschen gesichert.

Aufgrund der Erfahrungen mit der objektorientierten Datenbankanbindung (s. Abschnitt 8.6.3.5) wurde bei der Implementation des objektrelationalen Datenbankadapters auf Datenbanktransaktionen verzichtet. Inkonsistenzen werden stattdessen mithilfe der zentral über die Container gesteuerten Datenbankzugriffe vermieden, beispielsweise indem zusammengehörige Objekte[301] gemeinsam persistiert werden. Dennoch lässt sich für ein intermediarybasiertes System wie Scone nur ein Zustand *schwacher Konsistenz* (*weak consistency*) sicherstellen, da *das Web selbst* keine Mechanismen zur Gewährleistung der Konsistenz zwischen

[300] Werden Objekte in Java lediglich über eine dieser Klassen referenziert, so kann der Garbage Collector diese Objekte aus dem Hauptspeicher löschen und den entsprechenden Speicher wieder freigeben. Die beiden Klassen weisen unterschiedliche Strategien auf, nach denen die Objekte aus dem Speicher entfernt werden.

[301] Beispielsweise werden zusammen mit einem Link-Objekt auch immer die beiden NetNodes persistiert, auf die dieses Link-Objekt verweist. So werden Fehler bei Fremdschlüsseln vermieden.

der lokalen Kopie einer Ressource und dem Original bereitstellt:[302] Webseiten können geändert werden, ohne dass andere Sites oder Systeme wie Scone darüber informiert werden, was unvermeidbar zu (temporären) Inkonsistenzen führt. Dies ist allerdings eine grundsätzliche Schwäche des Webs, die sich nicht nur auf die hier eingesetzten Caching-Mechanismen bezieht (vergl. auch Cao & Liu 1998), sondern z. B. auch für Hyperlinks in Webseiten und Suchmaschineneinträge zutrifft (s. Abschnitt 3.3.4). Die mangelnde Konsistenzsicherung des Webs war laut Tim Berners-Lee eine bewusste Designentscheidung, die eine ausreichende Skalierbarkeit sicherstellt.[303] Sie basiert auf der Erkenntnis, dass der Grad an Konsistenz in großen verteilten Systemen[304] immer gegen andere Anforderungen wie Performanz, Komplexität und Skalierbarkeit abgewägt werden muss (Golding 1992).

8.6.3.8 Potenziale und Grenzen der Scone NetObjects für große verteilte Hypertext-Informationssysteme

Der für Scone entwickelte objektrelationale Datenbankadapter hat sich bewährt: Die Kapselung des Zugriffes auf die Objekte über eigene Container weist den Vorteil einer hohen Performanz bei einfacher Anpassbarkeit des Codes auf. Durch das Caching und das verzögerte Schreiben der Objekte treten für Benutzer in der Regel nicht einmal beim parallelen Abruf mehrerer Dokumente störende Verzögerungen auf.

Die Repräsentation der Hypertext-Strukturen des Webs stellte sich als weniger problematisch heraus, als es vom Autor dieser Arbeit erwartet wurde und in der Literatur für andere Hypertext-Systeme und Standards beschrieben wird (s. beispielsweise Specht 1997): Hypertext-Standards wie das Dexter Modell (s. Anhang C.1) oder XLink (s. Anhang C.3) verwenden strukturierte Datentypen für Hyperlinks, und es gilt daher als Herausforderung, sie auf relationale Datenbanken abzubilden. Durch die „Einfachheit" des Webs stellte sich diese Schwierigkeit nicht: HTML-Links sind in der Regel lediglich binäre Verknüpfungen, bei denen sowohl Start und Ziel durch URIs – also Attribute – dargestellt werden.

8.6.4 Erfassung von Benutzeraktionen im Web: *Scone AccessTracking*

Das Scone *AccessTracking* dient zur Erfassung von Benutzeraktionen mit ihrem Webbrowser. Die Funktionalität dieser Komponente wird im Anwendungskontext von Scone im Wesent-

---

[302] Klassisch wird zwischen (mindestens) zwei Graden von Konsistenz unterschieden: Nach dem traditionellen ACID-Prinzip müssen bei *starker Konsistenz* (strong consistency) sämtliche Daten des Systems immer in einem übergreifend synchron konsistenten Zustand sein. Dies ist aber bei verteilten Systemen mit unzuverlässiger oder vergleichsweise langsamer Datenübertragung oft nicht zu erreichen, weshalb man unterschiedliche Grade von schwacher Konsistenz (weak consistency) definiert hat (Adya 1999) und diese Konzepte schon seit Jahrzehnten bei Systemen wie USENET einsetzt (Kantor & Lapsley 1986).

[303] Dies äußerte *Tim Berners-Lee* in seiner Keynote zum *Semantic Web Project* bei der 9. WWW-Konferenz im Jahr 2000 in Amsterdam.

[304] Probleme mit der Konsistenz in großen verteilten Systemen wurden bereits früh erkannt und sogar im SQL92-Standard (bzw. ANSI/ISO X3.135-1992) berücksichtigt, der daher unterschiedliche Grade der Isolation in Transaktionen anbietet (z. B. „dirty read"), um Deadlocks und Performanzprobleme zu vermeiden.

lichen für zwei Zwecke benötigt: Erstens sollen Scone Plug-ins auf die Navigationsaktionen eines Anwenders reagieren können (s. Abschnitt 8.2.4), und zweitens soll es möglich sein, die Benutzung des Webs präzise zu protokollieren, um später diese Daten auszuwerten (s. Abschnitt 8.2.7).

Bei „normalen" Web Proxies lässt sich der Pfad eines Anwenders – der *Clickstream* – aus der Reihenfolge der übertragenen Objekte herleiten. Eine solche Datenbasis ist aber für Navigationswerkzeuge häufig nicht ausreichend, weil viele Details der Benutzerinteraktion verborgen bleiben. Beispielsweise ist oft entscheidend, *wie* ein Benutzer auf eine bestimmte Seite gelangt ist, also welche Daten er in ein *Formular* eingegeben hat, welchen *Link* er auswählte oder von welchen *Bedienelementen* des Browsers er Gebrauch machte. Diese Informationen sind in normalen Proxy-Protokollen nur ansatzweise oder gar nicht zu finden. Zudem sollen zeitnah Nachrichten über Benutzeraktionen an angemeldete Komponenten gesendet werden, damit die mit Scone entwickelten Erweiterungen unmittelbar auf die Handlungen des Anwenders reagieren können. Darüber hinaus müssen die erfassten Daten schnell zugreifbar und auszuwerten sein, Benutzer müssen sich eindeutig unterscheiden lassen, und technische Probleme, wie das Caching von Seiten im Browser, dürfen nicht dazu führen, dass Navigationsaktionen „verpasst" werden. Das Scone *AccessTracking* bindet JavaScript-Code in Webseiten ein und greift auf zahlreiche Protokoll- und Browser-Parameter zurück, um die aufgeführten Ziele zu erreichen.

8.6.4.1 Möglichkeiten zur Erfassung von Benutzeraktionen mit Intermediaries

Die Auswertung von Benutzeraktionen aus den Zugriffsprotokollen (*Logs*) von Web-Proxies und Servern wurde bereits umfangreich untersucht, und zahlreiche Systeme stehen zur Verfügung, um diese Daten für spezifische Zwecke aufzubereiten und zu analysieren (Pitkow 1998; Spiliopoulou, Mobasher et al. 2003). Die Möglichkeiten dieser Techniken leiden aber immer unter den Beschränkungen der Ausgangsdaten: Benutzer lassen sich nicht eindeutig identifizieren, mehrere Browser-Fenster werden nicht unterschieden, und einige Navigationsaktionen wie das Backtracking[305] sind kaum nachvollziehbar (Etgen & Cantor 1999). Die „lokale" Interaktion der Anwender mit dem Browser, wie Scrolling, Anklicken bestimmter Elemente mit der Maus oder Änderungen der Fenstergröße, bleibt gänzlich verborgen.

Da die Anforderungen des *AccessTrackings* von Scone mit den „normalen" Zugriffsprotokollen von Web-Proxies nicht zu erreichen waren, wurde ein neues Konzept entwickelt, das die benötigte Funktionalität auf Basis eines Intermediary realisiert. Dabei wird vom Intermediary in alle übertragenen *Webseiten* zusätzlicher JavaScript Programmcode integriert, der unter anderem zusätzliche Parameter aus dem Browser ausliest und an Scone sendet.

[305] Backtracking führt zumeist dazu, dass die Seiten aus dem Cache des Browsers geladen werden. Es werden dann keine neuen Einträge im Proxy- bzw. Server-Log erzeugt.

Die Idee, Webseiten zum Zwecke der Benutzungsanalyse funktional zu erweitern, ist nicht gänzlich neu: Analysesysteme für die Nutzung von Websites wie *Google Analytics*[306] umgehen die Einschränkungen gewöhnlicher Zugriffsprotokolle, indem sie *JavaScript* und *Cookies* einsetzen. Zu diesem Zweck muss aber vom Autor der Site ein Script in sämtliche Seiten eingebunden werden. Es dient zur Identifikation der Benutzer und liest einige zahlreiche System- und Browserparameter aus. Das Scone AccessTracking verwendet eine ähnliche Technik, ermittelt aber keine statistischen Daten des Zugriffs auf einzelne Sites, sondern erfasst die Interaktion eines Anwenders mit dem Browser für das gesamte Web.

Es gibt einige Forschungsprojekte, die technisch mit dem Scone AccessTracking verwandt sind, teilweise als Inspiration dienten oder parallel zu Scone entwickelt wurden. Sie können dem Bereich der „*Remote Usability Evaluation*“ zugeordnet werden, bei denen Testleiter und Probanden während eines Benutzbarkeitstests räumlich und/oder zeitlich voneinander getrennt sind (Hartson, Castillo et al. 1996):

- *WebVIP* war eines der ersten Werkzeuge, das die Benutzeraktionen mit dem Browser zum Zwecke der Benutzbarkeits-Evaluation von Websites detailliert aufzeichnete (Scholtz & Laskowski 1998). Das System verwendet zusätzliche JavaScript-Events, die auf das Anklicken von Links reagieren und so den genauen Pfad des Teilnehmers registrieren. Er hat zudem die Möglichkeit, Ergebnisse von Aufgaben und Feedback in ein Web-Formular einzugeben. Allerdings muss die zu evaluierende Website komplett *kopiert* werden, damit WebWIP den JavaScript-Code in alle Seiten einfügen kann. Für interaktive Websites lässt sich WebVIP daher kaum einsetzen.
- Für das Evaluationssystem *WET* sind ebenfalls die Web-Dokumente anzupassen. Der Autor muss wenige Zeilen JavaScript in jede Seite einbinden, damit das System registrieren kann, über welchen Elementen der Seite sich der Mauszeiger befindet, was angeklickt wurde und ob der Anwender etwas in ein Web-Formular eingegeben hat. Die so erfassten Daten werden mithilfe von Cookies zum *WET-Server* übertragen (Etgen & Cantor 1999).
- *WebQuilt* ist ein serverseitiger Intermediary in Java, der die Analyse beliebiger Websites unterstützt, ohne dass schreibender Zugriff auf die Ressourcen des Servers benötigt wird. Alle Browser-Anfragen gehen über den WebQuilt-Intermediary, der sie an die zu evaluierenden Site weiterleitet. Die Links in den übertragenen Dokumenten werden modifiziert, damit weitere Anfragen ebenfalls über den WebQuilt-Intermediary laufen (s. auch Abschnitt 8.6.5.6; Hong, Heer et al. 2001; Lorenzen-Schmidt & Nufer 2008).
- Ein Framework des *IBM Austin Research Labs* verwendet eine ähnliche Intermediary-Technik, um die bei Benutzern des Webs tatsächlich auftretenden *Latenz- und Ladezeiten* aufzuzeichnen. Diese Erweiterung für Webserver fügt automatisch einen als „*Time Keeper*“ bezeichneten JavaScript-Code in alle Seiten ein, der die Ladezeit registriert und

[306] Der Dienst „Google Analytics“ wird unter folgender Adresse angeboten: http://www.google.com/analytics/.

mithilfe von speziellen HTTP-Requests an einen Protokollserver übermittelt (Rajamony & Elnozahy 2001).

- Der *UsaProxy* ist ein weiterer Intermediary, der bei der Übertragung JavaScript-Code in alle Webseiten einbindet, um die genaue Interaktion des Anwenders mit den Elementen der Seite festzustellen. Der Code kann die Pfade des Mauszeigers aufzeichnen, registrieren wie ein Benutzer Formulare ausfüllt und welche Elemente er anklickt. Ein Ziel dieses Systems ist die entfernte *Evaluation von Ajax-Systemen*, also interaktiven Webseiten, bei denen Navigation nicht mehr unbedingt das Laden einer kompletten neuen Seite bedeutet (Atterer, Wnuk et al. 2006).

Scone vereint die Potenziale dieser fünf Projekte im *AccessTracking*: Es werden detaillierte Daten über das Benutzerverhalten erfasst und Browser-Parameter wie die Ladezeit ermittelt. Das Einbetten des zusätzlichen JavaScript-Codes in die Webseiten geschieht *automatisch* durch den Intermediary, sodass sich beliebige Sites mit dem System nutzen lassen.

8.6.4.2 Architektur und Funktionsweise des *Scone AccessTracking*

Das *AccessTracking* lässt sich funktional im Wesentlichen in drei Teilmodule gliedern, die sich jeweils aus mehreren Klassen zusammensetzen (Abb. 143): Das erste Teilmodul besteht aus einer Reihe von *DocumentEditor-MEGs*, die JavaScript-Code und JavaScript-Events in alle Webseiten einbetten. Dieser Code erfasst die Aktionen des Benutzers mit dem Browser und liest zahlreiche Browser-Parameter aus. Diese Daten werden über die zweite Teilkomponente an Scone übermittelt, wobei entweder der *AppletConnector* und den *RAS-Server* (s. Abschnitt 8.6.5.1) oder alternativ der *AccesstrackingGenerator* zum Einsatz kommt. Die dritte Teilkomponente ist der *EventDecoder*, der die Daten aufbereitet und an die *NetObject*-Komponente übermittelt. Im Folgenden wird die Funktionsweise dieser Teilmodule erläutert.

1. Einbinden von JavaScript-Code in Webseiten: Die DocumentEditor-MEGs

Die *DocumentEditor-MEGs* des AccessTrackings fügen zusätzlichen *JavaScript-Code* und *Event-Handler* zum Verarbeiten der Events in alle Dokumente ein.[307] Jeder der MEGs hat eine spezifische Aufgabe, beispielsweise ergänzt einer „`onClick`"-Attribute zu allen Link-Ankern (Münz & Gull 2010) und ein weiterer den Code zum Erfassen von *Lade-* und *Besuchszeiten*. Mittels der derart funktional erweiterten Webseiten lassen sich alle Tätigkeiten mit dem Dokument sowie die *Auswirkungen* der meisten Benutzeraktionen mit dem Browser erfassen. Hierzu gehören die wichtigsten Navigationsaktionen des Benutzers, wie das Anklicken von Links, das Senden von Formulardaten, Maus- und Fensteraktionen sowie das Backtracking.

[307] Eine besondere Herausforderung bestand darin, nicht mit bereits in den Seiten vorhandenem Code in Konflikt zu geraten. Es wird daher zuvor das übertragene Dokument analysiert, und die Ergänzungen werden gemäß den bereits vorhandenen Events angepasst.

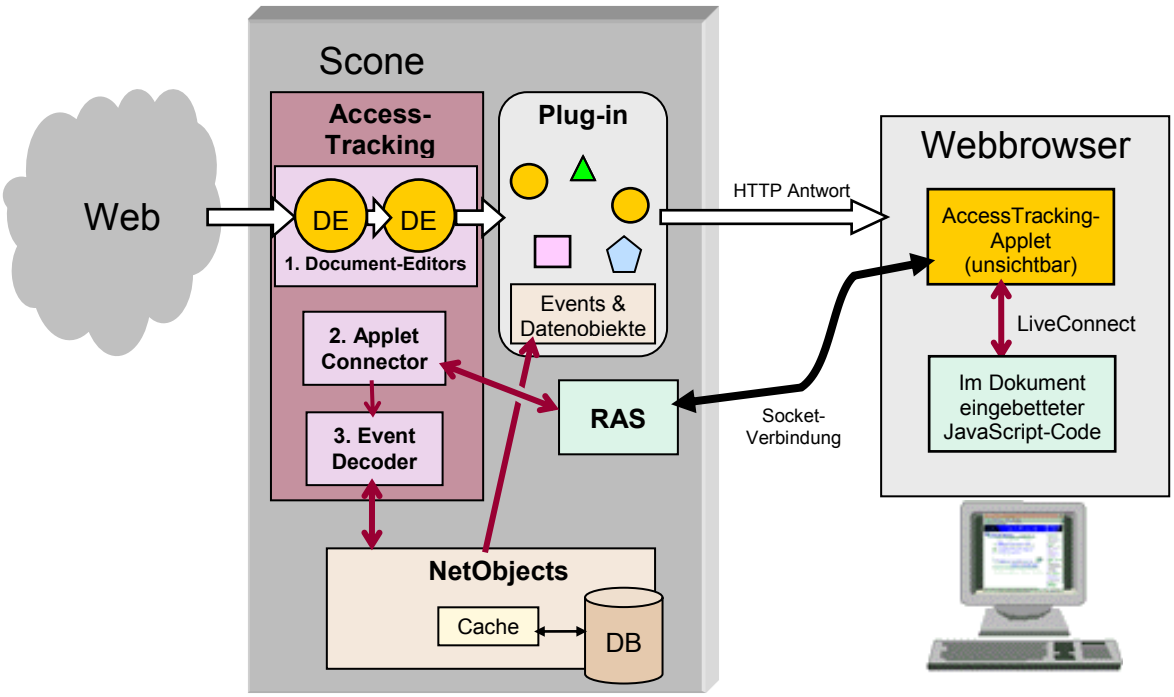


Abb. 143: Schematische Darstellung des Scone *AccessTrackings* mit dem AccessTracking-Applet.

Konkret bietet das Scone *AccessTracking* folgende Möglichkeiten:

- Klicks auf Link-Anker werden identifiziert, indem jeder Link ein zusätzliches „`onClick`"-Attribut erhält, das dazu führt, dass der Identifikator des Link-Objektes in den NetObjects an Scone übermittelt wird.
- Das Übermitteln eines Formulars wird durch zusätzliche „`onSubmit`"-Attribute in den „`form`"-Elementen der Seiten identifiziert. Die an den Server übermittelten Daten werden durch einen Monitor-MEG im Intermediary erfasst.
- Das *Backtracking* (die Verwendung des Back-Buttons) wird ebenfalls über JavaScript ermittelt, sodass solche Aktionen auch erkannt werden, wenn der Browser eine Seite aus seinem Cache lädt. Da kein JavaScript-Event für den Back-Button existiert, wird eine Backtracking-Aktion über die Länge des *History-Stacks* im Browser identifiziert.[308] Um herauszufinden, wie viele Seiten ein Benutzer zurückgesprungen ist, werden im *EventDecoder* (s. u.) parallele History-Stacks zu jedem Fenster und Frame mitgeführt.
- Der *Forward*-Button wird auf entsprechende Weise festgestellt.
- Ein *Reload* führt dazu, dass derselbe URI noch einmal vom Server geladen wird, ohne dass sich die History im Browser ändert, ein Link ausgewählt oder ein Formular abgesandt wird.

[308] Beim Backtracking bleiben alle Seiten auf dem History-Stack erhalten, er bleibt daher in Inhalt und Länge unverändert. Erst wenn der Benutzer von dieser Seite einen anderen Link auswählt, wird ein Teil der History überschrieben (Greenberg & Cockburn 1999). Die Länge des History-Stacks lässt sich mit JavaScript auslesen.

- *Fenster* (und *Browser-Tabs*) werden unterschieden, indem Scone dem Browser-Attribut „`window.name`“[309] einen eindeutigen Namen zuweist. Fenster, die noch keinen Namen haben, werden als neu geöffnete Fenster erkannt.
- *Frames* werden auf ähnliche Weise über die Namen des Frames und des Hauptdokuments identifiziert. Bei Frames weist das Attribut „`window.parent.name`“ den Namen des übergeordneten Bereichs auf, sodass sich zusammengehörige Frames bestimmen lassen.
- An den Anfang jedes Dokuments wird eine Zeile JavaScript eingefügt, die die Systemzeit des Browsers registriert. Dieser Code wird sofort ausgeführt, sobald die ersten Bytes einer Seite beim Browser ankommen und ermittelt somit den Beginn der Datenübertragung. Nachdem das Dokument fertig geladen wurde, wird mittels des „`onLoad`“-Events die *Ladezeit* der Seite bestimmt. Beim Verlassen einer Seite wird mit „`onBeforeUnload`“ der Zeitpunkt des Verlassens einer Seite oder des Schließens eines Browser-Fensters festgehalten.

Weitere Daten, wie die *Größe des Browser-Fensters*, *Scrolling* des Dokuments, die *Bewegung des Mauszeigers* und *Klickpositionen*, lassen sich leicht auf Basis des vorhandenen JavaScript-Codes erfassen und optional an Scone übermitteln. Gewöhnlich werden diese Daten aber nicht erhoben, da sie sehr umfangreich sein können und so unter Umständen die Benutzung des Browsers verlangsamen.

Die so im Browser ermittelten Benutzungsereignisse sind jeweils möglichst schnell vom Browser zum Intermediary zu übertragen, damit auf Scone basierende Erweiterungen unmittelbar auf die Aktionen der Benutzer reagieren können.

2. Übermitteln der Daten an Scone: AppletConnector und AccessTracking-Generator

Für das AccessTracking wurden zwei Konzepte entwickelt, um jederzeit vom Browser Daten an Scone übermitteln zu können.

Das erste Konzept verwendet ein unsichtbares, zusätzlich eingebundenes *Applet*. Der *JavaScript-Code* kommuniziert mittels *LiveConnect* mit dem Applet. Das Applet wiederum öffnet eine Socket-Verbindung zum *RAS-Server* von Scone zur Datenübertragung (s. Abb. 143). Der Vorteil dieser Technik ist ihre Effizienz: es kann durchgängig *eine* zustandsbehaftete Verbindung für sämtliche in einer Webseite erfassten Daten verwendet werden. Die Verbindung lässt sich umgekehrt auch nutzen, um Kommandos von Scone zum Browser zu schicken (s. Abschnitt 8.6.5.4). Die praktischen Defizite dieses Konzepts bestehen darin, dass das Laden und Starten des Applets etwas Zeit kostet und die Integration von Java in keinem der populären Browser so stabil ist, dass das Applet für mehrere Stunden fehlerfrei läuft.[310]

[309] Dies ist einer der wenigen JavaScript-Parameter, der über mehrere Webseiten hinweg seinen Wert beibehält.

[310] Es kam bei allen getesteten Browsern und allen Java Virtual Machines in unregelmäßigen Abständen dazu, dass sich die JVM „aufhing“ und danach keine Applets mehr ausgeführt wurden. Der Browser musste dann neu gestartet werden.

Für Kurzzeit-Benutzbarkeitsstudien von ca. 30 - 90 Minuten Länge ist es aber problemlos einsetzbar.

Das zweite Konzept zur Datenübertragung verwendet HTTP-Requests, die über den virtuellen URI http://_tracking.scone.de/ an einen *Generator-MEG* im Intermediary gerichtet werden (s. Scone Proxy 8.6.1). Diese Verbindungen werden auch nach dem vollständigen Laden einer Seite aufgebaut, indem der JavaScript-Code über das Image-Objekt (für den Benutzer unsichtbare) Grafiken vom AccessTracking-Generator anfordert;[311] die zu übermittelnden Benutzungsdaten sind dabei in dem URI des Requests als GET-Parameter codiert. Eine weitere Möglichkeit ergibt sich seit ca. 2004 in den meisten Browsern durch das *XMLHttpRequest-Objekt:* Mit seiner Hilfe können innerhalb von Webseiten beliebige Daten per HTTP zum und vom Server übertragen werden (Crane, Pascarello et al. 2005; Garrett 2005). Diese Technik ist 2005 unter dem Namen Ajax (**A**synchronous **J**avascript **a**nd **X**ML) für interaktive Webseiten populär geworden (s. Abschnitte 2.2.5 und 9.2).

3. Die Auswertung der Benutzeraktionen: Der EventDecoder

Die dritte Teilkomponente des *AccessTrackings* ist der *EventDecoder*, der die vom Browser an Scone übermittelten Daten auswertet und zu entsprechenden Ereignissen zusammenfasst. Es gibt mehrere Unterklassen des *EventDecoders*, die die Eigenheiten der verschiedenen Browser berücksichtigen.[312]

In der Regel wird für jede Navigationsaktion des Benutzers – dies sind alle Tätigkeiten, die zur Darstellung einer neuen Seite im Browser führen (siehe Weinreich, Obendorf et al. 2006b) – ein neues *Access*-Objekt innerhalb der NetObjects erstellt und abgespeichert (s. Abb. 140 und Abb. 143). Dieses Objekt enthält die oben angeführten Informationen, also beispielsweise, wie ein Benutzer auf eine Seite gelangt ist, wie lange er auf der Seite war, welchen Link er zuvor auswählte oder welche Daten er in ein Formular eingab. Mithilfe des *Observable*-Entwurfsmusters lassen sich angemeldete Plug-ins benachrichtigen, wenn ein solches Objekt erstellt oder aktualisiert wird, sodass sie auf die Aktivitäten des Benutzers reagieren können.

---

[311] Diese Anfragen werden an den *virtuelle URI* http://_tracking.scone.de/ gerichtet, d. h. der entsprechende Domain-Name existiert nicht, und stattdessen fängt ein *Intermediary-Monitor* in Scone die Anfrage ab, wertet die übergebenen Parameter aus und liefert ein kleines transparentes GIF-Bild zurück.

[312] Konkret handelt es sich bei diesen „Eigenheiten" hauptsächlich um Fehler in den Browsern. Beispielsweise liefert der Internet Explorer unter bestimmten Umständen eine falsche Größe des History-Stacks zurück, die daher bei der Auswertung berücksichtigt werden muss.



Abb. 144: Das Scone *UserHandling* wird über den Webbrowser bedient.

8.6.4.3 Identifikation und Verwaltung von Benutzern: *Scone UserHandling*

Ein weiterer Teil des Scone AccessTrackings ist die Verwaltung von Benutzerkonten, das *Scone UserHandling*. Es identifiziert Anwender mithilfe von Cookies (Kristol & Montulli 2000) und kann so mehrere Nutzer eines Computers unterscheiden und erlaubt es Anwendern, von mehreren Computern auf ein Konto zuzugreifen.

Die HTTP-Cookies des *UserHandlings* werden jedes Mal zum virtuellen URI des *Scone AccessTrackings* übertragen, wenn der Browser eine Benutzungsaktion an sie übermittelt. Auf diese Weise wird das Sicherheitskonzept von Cookies berücksichtigt,[313] auch wenn Benutzer auf beliebige Ressourcen im Web zugreifen.

Abb. 144 zeigt den Screenshots der Benutzerverwaltung von Scone. Scone bietet grundlegende Verwaltungsfunktionen wie die Registrierung neuer Benutzer und das Login zur Verfügung:

- Ist noch kein Benutzer-Cookie gesetzt, wird der Anwender beim ersten Aufruf einer Webseite automatisch zur Scone-Benutzerverwaltung geleitet. Diese ist auch unter dem virtuellen URI `http://users.scone.de/` zu erreichen (s. Abb. 144, links).
- Hier kann er, falls gewünscht, ein neues Benutzerkonto einrichten (s. Abb. 144, rechts).
- Ist er bereits als Benutzer registriert, so kann er sich mit seinem Benutzernamen und Passwort beim System anmelden.

Ein Anwender bleibt angemeldet, bis er sich unter einem anderen Namen in Scone einloggt.

8.6.4.4 Grenzen der Erfassung von Benutzeraktionen mit Intermediaries

Das *AccessTracking* erfasst viele Details der Benutzeraktionen, die im Übertragungsprotokoll eines „normalen“ Intermediary verborgen bleiben. Zudem ist es unabhängig vom System des Anwenders: Auf dem Computer, Tablet PC oder Smartphone des Benutzers muss keine

[313] Cookies lassen sich nur innerhalb der URIs eines Domain-Namens setzen und lesen, nicht aber über mehrere Websites hinweg (Kristol & Montulli 2000).

spezifische Software installiert werden. Darüber hinaus lässt sich das AccessTracking eines Scone-Plug-ins von mehreren Personen gleichzeitig nutzen, um es beispielsweise in Arbeitsgruppen einzusetzen.

Die eingesetzte Technik weist jedoch Grenzen auf: Zum einen lässt sich die Interaktion mit eingebundenen Objekten wie Flash-Animationen oder Java Applets nicht erfassen, ein Defizit das allerdings auch auf viele andere Ansätze zutrifft, wie die Modifikation des Browser-Quellcodes und Browser-Erweiterungen (s. Abschnitte 8.3.1 und 8.6.4.1). Darüber hinaus lassen sich Benutzeraktionen mit vielen Schnittstellenelementen des Browsers – z. B. Pulldown-Menüs und Cut-And-Paste-Operationen – nicht erkennen, sofern sie keine Auswirkung auf das dargestellte Dokument haben.

Um diese Beschränkungen zu umgehen, wurden für Scone zusätzlich zwei browser-spezifische Systeme für die Aufzeichnung von Benutzeraktionen entwickelt.

- Der *„IESpy"* ist eine Erweiterung in C++ für den *Internet Explorer*. Das System basiert auf einem *BrowserHelperObjekt* und einer Anwendung, die über die COM-Schnittstelle alle Events im Browser registriert (s. Abschnitt 8.3.1). So lassen sich nahezu sämtliche Benutzeraktionen mit dem *Internet Explorer* ermitteln. Die Daten werden vom *IESpy* zusammengefasst und über den AccessTracking-Generator an Scone übermittelt (Haß 2004; Obendorf, Weinreich et al. 2004).
- Anfang 2005 wurde zusätzlich eine Erweiterung für *Firefox* realisiert, die in dessen XUL-Schnittstelle integriert ist (Weinreich, Obendorf et al. 2006a). Die Erweiterung kann alle Benutzeraktionen mit den Bedienelementen des Browsers registrieren. Die erfassten Daten werden über den *AccessTracking-Generator* an Scone übermittelt. Diese Erweiterung wurde in einer mehrwöchigen Benutzungsstudie des Webs mit 25 Teilnehmern eingesetzt. Dabei sind über 350.000 Benutzeraktionen aufgezeichnet und später im Detail ausgewertet worden (siehe: Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Weinreich, Obendorf et al. 2008).

8.6.5 Hilfskomponenten für das Framework zur Erweiterung des Webs

Das Framework bietet zur Erfüllung der Anforderungen (s. Abschnitt 8.2) mehrere Hilfskomponenten, die von den zuvor vorgestellten Kernkomponenten als auch direkt von *Scone Plug-ins* (s. Abschnitt 8.6.6) verwendet werden können. Die Wichtigsten dieser Komponenten werden im Folgenden vorgestellt: Der *Remote Access Service (RAS)* ermöglicht die direkte, asynchrone Kommunikation zwischen Webbrowser und Scone (s. Abschnitt 8.6.5.1). Der *HTMLStreamTokenizer* zerlegt HTML-Seiten in einzelne Elemente („Token") und kann auch fehlerhaften HTML-Code interpretieren (Abschnitt 8.6.5.2). Der *DocumentParser* extrahiert Metadaten aus Webseiten (Abschnitt 8.6.5.3), *BrowserControl* dient dazu, Steuerbefehle an den Browser zu schicken (Abschnitt 8.6.5.4), und der *Thumbnail-Generator* erstellt mithilfe eines in Java geschriebenen Web-Clients Thumbnails von Webseiten (Abschnitt 8.6.5.5). Das *ServerSide-Plug-in* dient dazu, Scone auch seitens eines Webservers einsetzen zu können

(Abschnitt 8.6.5.6), und das *UserTestTool „TEA“* ist eine Erweiterung von Scone, die die Durchführung partizipativer Tests von Websites und Scone Plug-ins unterstützt (Abschnitt 8.6.5.7).

8.6.5.1 Ein leichtgewichtiger Dienst zur verbindungsorientierten, asynchronen Kommunikation im Web: *Scone RAS*

Das Framework bietet unter dem Namen *Scone RAS („Remote Access Service“)* einen Dienst, der anderen Systemen per TCP/IP-Socket-Verbindung den Zugriff auf Klassen und Objekte von Scone ermöglicht. Diese Hilfskomponente wurde speziell für die *Kommunikation* zwischen dem Browser bzw. in Webseiten eingebundene Programme und mit Scone entwickelten Systemen ausgerichtet. Zusätzlich wird eine Bibliothek für Java-Applets angeboten, die die Verbindungen zwischen Applets und dem RAS-Server kapselt und über *LiveConnect* die Kommunikation mit beliebigen JavaScript-Funktionen im Browser ermöglicht. Der Dienst lässt sich ebenfalls aus Flash-Objekten, ActiveX-Controls oder direkt aus JavaScript-Programmen (mittels des für Ajax-Anwendungen bekannten XMLHttpRequest-Objekts, siehe: Crane, Pascarello et al. 2005; Garrett 2005) nutzen.

Der RAS-Server wird mittels der Plug-in-Requirements (s. Abschnitt 8.6.6) aktiviert, an einen IP-Port gebunden und horcht als Server-Socket auf Client-Anfragen. RAS-Verbindungen werden somit von der Client-Anwendung initiiert, der Server kann jedoch über eine bestehende Verbindung auch Daten an den Client senden. Beim Verbindungsaufbau gibt der Client den Klassennamen eines *ConnectionHandlers* an. Wird die entsprechende Klasse gefunden, so instanziiert der Server ein Objekt der Klasse und übergibt die weiteren Parameter an das Objekt. Die Client-Applikation und der *ConnectionHandler* können daher das Protokoll der Anwendung selbst definieren, sodass das Konzept beliebig erweiterbar ist.

Mithilfe des RAS-Servers, eines RAS-Applets und des LiveConnect-Zugriffs auf die JavaScript-Funktionen des Browsers lassen sich Daten innerhalb von Webseiten nachladen, auch *nachdem* eine Seite bereits übertragen wurde und *während* der Anwender mit der Webseite interagiert. Dieses Konzept macht *interaktive Erweiterungen* für Webbrowser *auf Basis von Intermediaries* erst möglich. Mit dem RAS-Konzept wurde folglich bereits vor 2002 eine asynchrone Interaktionstechnik für Webbrowser konzipiert und realisiert, die in vergleichbarer Form erst gegen 2005 unter dem Namen *Ajax* (oder „Web 2.0“) populär wurde. Darüber hinaus ist es mit dem RAS-Applet (im Gegensatz zu Ajax) ebenfalls möglich, Daten im „Push-Modell“ an Webbrowser zu *schicken*.

Der RAS-Server wird von den Scone-Komponenten *AccessTracking* (s. Abschnitt 8.6.4) und *BrowserControl* (s. Abschnitt 8.6.5.4) verwendet. Ein weiteres Beispiel für den Einsatz des RAS sind die HyperScout-Prototypen. Hier bezweckt der Dienst das Nachladen von zusätzlichen Informationen zu Hyperlinks, sobald der Benutzer die Maus über einen Link-Anker bewegt. Dies gewährt dem HyperScout-Client zusätzliche Zeit zum *Prefetching* von Informationen zu den Link-Zielen, sodass sie nicht bereits bei der Übertragung der aktuellen Seite vorliegen

müssen (s. Abschnitt 5.5.3). Weitere Details zum RAS-Konzept findet man in (Buchmann 2002: 39ff).

Seite 2010 gibt es Bestrebungen anderer Projekten, eine ähnliche Funktionalität wie Scone RAS zu realisieren. Im Mai 2010 hat Google die *Channel API* für das Web Application Framework *Google App Engine* vorgestellt. Zur Channel API gehören Bibliotheken in Java und JavaScript, die ähnlich wie beim RAS den Aufbau persistenter, bidirektionaler Verbindungen zwischen Browser und Server erlauben. Es wird aber (wie bei Ajax-Anwendungen) das XMLHttpRequest-Objekt verwendet und somit auf das langsamere, verbindungslose HTTP-Protokoll gesetzt.[314] Für HTML5 ist eine neue Schnittstelle namens *WebSocket* vorgesehen, die ein einfaches Protokoll auf TCP/IP-Basis verwendet und zum Aufbau von leichtgewichtigen, dauerhaften und bidirektionalen Verbindungen zwischen Browser und Server dient. Web Sockets von der Funktionalität her vergleichbar mit dem Scone RAS, aber für Online-Applikationen vorgesehen (Weßendorf 2011). Für *WebSocket* liegt seit April 2011 ein Entwurf vor,[315] und der Standard wird von vielen aktuellen Browsern außer dem Internet Explorer unterstütz. Diese Einschränkung besteht beim RAS-Konzept nicht.

8.6.5.2 Ein performanter, fehlertoleranter HTML-Parser: *HTMLStreamTokenizer*

Damit sich Webseiten von Scone analysieren und modifizieren lassen, wurde ein Parser benötigt, der alle im Web zugänglichen (X)HTML-Dokumente schnell und zuverlässig interpretiert (s. Anforderungsanalyse in den Abschnitten 8.2.1 und 8.2.3). Tests zeigten, dass die gängigen XML- und HTML-Parser in Java für Scone ungeeignet waren, da sie Probleme mit Syntaxfehlern in HTML-Seiten hatten (s. Abschnitt 8.6.1): Ein Großteil[316] der Dokumente im Web weist syntaktisch fehlerhaften (X)HTML-Code auf, sodass diese Parser bei Tests die Verarbeitung von Seiten häufig mit Fehlermeldungen abbrachen oder nur Teile der Dokumente zur Verfügung stellten. Für Scone wurde daher ein HTML-Parser benötigt, der sich bei der Interpretation ähnlich tolerant verhält wie gängige Webbrowser.[317]

Der *HTMLStreamTokenizer* von Scone erfüllt diese Anforderungen. Der HTML-Parser basiert auf dem gleichnamigen Open-Source-Projekt von Arthur Do,[318] das bereits über grundlegende Konzepte zum Umgang mit Fehlern im HTML-Code verfügt. Diese Mechanismen

---

[314] Die Dokumentation der *Channel API* findet man unter: http://code.google.com/appengine/docs/java/channel/.

[315] Den *WebSocket*-Entwurf findet man unter: http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-07.

[316] Bei einer umfangreichen Stichprobe Anfang 2005 hatten über 95% der untersuchten Webseiten einen fehlerhaften HTML-Code (Bleich 2005; Leporda 2005). Eine weitere Studie kam 2008 zu einem vergleichbaren Ergebnis (Wilson 2008).

[317] Hier kann nur spekuliert werden, was die Entwickler des Netscape Navigators und später der anderen Browser bewogen hat, dieses Verhalten zu implementieren: Vermutlich war die Fehlertoleranz darin begründet, dass in den Anfangszeiten des Webs viele Seiten ohne Autorenwerkzeuge erstellt wurden und sie daher Syntaxfehler aufwiesen. Spätere Browser mussten dasselbe Verhalten aufweisen, um „kompatibel“ zu sein.

[318] Der *HtmlStreamTokenizer* ist unter folgender Adresse zu finden: http://sourceforge.net/projects/htmltok/.

wurden für Scone mithilfe eines mehrmonatigen Praxiseinsatzes verfeinert, sodass der Parser gängige Fehler in HTML-Seiten mit vergleichbaren Heuristiken korrigiert wie populäre Browser.

Zwei Beispiele illustrieren die Funktionsweise der eingesetzten Heuristiken:

- Häufig findet man Syntaxfehler bei der Angabe von Attributen für HTML-Elemente. In der folgenden Zeile fehlen u. a. die schließenden Anführungszeichen des dritten Attributs:

  ```
  <img width=20 height='10' src="http://www.scone.de/logo.gif>
  ```

  Der HTML-Parser von Scone korrigiert das Element, indem es die schließende spitze Klammer als Ende des Elementes berücksichtigt. Dies macht der Parser allerdings nur, wenn dahinter die Zeile endet; ansonsten wird die spitze Klammer als Teil des Attributwerts interpretiert, da populäre Webbrowser ebenso verfahren. Nach der Verarbeitung durch Scone wird folgender Code an den Browser übertragen:

  ```
  <img width="20" height='10' src="http://www.scone.de/logo.gif">
  ```

- Ein zweiter üblicher Fehler betrifft die irreguläre Verwendung von Steuerzeichen, wie des „Ampersand"-Symbols (auch „Kaufmanns-Und"), das eine HTML-Entity einleitet. Häufig wird es nicht entsprechend codiert. Scone erkennt diesen Fehler und korrigiert den ungültigen HTML-Code „ `AT&T.` " zu „ `AT&T.` ", da „`&T.`" keine Entity darstellt.

Als Resultat liefert der HTMLStreamTokenizer einen von gängigen Fehlern bereinigten, gut weiter zu verarbeitenden Datenstrom von HTML- und Text-Token. Die Zuverlässigkeit der entwickelten Heuristiken hat sich in einer Langzeitstudie gezeigt, in der 25 Personen über einen Zeitraum von bis zu 28 Wochen mit Scone auf über 150.000 Webseiten zugriffen, ohne dass sie über Probleme berichteten (s. Abschnitt 8.7).

Der HTMLStreamTokenier wird sowohl im Scone Proxy (s. Abschnitt 8.6.1.1) als auch im *Scone Robot* eingesetzt (s. Abschnitt 8.6.2.2).

8.6.5.3 Ein Interpreter für Web-Dokumente: *Scone DocumentParser*

Zur weiteren Analyse von HTML-Dokumenten bietet Scone den sogenannten *DocumentParser*. Diese Hilfskomponente extrahiert Meta-Informationen aus Webseiten und stellt sie als Objekte in den *Scone NetObjects* zur Verfügung (s. Abschnitt 8.6.3).

Die Informationen werden aus dem vom *HTMLStreamTokenizer* (s. Abschnitt 8.6.5.2) vorbereiteten HTML-Code und dem HTTP-Header der Seite gewonnen. Zu jeder Seite erstellt der DocumentParser ein entsprechendes *HTMLNode-Objekt* (s. Abschnitt 8.6.3.2), das Attribute wie die *letzte Aktualisierung* einer Seite, den *Seitentitel* und die Werte von Meta-Tags (wie *Autor*, *Beschreibung* und *Schlüsselwörter*) enthält (vergl. Daviel 1996). Weiterhin generiert der Parser mittels eines heuristischen Verfahrens eine kurze *Zusammenfassung* der Webseite,

die auf den ersten Zeilen des Inhaltsbereiches beruht. Ein *Fingerprint* repräsentiert den gesamten Seiteninhalt, sodass sich Änderungen seit dem letzten Besuch feststellen lassen.

Eine weitere Aufgabe des DocumentParsers ist die Auswertung struktureller Informationen des Webs. Hierzu gehören Daten über die internen und externen *Links* sowie die *eingebundenen Objekte*. Welche Informationen extrahiert und in den NetObjects bereitgestellt werden sollen, wird mittels der „*requirements*“[319] im Scone Plug-in definiert (s. Abschnitt 8.6.6).

8.6.5.4 Entfernte Steuerung des Browsers: *Scone BrowserControl*

Eine weitere Hilfskomponente gewährleistet von Scone aus den Zugriff auf viele Funktionen des Webbrowsers. Die *BrowserControl*-Komponente nutzt den RAS-Server (s. Abschnitt 8.6.5.1) und das Applet des *AccessTrackings* (s. Abschnitt 8.6.4), um Kommandos an den Browser zu schicken. Dazu bindet der Intermediary in jede Webseite neben dem unsichtbaren Applet einige Zeilen JavaScript-Code ein, der vom AccessTracking-Applet über LiveConnect aufrufbar ist.

BrowserControl bietet Funktionen, um von Scone aus *Größe und Position* des Browser-Fensters zu ändern, den Browser auf dem Desktop in den *Vordergrund* zu holen, ein *neues Fenster* zu öffnen und eine *neue Webseite* zu laden. Darüber hinaus lässt sich praktisch jedes beliebige weitere JavaScript-Kommando „ferngesteuert“ im Browser ausführen, indem es als Quellcode an eine „generische Funktion“ (s. Abb. 145) übergeben wird, die den übergebenen Code direkt im Browser ausführt.

```
function _scone_anyFunction(func) {
      eval(func);
})
```

Abb. 145: JavaScript-Funktion von *Scone* zur Ausführung beliebiger Kommandos im Browser.

Mit der *BrowserControl*-Hilfskomponente steht Entwicklern von Scone Plug-ins ein Mechanismus zur Fernsteuerung von Webbrowsern zur Verfügung, für die bis heute (nach Kenntnis des Autors) keine vergleichbare Lösung existiert (s. auch Abschnitt 8.6.5.1).

8.6.5.5 Zur grafischen Repräsentation von Webseiten: *Thumbnail-Generator*

Eine beliebte Möglichkeit zur Repräsentation von Webseiten sind *Thumbnails*. Diese verkleinerte grafische Darstellung bietet gerade beim Wiedererkennen von Webseiten häufig Vorteile (Bederson, Hollan et al. 1996; Hightower, Ring et al. 1998; Zühlsdorff 2002).

Die automatische Erstellung der Thumbnails von Webseiten ist relativ aufwendig, da ein (programmierbarer) Webbrowser benötigt wird, der die Dokumente vollständig mit ihren eingebetteten Objekten rendert und dessen Ausgabe sich als Grafik auslesen lässt. Andere

[319] Beispiele für solche Requirements sind PARSEDOCUMENT für das Auswerten inhaltlicher Informationen, CONSIDERLINKS für die Erstellung von Link-Objekten zu jedem Link-Anker im Dokument und SAVESOURCECODE, falls der Quellcode eines Dokuments im HTMLNode gespeichert werden soll.

Projekte verwenden hierfür entweder die COM-Schnittstelle des Internet Explorers (z. B. Kaasten & Greenberg 2001), eine zusätzliche Workstation mit einem „ferngesteuerten" Browser (z. B. Kopetzky & Mülhäuser 1999) oder eine externe Thumbnail-Datenbank,[320] die Webseiten als kleine Grafiken bereitstellt (Joho & Jose 2006).

Da Scone in Java implementiert wurde und das System Thumbnails beliebiger Seiten und beliebiger Größe erstellen sollte, wurde ein in Java-Programme integrierbarer Webbrowser benötigt. Der *ThumbnailGenerator* von Scone bindet hierfür die Java-Komponente *WebView* ein.[321] Zur Erstellung eines Thumbnails müssen dem Generator lediglich die Adresse als *SimpleURI* (s. Abschnitt 8.6.3.3ff) und das Format der Ausgabe übergeben werden. Das System zeichnet dann in einem nicht sichtbaren Bereich das entsprechende Dokument, die Ausgabe wird auf das gewünschte Maß skaliert und als Objekt der Klasse „`java.awt.Image`" zurückgeliefert.

Diese Hilfskomponente wurde u. a. in den Projekten „SessionGraphs SGA" (Mayer 2009) und „Google Rückspiegel" (Wollenweber 2002) verwendet (s. Abb. 146 und Abschnitt 8.7).

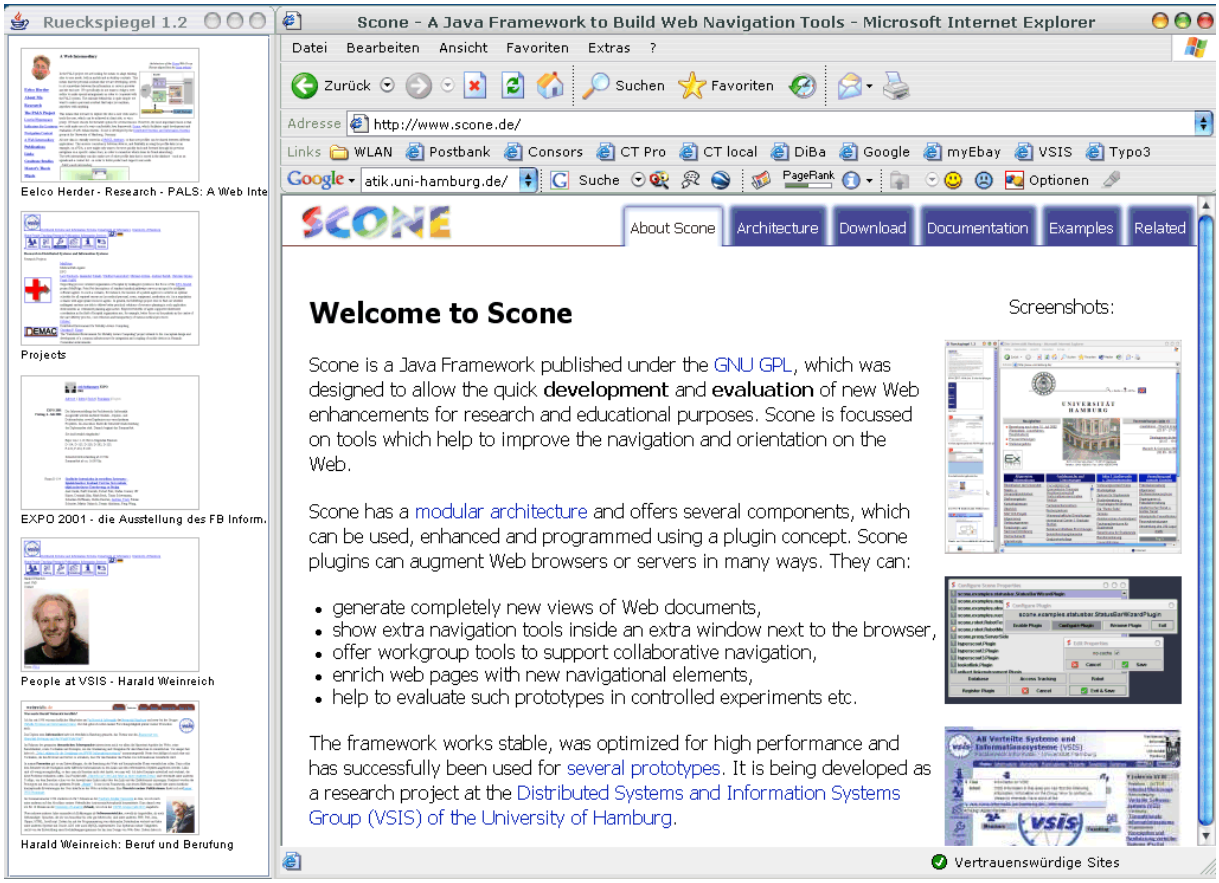


Abb. 146: Der Scone Rückspiegel zeigt Thumbnails von Seiten, die auf die aktuelle Seite verweisen.

[320] Beispiele für solche Dienste sind „*websnapr 2.0*" (www.websnapr.com) und *ThumbShots* (www.thumbshots.de).

[321] *WebView* wurde von Horst Heistermann entwickelt. Die Komponente war zuerst kostenlos erhältlich, ab 2001 ein kommerzielles Produkt, wird aber seit 2007 nicht mehr vertrieben.

8.6.5.6 Der Intermediary als Reverse Proxy: *Scone ServerSide-Plug-in*

Bereits in (Barrett & Maglio 1998) wird beschrieben, dass Intermediaries sowohl auf Clientseite, für Workgroups als auch auf Serverseite einsetzbar sind. Es gibt aber einen grundsätzlichen Unterschied zwischen einem clientseitigen und einem serverseitigen Intermediary: Anstatt als Proxy für *alle* zugegriffenen Websites weniger Benutzer zu arbeiten, sind serverseitige Intermediaries genau für *eine* Website zuständig, dessen Seiten sie für *alle* Besucher modifizieren und unter der eigenen Adresse (sozusagen gespiegelt) zur Verfügung stellen (s. Abb. 147). Man nennt solche Systeme daher auch „Reverse Proxys“.[322] Beispielsweise könnte Scone mit dem ServerSide-Plug-in die Ressourcen und Dienste der Site `www.host.de` unter der neuen Adresse `www.newhost.de` modifiziert bereitstellten.

Eine solche Replikation eines Webservers erfordert zusätzliche Funktionen im Intermediary, um die Anfragen des Browsers anzupassen und an den zu repräsentierenden Server weiterzuleiten. Ebenso müssen die Antworten des Servers auf die neue Adresse umgesetzt werden. Die entsprechenden Methoden zum serverseitigen Einsatz von Scone bzw. WBI wurden daher im Rahmen von Scone als *ServerSide-Plug-in* konzipiert und realisiert. Sie sind nun Teil der Standard-Distribution von WBI.

Das ServerSide-Plug-in besteht im Wesentlichen aus zwei MEGs (s. Abb. 147):

- Der „`RewriteRequestEditor`“ übersetzt den URI und den Host im HTTP-Request des Browsers. Diese geänderte Anfrage wird dann an den Webserver weitergeleitet, der sie beantwortet, als wäre sie direkt vom Browser an ihn gerichtet worden. Im Beispiel ist die Anfrage an `www.newhost.de` auf `www.host.de` abzuändern und weiterzuleiten.
- Der „`RewriteRequestEditor`“ passt die in der abgerufenen Seite enthaltenen Adressen an, damit der Browser den „Eindruck“ gewinnt, die Antwort direkt von einem Webserver unter der Adresse `www.newhost.de` erhalten zu haben. Insbesondere sind alle Verknüpfungen in den Dokumenten, die auf Ressourcen des ursprünglichen Servers zeigen, auf die neue Adresse abzuändern. Dies gilt nicht nur für Hyperlinks, sondern auch für eingebundene Objekte.[323] So wird gewährleistet, dass alle Anfragen einer Sitzung ebenfalls durch den Intermediary bearbeitet werden und nicht direkt an die Originaladresse `www.host.de` gehen.[324] Im Beispiel wäre der Link zur Seite `http://www.host.de/info.html` auf die Adresse `http://www.newhost.de/info.html` zu übersetzen.

---

[322] Es gibt zahlreiche Einsatzbereiche für serverseitige Intermediaries (*Reverse Proxies)*. Hierzu gehört das *Sichern von Web-Servern*, das *Load-Balancing* und das Transkodieren von Protokollen und Daten. Entsprechende Module gibt es beispielsweise für den Apache-Webserver (http://httpd.apache.org/docs/2.0/mod/mod_proxy.html) und den *Caching Proxy SQUID* (http://wiki.squid-cache.org/SquidFaq/ReverseProxy).

[323] Sofern andere Objekte wie Bilder nicht durch den Intermediary bearbeitet werden sollen, können diese URIs auch direkt auf Objekte des originalen Web-Servers verweisen.

[324] Benutzer können normalerweise auch weiterhin unter der alten Adresse auf den Originalserver zugreifen, sofern der Domain-Name nicht geändert wurde.

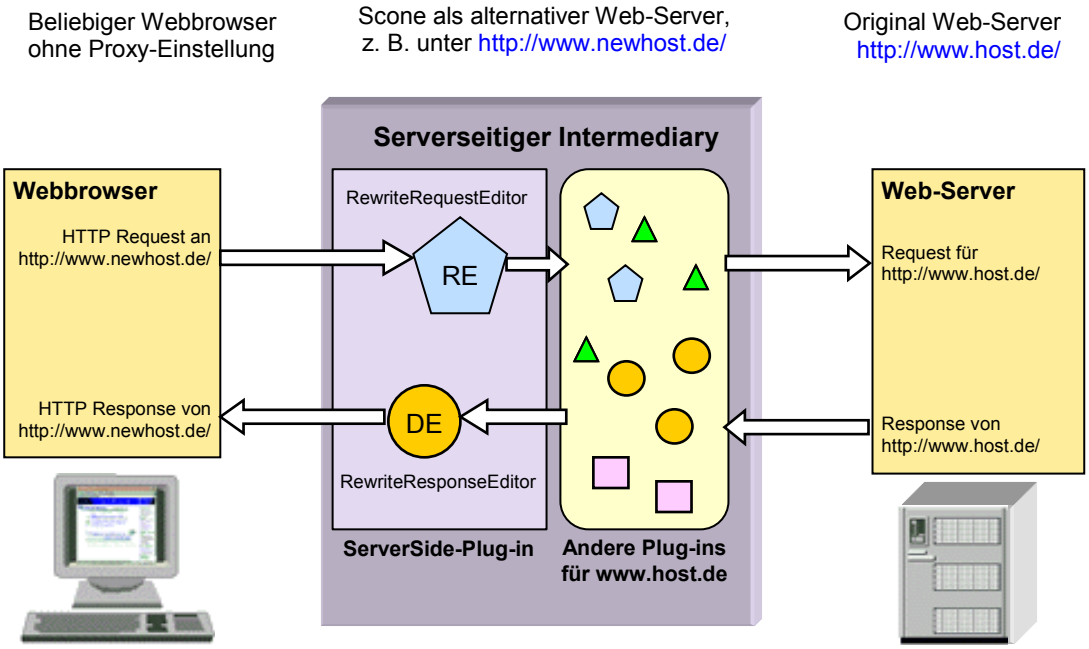


Abb. 147: Beispiel für den Einsatz des Intermediary von Scone auf Serverseite.

Mit dieser Methode werden alle Anfragen an eine Domain durch Scone geleitet, und es lassen sich nahezu beliebige Erweiterungen einer Website vornehmen und testen, ohne dass der Webserver selbst geändert werden muss.

8.6.5.7 Evaluation von Websites und Scone Plug-ins: *TEA UserTestTool*

Die letzte hier vorgestellte Hilfskomponente ist recht umfangreich und für sich bereits eine eigene sinnvolle Anwendung von Scone. Das Plug-in *TEA UserTestTool*[325] bietet den Entwicklern von Scone Plug-ins vielfältige Möglichkeiten zur Evaluation ihrer Systeme. Das Werkzeug lässt sich aber auch unabhängig davon als Testumgebung für die Untersuchung von webbasierten Diensten und Websites einsetzen.

Das TEA UserTestTool entstand aus der Motivation heraus, dass partizipative Softwaretests in der Regel recht arbeitsaufwendig sind, eine hohe Kompetenz der Testleiter erfordern und daher mit erheblichen Kosten verbunden sind. Es gibt zwar zahlreiche Systeme zur Unterstützung von Benutzertests, aber nur wenige bieten eine teilweise Automation der Testabläufe an (Ivory & Hearst 2001; Lorenzen-Schmidt & Nufer 2008). Folglich müssen häufig die Experten, die einen Test gestaltet haben, auch selbst die Durchführung überwachen.

Bezogen auf das Web fehlt ein System, das Personen bei Benutzbarkeitstests unterstützt und dabei den Webbrowser in den Versuch aktiv mit einbezieht, um beispielsweise bestimmte Seiten zu laden, auf Benutzeraktionen mit dem Browser zu reagieren oder das Vorgehen der Teilnehmer aufzuzeichnen. Eine solche Unterstützung schafft die Möglichkeit, Tests auch durch Dritte durchführen zu lassen und genauer reproduzierbare Testabläufe zu erhalten.

---

[325] *TEA* steht für *Test Environment for Automation* und wurde wegen des zu Scone (kleines britisches Milchbrötchen) passenden Akronyms gewählt.

Die Auswertung der Testdaten gilt als weiterer arbeitsintensiver Abschnitt partizipativer Testverfahren (Nielsen 1993b), wobei der Aufwand von der Qualität der erfassten Daten abhängt. Video-, Maus- und Tastaturprotokolle erfordern in der Regel eine mühevolle Nachbearbeitung der Daten, um die Aufzeichnungen bestimmten Aktionen und Aufgaben zuordnen zu können. Bei der Evaluation von webbasierten Systemen sollten daher die Navigationsschritte der Teilnehmer mit dem Browser in zweckmäßiger, zur einfachen Weiterverarbeitung geeigneter Weise protokolliert werden.

Das TEA UserTestTool wurde entwickelt, um die aufgeführten Ziele zu erreichen. Es lehnt sich an eine Entwicklung des renommierten *HCI-Labs* von Ben Shneiderman und Ben Bederson der University of Maryland an, die ein Java-Programm für die Teilautomation unterschiedlicher Studien einsetzen. Das System des HCI-Labs wurde 2002 vom Autor dieser Arbeit prototypisch in Scone integriert, um die in (Obendorf & Weinreich 2003) beschriebenen Tests durchzuführen. Die Kombination erwies sich damals als sehr nützlich. Diese Erfahrungen und der Mangel an anderen Systemen für die Durchführung von Tests mit Websites und mit neuen Web-Prototypen führten zu der Idee, ein entsprechendes Werkzeug im Rahmen einer Diplomarbeit zu entwickeln (siehe Haß 2004). Das hierbei realisierte TEA UserTestTool[326] geht über die Möglichkeiten des Systems vom HCI-Lab hinaus und erfüllt alle oben aufgeführten Anforderungen für die Evaluation von webbasierten Systemen.

Neue Tests können (anders als beim Vorgänger vom HCI-Lab) *ohne* Java-Programmierung in Form von XML-Dateien definiert werden. Die XML-Sprache des TEA UserTestTools besteht aus 65 Elementen mit zahlreichen Attributen. Ein entsprechendes XML-Schema (Biron, Permanente et al. 2004; Thompson, Beech et al. 2004) vereinfacht die Erstellung der XML-Dateien mithilfe gängiger XML-Werkzeuge wie XML-Spy.[327]

Aus der XML-Definition werden die Ein- und Ausgabeelemente des Testwerkzeuges sowie der Testablauf abgeleitet. Es gibt Elemente für die Beschreibung von Ausgabetexten, Eingabefeldern, Knöpfen, Auswahlboxen, Likert-Skalen als auch Schieberegler für kontinuierliche Skalen. Die Bezeichner der Elemente und ihr Aussehen lassen sich über die Attribute festlegen und an die Erfordernisse des Tests anpassen. Darüber hinaus können wiederkehrende Teile eines Tests als "Templates" definiert werden. Abb. 148 zeigt einen Ausschnitt aus einer solchen XML-Definitionsdatei zur Illustration.

Das TEA UserTestTool wird gestartet, indem man im Browser die virtuelle Adresse http://usertest.scone.de/ aufruft (vergl. Abschnitt 8.6.4.2). Als Folge wird der Startbildschirm des Werkzeugs neben dem Browser dargestellt. Hier können neue Testläufe eingeleitet oder zuvor abgebrochene Tests fortgesetzt werden (s. Abb. 149).

---

[326] Das System basiert auf der Diplomarbeit Haß 2004 und wurde vom Autor dieser Dissertation um mehrere Ein- und Ausgabeelemente als auch Ablaufkontrollfunktionen erweitert und bietet nun die hier und in (Obendorf, Weinreich et al. 2004) geschilderten Möglichkeiten.

[327] XML-Spy wird von *Altova Inc.* angeboten: http://www.altova.com/products/xmlspy/xmlspy.html.

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!-- Wurzelknoten mit dem Link zur XML-Schema-Definition -->
<userTest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation='userTestDescription.xsd'>

  <!-- Der Layout-Bereich definiert wieder verwendbare Elemente -->
  <layout>

    <!-- Definition eines Textes, der oft verwendet wird. -->
    <template name="headOfPage">  <!-- Name des Templates -->
      <text name="relax">Please remember that we are<b>not testing your performance!</b>
           You are helping us to improve our client's Website.</text>
    </template>

    <!-- Eine Dropdown-Box mit vordefinierten Werten. Diese lassen sich auch überschreiben -->
    <template name="chooser">    <!-- Name des Templates -->
      <dropDownBox name="ddbox">
        <item selectable="false" showThisFirst="true">Please choose...</item>
        <item value="0">very ugly</item> <item value="5" >ugly</item>
        <item value="10">not bad</item>  <item value="15">good</item> <item value="20">great</item>
      </dropDownBox>
    </template>
  ...

  </layout>

  <task name="introduction">
     ...
  </task>

  <task name="empirical">
    <title>Some Empirical Questions</title>      <!-- Titel des Test-Fensters -->
    <text name="IntroductionText">
      <h2>Empirical Questions</h2>
      <p>Please fill in these empirical questions for statistical reasons</p>
    </text>

    <dropDownBox name="user_age" description="How old are you?" bottomPadding="40">
        <item selectable="false" showThisFirst="true">Please choose...</item>
        <item value="10">up to 10</item>
        <item value="20">11 - 20</item> <item value="30">21 - 30</item>
        <item value="40">31 - 40</item> <item value="50">over 50</item>
    </dropDownBox>

    <slider name="webuse_per_week"
            description="How many hours per week do you approximately use the Web?"
            majorTickSpacing="10" minorTickSpacing="2" minimum="0" maximum="80" default="0"
            paintLabels="true" bottomPadding="40" highlighted="false" />

    <button name="Next" text="Continue to First Task" enabled="true" highlighted="true">
      <startNextTask/>      <!-- Knopf startet die erste Aufgabe -->
    </button>

  </task>
...
```

Abb. 148: Ausschnitt einer XML-Datei zur Definition eines Testablaufes mit dem TEA UserTestTool.

Beim Start einer Studie wird die für den Testlauf gewählte XML-Beschreibungsdatei ausgewertet und die in ihr aufgeführten Schritte werden als Java-SWING-Oberfläche präsentiert. Abb. 150 zeigt links eine Ausgabeseite mit einer Einleitung für den Benutzer und rechts die Folgeseite mit unterschiedlichen Fragen zur Person. Die gezeigten Seiten wurden aus dem Beispielcode in Abb. 148 generiert.

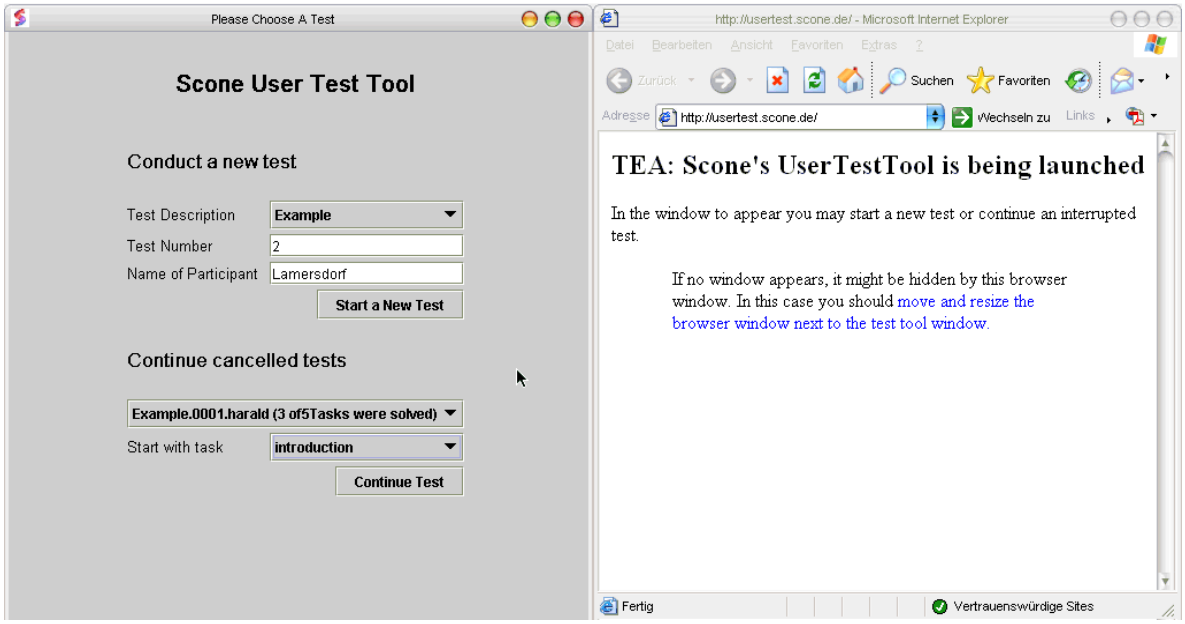


Abb. 149: Links der Start-Bildschirm des *UserTestTools*, rechts ist der Browser zu sehen.

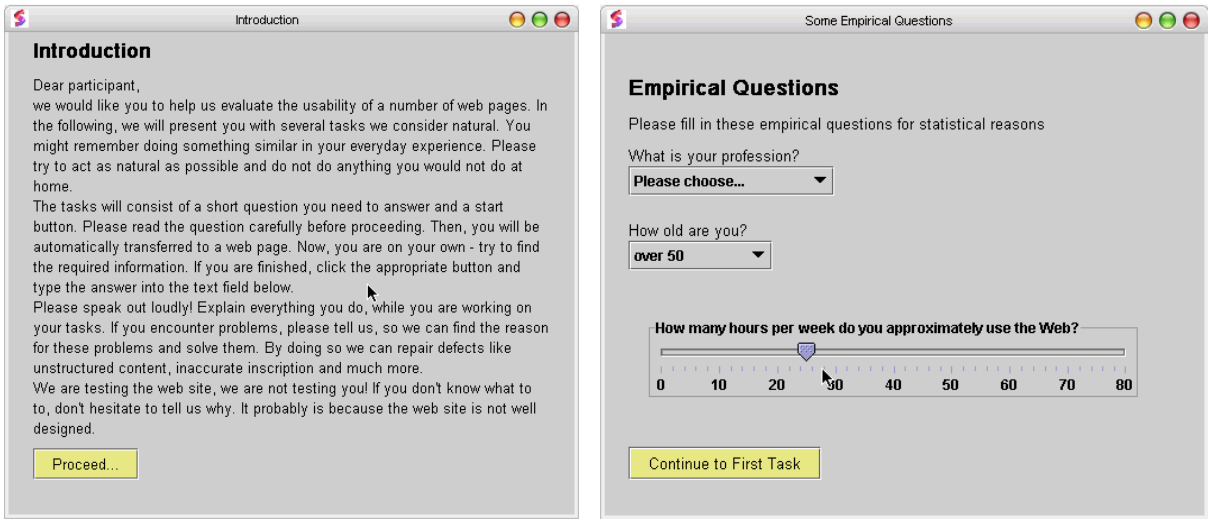


Abb. 150: Zwei Bildschirme des *Scone UserTestTools* in Aktion.

Die einzelnen Elemente der Formulare können mit Aktionen verknüpft werden. So führt der Klick auf den Knopf in der Abb. 150 links „Proceed…“ zur Anzeige des Fragebogens rechts. Es lassen sich Kommandos an den Browser schicken, um eine neue Seite im Browser darzustellen oder die Größe des Browsers anzupassen. Für die Steuerung des Browsers wird die *BrowserControl*-Komponente von Scone eingesetzt (s. Abschnitt 8.6.5.4). Darüber hinaus können Timer definiert werden, die nach einer bestimmten Zeit zu Ereignissen führen. So ist es unter anderem möglich, dem Benutzer Webseiten nur für eine vorgegebene Zeit zu zeigen oder die Bearbeitung von Aufgaben nach einer vorgegebenen Dauer abzubrechen.

```
<?xml version="1.0" encoding="UTF-8"?>
<userTestResult completedTasks="3" numberOfTasks="5" testPerson="harald"
          timeStamp="Sat Apr 24 15:35:27 GMT+01:00 2004" xmlFile="config/userTestDescriptions/Example.xml">
  <task name="introduction" timeStamp="1082817327733">
    <buttonClick action="click" name="nextButton" timeStamp="1082817330237"/>
  </task>
  <task name="Task 1" timeStamp="1083000428739">
    <buttonClick action="click" name="Next" timeStamp="1083000447336"/>
    <textFieldText action="" name="location" timeStamp="0"/>
  </task>
  <task name="Questions" timeStamp="1083000447356">
    <buttonClick action="click" name="endButton" timeStamp="1083000463769"/>
    <dropDownBoxText action="very ugly" name="webSiteDesign" timeStamp="1083000447386"/>
    <dropDownBoxValue action="0" name="webSiteDesign" timeStamp="1083000447386"/>
    <likertScaleSelection action="2" name="scr_simple" timeStamp="1083000453875"/>
    <likertScaleSelection action="6" name="scr_effective" timeStamp="1083000455287"/>
    <likertScaleSelection action="4" name="scr_quickly" timeStamp="1083000456298"/>
    <likertScaleSelection action="" name="scr_organization" timeStamp="0"/>
  </task>
  ...
</userTestResult>
```

Abb. 151: Beispiel-Ausgabe eines Tests als XML-Datei.

```
userTestResult      harald    config/userTestDescriptions/Example.xml   Sat Apr 24 15:35:27 GMT+01:00 2004  3/5
task                introduction                    1082817327733
buttonClick         nextButton      click           1082817330237
task                Task 1                          1083000428739
buttonClick         Next            click           1083000447336
textFieldText       location                        0
task                Questions                       1083000447356
buttonClick         endButton       click           1083000463769
dropDownBoxText     webSiteDesign   very ugly       1083000447386
dropDownBoxValue    webSiteDesign   0               1083000447386
likertScaleSelection scr_simple     2               1083000453875
likertScaleSelection scr_effective  6               1083000455287
likertScaleSelection scr_quickly    4               1083000456298
likertScaleSelection scr_organization               0
...
```

Abb. 152: Beispiel-Ausgabe eines Tests als CSV-Datei.

Die Aktionen des Benutzers mit dem Browser und mit dem TEA UserTestTool (Mausklicks, Formulareingaben etc.) werden protokolliert und in zwei Dateiformaten abgespeichert: Die *XML-Ausgabedatei* (s. Abb. 151) gibt die Strukturen der Aufgaben wieder und bietet die Möglichkeit, die Daten mit gängigen XML-Werkzeugen weiterzuverarbeiten. Die zweite Ausgabedatei im *CSV-Format*[328] (s. Abb. 152) ist hingegen für die Analyse mit Programmen wie SPSS und Microsoft Excel vorgesehen.

Mit dem *TEA UserTestTool* steht damit eine Software zur Verfügung, die die systematische Evaluation der mit Scone realisierten Prototypen unterstützt und Protokolldateien in einfach auszuwertenden Formaten zur Verfügung stellt.

Ein Bericht über das TEA UserTestTool und Scone zur prototypischen Realisierung und Evaluation von neuen Web-Erweiterungen wurde auf der ACM-Konferenz *CHI 2004* als Beitrag in der Session „Late Breaking Results“ präsentiert (Obendorf, Weinreich et al. 2004).

[328] CSV steht für *Comma Separated Values* und bezeichnet ein Textformat zum Austausch von Daten in Listenform (Shafranovich 2005).

8.6.6 Anwendung der Plug-in-Schnittstelle des Frameworks

Neue Entwicklungen auf Basis des Frameworks werden über die Plug-in-Schnittstelle realisiert. Jedes Scone-Plug-in benötigt eine Klasse zur Initialisierung des Plug-ins, die von der abstrakten Oberklasse „`scone.Plugin`“ erbt und sie implementiert. Die eigentliche Funktionalität der Erweiterung wird in der Regel in weiteren Klassen programmiert. Für jedes Plug-in sollte ein eigenes Java-Paket definiert werden.

Dem Entwickler stehen bei der Realisierung von Plug-ins alle zuvor beschriebenen Kern- und Hilfskomponenten des Frameworks zur Verfügung. Die kombinierte Nutzung dieser Komponenten erlaubt es, vergleichsweise schnell und einfach Werkzeuge zur Unterstützung der Navigation und Orientierung im Web als auch andere Web-Erweiterungen zu erstellen (s. Abschnitt 8.2ff). Die meisten Scone-Komponenten bieten mehrere Abstraktionsebenen, um den individuellen Anforderungen gerecht zu werden. Ein Plug-in kann beispielsweise mittels des Intermediary nicht nur auf den Bytestrom und die Protokollparameter zwischen Browser und Server zugreifen, HTML-Dokumente können auch als eine Folge von Objekten wie HTML-Tags, Links und Texte bearbeitet werden (s. Abschnitt 8.6.5.1). Eine dritte Abstraktionsebene bietet die NetObject-Komponente (s. Abschnitt 8.6.3): Sie stellt Meta-Informationen des Dokuments wie Sprache, Anzahl der Links und eine Zusammenfassung bereit (s. Abschnitt 8.6.5.3).

Die benötigten Funktionen und Komponenten werden mithilfe von „Scone-Requirements“ definiert. Sie stellen sicher, dass das Framework nur notwendigen Komponenten aktiviert, keine überflüssigen Berechnungen durchführt oder ungenutzte Datenobjekte erstellt.

Das folgende Beispiel in Abb. 153 zeigt die Klasse „`scone.examples.ExamplePlugin`“. Sie illustriert, wie man in einem Scone Plug-in den *Intermediary*, den *Robot* und das *AccessTracking* einsetzt.

Das Plug-in fordert in „`getRequirements()`“ folgende Funktionalität an: Benötigt wird das Parsen der HTML-Dokumente durch den *DocumentParser* („`PARSEDOCUMENT`“), wobei Links berücksichtigt und für sie eigene Objekte erstellt werden sollen („`CONSIDERLINKS`“). Zudem wird das *AccessTracking* von Scone aktiviert („`ACCESSTRACKING`“), das die Browser-Aktionen des Nutzers erfasst (Abschnitt 8.6.4). Mittels des *RessourceGenerators* („`RESOURCEGENERATOR`“) wird Scone zum Webserver für lokale Dateien. Der RAS-Server soll ebenfalls gestartet werden; die Methode „`getRasPort()`“ liefert den IP-Port zurück, auf dem der Dienst auf Verbindungsanfragen wartet (s. Abschnitt 8.6.5.1).

```
package scone.examples;
import scone.*;
import scone.netobjects.*;
import scone.proxy.*;
import scone.robot.*;
import scone.util.*;

// Das ExamplePlugin muss von der Klasse scone.Plugin erben.
public class ExamplePlugin extends Plugin{

  private scone.robot.Robot robot; // Referenz zum Robot

  // die Requirements geben an, welche der Komponenten benötigt
  // und welche ihrer Funktionen eingesetzt werden.
  public int getRequirements(){  // Requirements
     return PARSEDOCUMENT | CONSIDERLINKS | ACCESSTRACKING | RESOURCEGENERATOR | RAS;
  }

  // Der TCP-Port, auf dem der RAS-Server von Scone auf Anfragen warten soll.
  public int getRasPort() {
     return 8084;
  }

  // Die Initialisierung des Plguins.
  public void init(){
      // Hole eine Instanz des Crawlers.
      robot = scone.robot.Robot.instance();
      // Das Plug-in soll über Zugriffe (neue Access-Objekte) informiert werden.
      AccessCache.putObserver(this);
      // Ein MEG, um die vom Browser angefragten HTML-Dokumente zu Filtern.
      ExampleMEG em=new ExampleMeg(this);
      em.setup("Example Document Editor",HTDOCCONDITION,60);
      addMeg(em);
      // Ein Generator um lokale Daten bereit zu stellen
      GeneralResourceGenerator.addPath("/_example/ ", "resources/example");
  }

  // Diese Methode wird vom AccessCache aufgerufen, wenn ein neues Access-Objekt erstellt wird
  public void update(Observable o, Object arg) {

     // Wurde die Funktion vom AccessCache aufgerufen?
     if (o instanceof AccessCache) {
        // Es ist demnach ein Acess-Event
        AccessEvent e = (AccessEvent) arg;
        HtmlNode hNode = HtmlNodeCache.check(e.getNode());
        // Wurde auf ein HTML-Dokument zugegriffen?
        if (hNode != null) {
           // Stelle eine Suchanfrage an Goole für verwandte Seiten.
           String googleSuche = "http://www.google.de/search?q=related:"
                        + hNode.getSUri().getHost().getSUri().toString()
                        + "&hl=de&ie=ISO-8859-1&safe=off&btnG=Google-Suche&num=20";
           SimpleUri uri = new SimpleUri(googleSuche);
           // Starte den Robot, lade die angegebene URL, ohne weiteren Links zu folgen.
           rt = new RobotTask(uri, 1, RobotTask.ALL, this);
           robot.scan(rt);
        }
     }
  }

  // Diese Methode wird vom Robot bei neu bereitgestellten Seiten aufgerufen.
  public void robotNewPage(RobotHtmlNode robotHtmlNode, RobotTask robotTask) {
     ...
  }
  ...
}
```

Abb. 153: Beispiel-Code für ein Scone Plug-in.

Die Initialisierung des Plug-ins erfolgt in der Methode `init()`. In diesem Beispiel wird zuerst eine Instanz des Crawlers *„Scone Robot“* (s. Abschnitt 8.6.2) angefordert. Dann wird beim *AccessCache* das Plug-in-Objekt als „Observer“ angemeldet, sodass Nachrichten an das Objekt geschickt werden, sobald der Benutzer eine Seite im Browser lädt. Als Drittes wird ein MEG im Intermediary installiert, mit dem die übertragenen Objekte gefiltert werden können.

Der MEG ist in der hier nicht dargestellten Klasse „`scone.examples.ExampleMEG`“ implementiert. Dies könnte beispielsweise ein *DocumentEditor* sein (s. Abschnitt 8.6.1), der zusätzliche Elemente in die Seiten einfügt, die mithilfe des RAS-Systems weitere Daten innerhalb einer interaktiven Erweiterung von Scone abruft. Es lassen sich auch zusätzlich HTML-Elemente in die Dokumente einbinden oder von Scones *GeneralResourceGenerator* (z. B. JavaScript-Dateien oder Grafiken) per HTTP abrufen.

Im Beispiel fängt der Generator alle Browser-Anfragen ab, die den Pfad „`/_example/`“ in dem URI enthalten. Er wertet den restlichen Teil des URI aus und sucht im lokalen Dateisystem (im Unterverzeichnis „resources/example“ der Scone-Installation) nach der entsprechenden Datei. Das dort gefundene Objekt wird zum Browser übertragen.

Die Anmeldung des Plug-in-Objekts als Observer (s. o.) führt dazu, dass die Methode „`update()`“ vom AccessCache bei Änderungen an den Access-Objekten aufgerufen wird (s. Abschnitt 8.6.4). Sofern es sich um eine HTML-Seite handelt, wird sodann ein neuer Auftrag für den Robot erstellt. Im Beispiel stellt er per HTTP eine Anfrage an Google nach „verwandten“ Webseiten (vergl. Abschnitt 8.6.2). Der Robot ruft dann die Methode „`robotNewPage()`“ auf, sobald er ein neues Objekt heruntergeladen hat. Hier könnte beispielsweise ein Thumbnail der Seite erstellt werden.

8.6.6.1 Konzepte zur Konfiguration von Plug-ins

Plug-ins werden im grafischen Interface zur Konfiguration von Scone registriert und konfiguriert. Zur Registrierung muss der Name der Plug-in-Klasse angegeben werden. Das Plug-in erscheint dann in der Liste der bekannten Plug-ins (s. Abb. 154). Hier ist es durch Anklicken zu aktivieren bzw. zu deaktivieren (s. Abb. 155). Der Zustand wird durch das Icon vor dem Namen des Plug-ins kenntlich gemacht. Durch einen Doppelklick gelangt man in das Konfigurationsmenü des Scone Plug-ins, in dem sich seine Parameter konfigurieren lassen (s. Abb. 156).

Zu jedem Plug-in gehört eine Konfigurationsdatei entsprechenden Namens (beispielsweise „`hyperscout3.Plugin.xml`“ für das *HyperScout3-Plug-in*, s. Abb. 157), in der die Plug-in-Parameter gespeichert sind. Hierfür wurde ein XML-Schema definiert, das grundlegende Arten von Parametern unterstützt, z. B. Boole'sche Werte, Zeichenketten oder eine Auswahl aus einer vorgegebenen Liste. Aus dieser XML-Datei wird bei Aufruf des Plug-in-Konfigurationsmenüs automatisch ein grafisches Interface generiert, mit dessen Hilfe sich diese Parameter einstellen lassen (s. Abb. 156). Änderungen an den Werten werden danach in der XML-Datei persistiert. Das in Abb. 156 dargestellte Konfigurationsmenü korrespondiert mit der XML-Datei in Abb. 157.

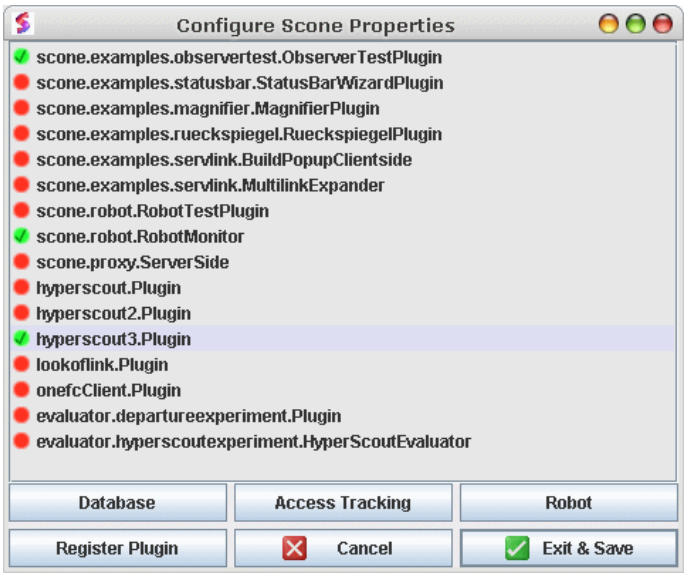


Abb. 154: Eine Liste aller registrierten Plug-ins im Scone-Framework.



Abb. 155: Grafische Schnittstelle zum Aktivieren und Konfigurieren des Plug-ins „hyperscout3.Plugin“.

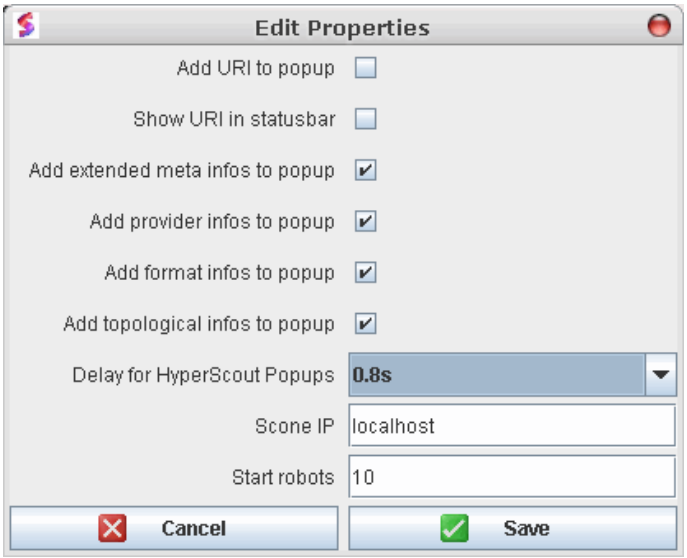


Abb. 156: Die grafische Schnittstelle zur Konfiguration der Parameter eines Plug-in.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
   <property name="Add URI to popup" type="boolean" value="false">
   </property>
   <property name="Show URI in statusbar" type="boolean" value="false">
   </property>
   . . .
   <property name="Delay for HyperScout Popups" type="choice" value="0.8s">
      <value>0.5s</value>  <value>0.8s</value>   <value>1.0s</value>
    </property>
   <property name="Scone IP" type="text" value="localhost">
   </property>
   <property name="Start robots" type="text" value="10">
   </property>
</properties>
```

Abb. 157: Ein Ausschnitt aus der Konfigurationsdatei „hyperscout3.Plugin.xml“.

Weitere Informationen zur Installation von Scone, zum Plug-in-Interface und zur Programmierung des Frameworks auf der Website des Projekts www.scone.de zu finden. Das dort erhältliche Scone-Paket beinhaltet zahlreiche weitere dokumentierte Beispiele. Darüber hinaus steht unter http://www.scone.de/doc/tutorial/intro.html ein Tutorium für die Programmierung von Scone bereit.

Die vollständige API-Dokumentation findet man unter: http://www.scone.de/doc/scone/.

8.7 Beispiele zum Einsatz des Frameworks: entwickelte Prototypen

Das Framework Scone hat sich in der Praxis im Rahmen zahlreicher Projekte bewährt. Die Anwendung blieb dabei nicht nur auf Erweiterungen zur Unterstützung der Navigation und Orientierung im Web beschränkt, es ist auch in Projekten zur Evaluation der Web-Nutzung eingesetzt worden. Die folgende Übersicht von sieben auf Basis des Frameworks erstellten Prototypen gibt einen Einblick in die Möglichkeiten und belegt seine Praxistauglichkeit. Scone wurde dabei nicht nur an der Universität Hamburg eingesetzt, sondern unter anderem auch von Forschern an der RWTH Aachen, der Universiteit Utrecht, der Università di Milano-Bicocca, der Universiteit Twente, der Technical University of Denmark in Lyngby und der University of South Australia.

Eine Vergleichsstudie unterschiedlicher Visualisierungstechniken für Link-Marker

In einem Forschungsprojekt, das der Autor dieser Arbeit gemeinsam mit Hartmut Obendorf durchführte, diente Scone zur Umsetzung und Evaluation unterschiedlicher *Visualisierungs- und Interaktionsformen für erweiterte Hyperlinks im Web*. Unter anderem wurde das Konzept der „*Links-on-Demand*“ (Bernstein 1996; Weinreich, Obendorf et al. 2001) erstmals für das Web implementiert und evaluiert (s. Abb. 158). Scone passte für die Tests Web-Dokumente „on-they-fly“ an, damit beliebige Websites einbezogen werden konnten. Zudem diente Scone zur Studiendurchführung und zur Protokollierung der Benutzeraktionen (s. Abschnitt 8.6.5.7). Dies gewährleistete eine hohe Realitätsnähe und eine gute Reproduzierbarkeit der Tests. Die Ergebnisse dieser Studien wurden auf der Konferenz WWW 2003 publiziert (Obendorf & Weinreich 2003).

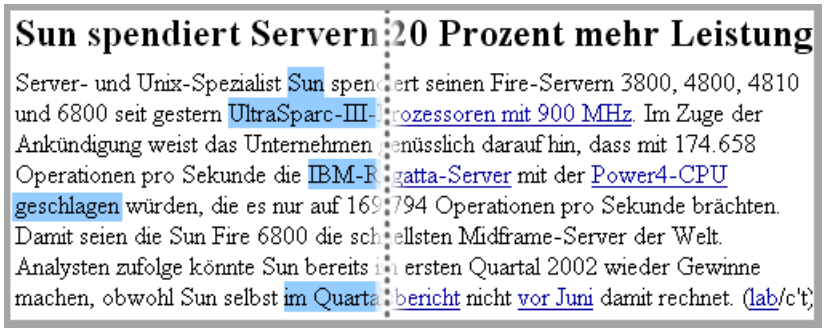


Abb. 158: Zwei unterschiedliche Visualisierungsformen für Links im Web.

Web-Rückspiegel

Der *Web-Rückspiegel* ist ein Prototyp, der in einem Fenster neben dem Browser eine Reihe von Thumbnails von Webseiten auflistet, die auf das gerade im Browser dargestellte Dokument verweisen. Hierdurch wird es möglich, im Web „rückwärts“ zu navigieren. Die Erweiterung soll als zusätzliche Navigations- und Orientierungshilfe dienen und den Zugriff auf inhaltlich verwandte Dokumente vereinfachen. Das Plug-in fordert von Google mittels des *Scone Robots* eine Liste von „Backlinks“ an. Die laut Google relevantesten dieser Seiten werden als Thumbnails dargestellt und lassen sich im Browser durch Anklicken öffnen (s. Abb. 146 auf Seite 352; Wollenweber 2002; Weinreich, Obendorf et al. 2004).

Navigation Visualizer

Der *Navigation Visualizer* wurde von Eelco Herder an der Universität Twente entwickelt (s. Abb. 159). Das System verwendet eine Kombination aus Data-Mining-Techniken, „*Dynamic Queries*“ (Ahlberg, Williamson et al. 1992) und unterschiedlichen Visualisierungsmethoden, um mit Scone erfasste Benutzungspfade im Web auszuwerten und Strukturen sichtbar zu machen (Herder 2003; Herder 2006).

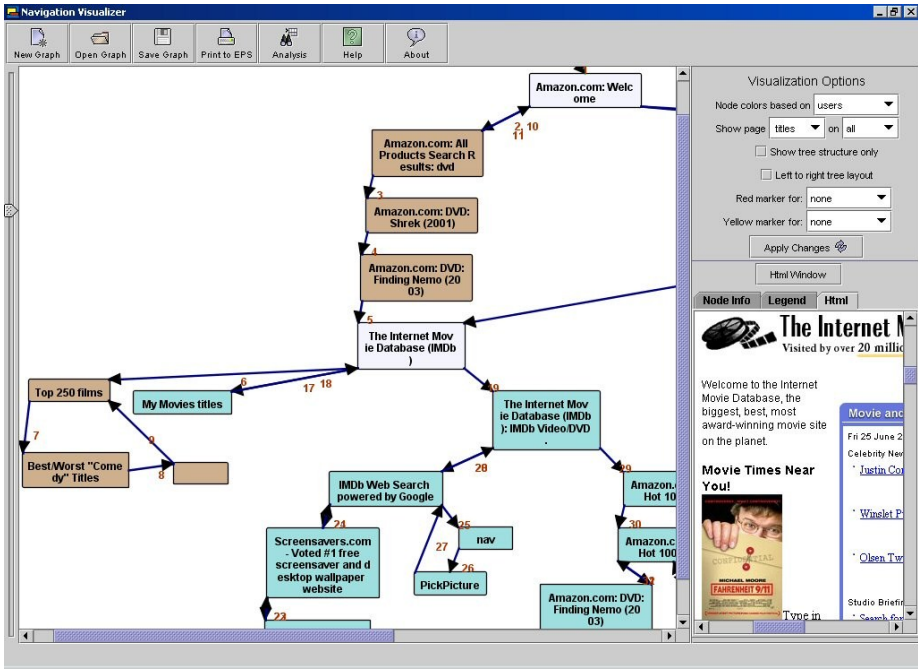


Abb. 159: Der *Navigation Visualizer* ist eine interaktive Analyseplattform für Benutzerpfade im Web.

Improving Trust in Wikipedia

Thomas Korsgaard von der Technical University of Denmark in Lyngby hat auf Basis von Scone das Kollaborationssystem *WRS* für Wikipedia realisiert, mit dem Leser jeden Artikel bewerten können, damit sie über die Beurteilungen anderer (ihnen bekannter) Leser eine bessere Einschätzung der Qualität von Artikeln in der Enzyklopädie erhalten (Korsgaard 2007). Seit Ende 2008 wird ein Nachfolgeprojekt an der Universität entwickelt, das ebenfalls Scone einsetzt.

SessionGraphs (SGA)

Das Projekt *SessionGraphs (SGA)*[329] von Matthias Mayer ist eine Navigationshilfe, die das Wiederfinden von Webseiten nach mehreren Tagen oder Wochen vereinfachen soll. Das Tool zeichnet hierfür in einem Fenster neben dem Browser dynamisch einen animierten Graphen der Benutzerpfade während des Surfens im Web auf (Abb. 160). Diese Graphen werden mit Attributen der Dokumente angereichert und nach dem Beenden einer Aufgabe oder einer Sitzung als eine Art „Icon“ bereitgestellt (Mayer 2008; Mayer 2009). Eine am HCI-Lab der University of Maryland durchgeführte Studie hat gezeigt, dass dieses Tool das Zurückkehren zu früher besuchten Seiten in vielen Fällen vereinfachen kann (Mayer & Bederson 2001). Das Scone-Framework wird in diesem Projekt als Schnittstelle zum Browser verwendet: Es überwacht die Benutzeraktionen, öffnet Browser-Fenster mit ausgewählten *SessionGraphs* und erstellt Thumbnails von Seiten.

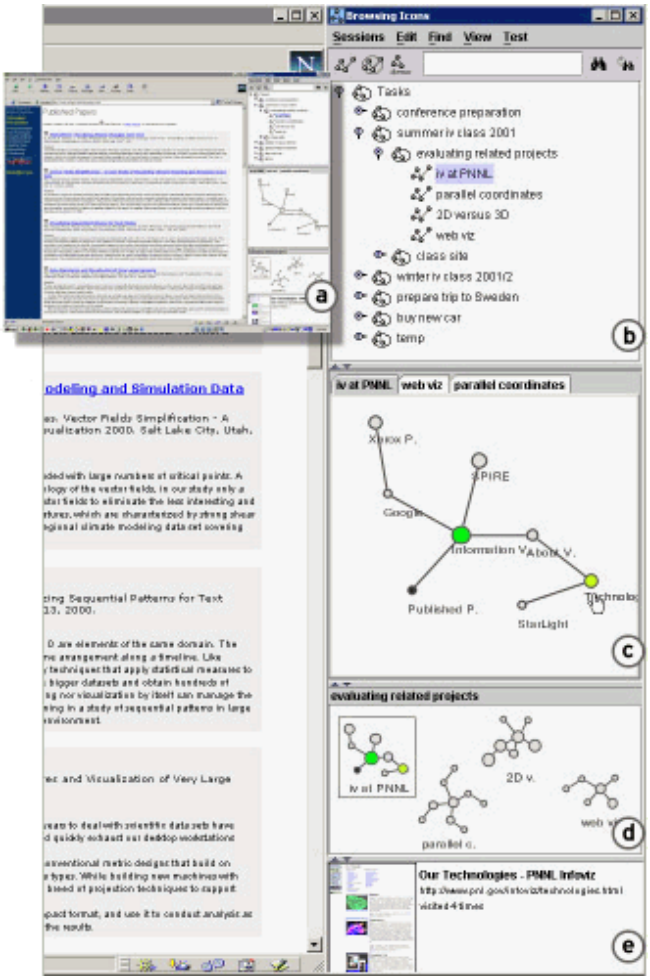


Abb. 160: Das *SessionGraphs-System.*

Collaborative Internet Experience

Frank Wollenweber hat mithilfe von *Scone* und dem *OneFC-Framework* (Baier 2005) das System *Collaborative Internet Experience* (kurz: *CoInternet*) entwickelt (s. Abb. 161). Es bietet Benutzern die Möglichkeit, „gemeinsam“ im Web zu surfen, indem sie Meinungen und Erfahrungen austauschen können. Zudem erlaubt es ihnen, Annotationen zu beliebigen Web-Ressourcen hinzuzufügen. Diese Informationen werden sowohl in einem eigenen Client neben dem Browser (s. Abb. 161, links) als auch direkt im Dokument bei Suchausgaben und Hyperlinks (s. Abb. 161, rechts) dargestellt (Baier, Weinreich et al. 2004; Wollenweber 2004).

[329] Das System hieß ursprünglich „BrowsingIcons“.

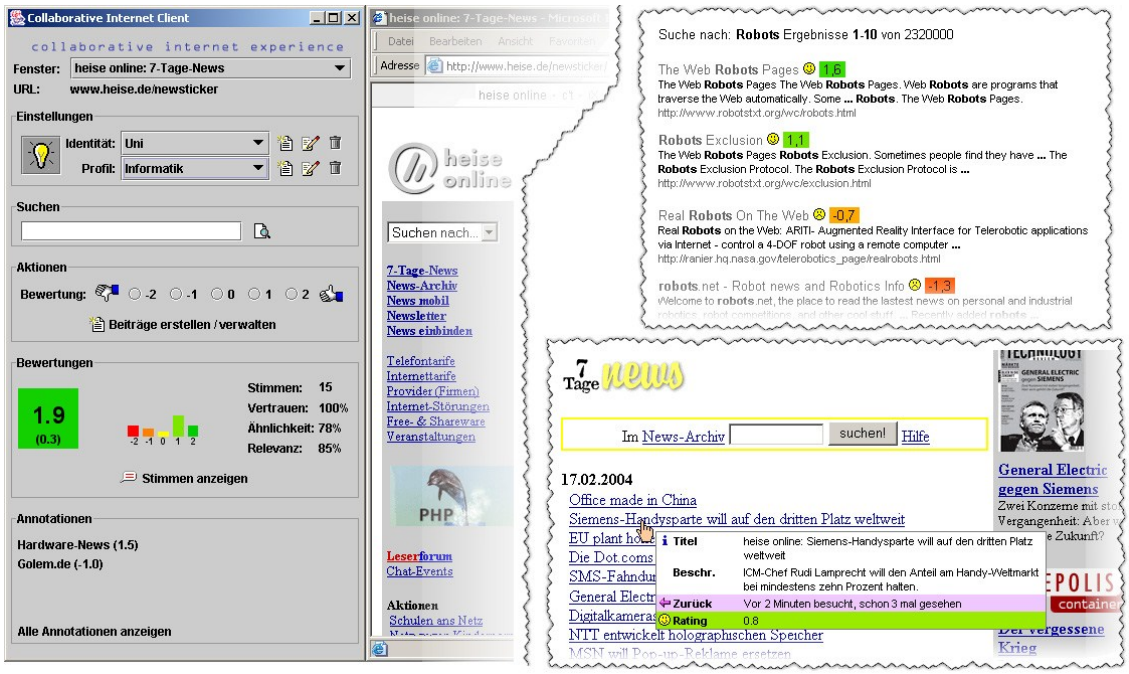


Abb. 161: Die wichtigsten Schnittstellenkomponenten des *CoInternet*-Systems: Links ist der Client zu sehen, rechts Beispiele für bewertete Suchausgaben und Link-Previews.

Eine Langzeitstudie zur Benutzung des Webs

Neben der Verwendung als Framework zur Erstellung von Forschungsprototypen wurde Scone im Rahmen einer *Langzeitstudie zur Benutzung des Webs* eingesetzt, bei der das Framework seine Stabilität unter Beweis stellte. Im Rahmen der Studie wurde die alltägliche Nutzung des Webs von 25 Teilnehmern aufgezeichnet, wobei sie während einer Studiendauer von 52 bis 195 Tagen (Mittelwert: 105 Tage) durchgängig mit Scone als Intermediary auf das Web zugriffen. Eine angepasste Version des Scone AccessTrackings zeichnete alle Navigationsaktionen der Teilnehmer mit dem Browser auf. Insgesamt konnten so fast 160.000 Aktionen registriert werden, was die Studie zu der seinerzeit umfangreichsten ihrer Art machte (vergl. Catledge & Pitkow 1995; Tauscher & Greenberg 1997; Cockburn & McKenzie 2001). Die Ergebnisse dieser Studie wurden u. a. in (Weinreich, Obendorf et al. 2006a; Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Weinreich, Obendorf et al. 2008) publiziert.

8.8 Möglichkeiten und Grenzen von Intermediary-Frameworks zur Erweiterung der Benutzungsschnittstelle des Webs

Das vorgestellte Framework wurde entwickelt, um Konzepte zur besseren Benutzbarkeit des Webs einfacher und effizienter realisieren zu können und um die Konzepte des HyperScout-Systems zum Zwecke der Evaluation zu implementieren. Beide Ziele wurden erreicht: Die im vorherigen Abschnitt 8.7 vorgestellten Projekte zeigen die erfolgreiche Anwendung des Frameworks, und unterschiedliche Studien haben belegt, dass Frameworks Entwicklungsprozesse beschleunigen und verbilligen können und sich die Softwarequalität potenziell er-

höht (Sneed & Jungmayr 2011). Zudem wurden alle vorgestellten HyperScout-Prototypen (siehe Kapitel 5 bis 7) auf Basis von Scone erstellt und evaluiert.

Es gibt Konzepte zur Erweiterung der Navigationsmöglichkeiten des Webs, die sich mit den für das Framework gewählten Techniken problemlos umsetzen lassen, und andere, für die sich das Framework weniger gut eignet. Folgende Einsatzbereiche bieten sich an:

- Die Konzepte von Scone sind prädestiniert für Erweiterungen des Webs, bei denen die Darstellung von Webseiten und die Interaktion mit ihnen optimiert werden. Dies können serverseitige Werkzeuge sein, die mithilfe des Intermediary und der Parser von Scone HTML-Dokumente für bestimmte Anforderungen anpassen (beispielsweise für mobile Geräte), sie mit zusätzlichen Navigationselementen anreichern oder neuartige Dienste integrieren.
- Entsprechend lassen sich auf Clientseite die vom Anwender zugegriffenen Web-Ressourcen benutzerspezifisch anpassen. Mithilfe des Crawlers können ergänzende Informationen zusammengestellt und eingefügt werden.
- Intermediary-basierte Systeme eignen sich ebenfalls besonders für die Entwicklung von webbasierten Kollaborationswerkzeugen. Dabei greifen alle Gruppenmitglieder über den Intermediary des Frameworks auf das Web zu. Scone stellt die notwendigen Komponenten für die Identifikation der Benutzer und ihrer Aktionen zur Verfügung.
- Konzepte, die eine softwaretechnische Systemerweiterung sowohl auf Client- als auch auf Workgroup- und/oder Serverseite erfordern, sind ebenfalls gut realisierbar, da Scone in jedem dieser drei Bereiche einsetzbar ist. Der Vorteil für Entwickler ist, dass sie jeweils dasselbe Framework mit derselben Schnittstellen nutzen können.
- Eine weitere besondere Stärke ist die Plattformunabhängigkeit der entwickelten Konzepte. Es lassen sich Erweiterungen entwickeln, die unverändert für zahlreiche Browser oder Server-Systeme eingesetzt werden können, auch wenn diese jeweils *inkompatible* oder *gar keine* geeigneten Erweiterungsschnittstellen bieten. Ein aktuelles Beispiel sind die Webbrowser von *Smartphones* und *Tablet PCs*, die in der Regel über keine APIs für *Browser-Add-ons* verfügen (s. Kapitel 9.3.4) und dennoch mit *Scone* erweiterterbar sind.

Die Grenzen des Frameworks ergeben sich weniger aus Defiziten der Komponenten von Scone, sondern leiten sich primär mit dem Gesamtkonzept des intermediary-basierten Frameworks, der eingesetzten Sprache *Java* und der daraus folgenden begrenzten softwaretechnischen Integrierbarkeit in Webbrowser und Server ab:

- Eine nahtlose Einbindung in die Bedienelemente des Browsers (z. B. in Form einer Toolbar) ist mit Scone (momentan) nicht möglich. Das ist insbesondere darauf zurückzuführen, dass die proprietären APIs der Browser den Einsatz spezifischer Programmiersprachen erfordern (beispielsweise XUL für Firefox oder C++/COM für den Internet

Explorer). Hierfür gibt es aber zahlreiche alternative Ansätze, die sich auch in Kombination mit Scone nutzen lassen (s. Abschnitt 8.9).

- Auch auf Serverseite sind der Integration Grenzen gesetzt. Zwar kann das Framework mittels des *ServerSide-Plug-ins* (s. Abschnitt 8.6.5.6) als erweiterte Kopie beliebiger Websites fungieren, aber eine technische Integration in einen Anwendungsserver ist nicht vorgesehen (s. Abschnitt 8.3.2). Eine auf Scone basierende Erweiterung hat daher in der Regel keinen direkten Zugriff auf interne Funktionen des Servers.

Trotz der technischen Einschränkungen ist Scone für viele Konzepte zur Erweiterung des Webs einsetzbar, die auf den ersten Blick eine engere Systemintegration benötigen. Dafür wird prototypisch eine Annäherung an die optimale Lösung realisiert, die sich für eine *Evaluation* des Systems mit Benutzern eignet. Beispielsweise können Bedienelemente in einem Fenster *neben* dem Browser (siehe z. B. Abschnitt 8.7, Projekt „CoInternet“) statt in der *Browser-Sidebar* dargestellt werden, wodurch sich ein vergleichbarer Zusammenhang mit der Benutzungsoberfläche des Browsers herstellen lässt.

Scone ist hingegen *nicht* für die Entwicklung kommerzieller Produkte vorgesehen. IBMs Lizenzbedingungen für das in Scone integrierte Framework WBI beschränken den Einsatz des Systems auf akademische und nicht-kommerzielle Projekte. Die Stärke von Scone liegt in der Möglichkeit, schnell experimentelle Erweiterungen in Java zu implementieren, um diese dann zu evaluieren, ohne dass Entwickler die proprietären und oft komplizierten APIs der vielfältigen heutigen Browser oder Server programmieren müssen.

8.9 Alternative Techniken zur Entwicklung von Browser-Erweiterungen

Es gibt mehrere Projekte, die mit ähnlichen Zielsetzungen wie Scone entwickelt wurden. Verwandte Projekte, die ebenfalls auf einem Intermediary bzw. Web-Proxy basieren, wurden bereits in Kapitel 8.4.3 vorgestellt. In den letzten Jahren werden intermediary-basierte Systeme aber vornehmlich für die Entwicklung von Web-Erweiterungen für *Workgroups* und auf *Serverseite* eingesetzt, zumal inzwischen Schnittstellen für *Browser-Extensions* verfügbar sind, die für Endanwender einfacher als ein Intermediary zu installieren sind. Sie erlauben ebenfalls die Modifikation der angezeigten Webseiten und eignen sich (mit Einschränkungen) zur Entwicklung vieler Systeme zur Verbesserung der Navigation und Orientierung im Web.

Das erste System, mit dem sich direkt das im Browser dargestellte Dokument manipulieren ließ, war das ***Arakne Environment***. Es wurde ab 1998 an der Universität Aarhus als *Open Collaborative Hypermedia System* für das Web entwickelt. Es war in Java geschrieben und verwendete eine Java-COM-Brücke für den Zugriff auf den Internet Explorer und das DOM des angezeigten Dokuments (s. Bouvin 2000); die technische Basis machte die Installation aber relativ aufwendig.

Ein 2005 veröffentlichtes Add-On für Firefox zur Entwicklung eigener Browser-Erweiterungen ist ***Greasemonkey***.[330] Es gestattet die Individualisierung von Webseiten mithilfe von *User Scripts*. Diese JavaScript-Programme sind an URIs oder Domain-Namen gebunden und werden jedes Mal ausgeführt, wenn eine entsprechende Seite im Browser geladen wird. Greasemonkey wird in der Regel für kleinere Erweiterungen eingesetzt, die das Erscheinungsbild und das Verhalten einzelner Seiten an spezifische Benutzerbedürfnisse anpassen,[331] es eignet sich mit Einschränkungen aber auch als Prototyping-Werkzeug für die Entwicklung neuer Browser-Add-Ons, zumal für die Installation eines *User Scripts* kein Neustart des Browsers erforderlich ist (Pilgrim 2005: 2). Mit Greasemonkey kann man aber *nicht* die Benutzungsschnittselle des Browsers ergänzen, also z. B. *Toolbar*, *Sidebar* oder *Navigationsbuttons* hinzufügen. Greasemonkey lässt sich mit Scone nur eingeschränkt vergleichen, da es nur JavaScript unterstützt und beispielsweise keine Komponenten zur Mehrbenutzerunterstützung, zur Persistierung von Daten und keinen Crawler bietet. Es sind an Greasemonkey angelehnte Erweiterungen für fast jeden anderen populären Browser[332] verfügbar, die aber alle nicht ganz die Funktionalität des Originals bieten.

Seit Mai 2010 bieten die Entwickler von Firefox das ***Firefox Add-on SDK*** an (anfangs *Jetpack SDK* genannt).[333] Dieses System ist für die Programmierung und das Debugging neuer Browser-Erweiterungen mit JavaScript direkt im Browser vorgesehen. Dafür integriert es den JavaScript-Debugger *Firebug*[334] und das JavaScript-Framework *jQuery*.[335] Der entscheidende Unterschied zu *Greasemonkey* ist, dass diese Erweiterungen auf das Browser-Fenster und Tabs angewendet werden und nicht nur für spezielle URIs vorgesehen sind. Scripte für das *Firefox Add-on SDK* können auf verschiedene Browser-Ereignisse reagieren, auf das Clipboard zugreifen, Tabs manipulieren und die Benutzungsschnittstelle des Browsers erweitern, beispielsweise neue Elemente im *Statusbereich* und eine Art *Sidebar* (genannt „*Slidebar*“) hinzufügen. Das SDK verwendet ein Sandbox-Konzept ähnlich dem von Java Applets, um das System des Anwenders vor schädlichen Funktionen in entsprechenden Erweiterungen zu schützen. Zusammen mit dem *Mozilla-Chromeless-Projekt*[336] lässt sich sogar die gesamte Benutzungsschnittstelle des Browsers neu gestalten. Im Vergleich mit *Scone* hat das *Firefox Add-On SDK* einige Einschränkungen: so gibt es keinen Crawler, nur sehr einfache Persistenzmechanismen und nur eingeschränkten Zugriff auf die History des Benutzers. Für die

---

[330] Die Erweiterung *Greasemonkey* findet man unter: https://addons.mozilla.org/de/firefox/addon/greasemonkey/.

[331] Ein Verzeichnis verfügbarer User Scripts für Greasemonkey befindet sich unter: http://userscripts.org/.

[332] Beispiele sind die Erweiterung *Trixie* (http://www.bhelpuri.net/Trixie/) für den Internet Explorer und *GreaseKit* für Safari (http://8-p.info/greasekit/).

[333] Die Homepage des Jetpack-Projektes ist: https://addons.mozilla.org/en-US/developers/.

[334] Firebug ist der JavaScript-Debugger für Firefox. Die Homepage des Projektes lautet: http://getfirebug.com/.

[335] JQuery ist eines von zahlreichen JavaScript-Frameworks, die nicht nur zahlreiche komfortable Funktionen zur Manipulation des DOM und zum Event-Handling bieten, sondern als Abstraktionsebene die Unterschiede verschiedener Browser kapselt. http://jquery.com/.

[336] Das Chromeless-Projekt findet man unter der Adresse: http://mozillalabs.com/chromeless/.

Realisierung der HyperScout-Prototypen eignet sich Jetpack damit nur eingeschränkt. Die Integration in den Browser und die Möglichkeiten zur Erweiterung der Benutzungsschnittstelle des Browsers machen das System aber zu einer hervorragenden Ergänzung für das Framework Scone.

8.10 Fazit

In dieser Arbeit wurde ein Konzept für ein Framework entwickelt und realisiert,[337] das nicht nur von praktischem Nutzen für viele Forscher und Entwickler war und ist, sondern dass das sich auch durch zahlreiche konzeptionelle Innovationen auszeichnet. Zum einen ist das Gesamtsystem bis heute alleinstehend in seinen Möglichkeiten zur plattformunabhängigen Erweiterung von Webbrowsern und Servern. Darüber hinaus weisen auch die Komponenten von *Scone* zahlreiche Neuerungen auf. Beispiele für innovative Lösungen findet man in jeder der Kernkomponenten:

- Für den *Intermediary* ist ein neues Konzept zur internen Verarbeitung von Daten entwickelt worden, das IBM in die „offizielle" WBI-Version übernommen hat: Die *Object-Streams* ermöglichen den Transfer beliebiger Objekt- und Datentypen innerhalb von Intermediary-Architekturen und erhöhen die Effizienz bei der Verkettung mehrerer Filter innerhalb des Systems (s. Abschnitt 8.6.1.1).
- Das RAS-System hebt die Beschränkung von Intermediaries auf eine Änderung der Webseiten *während* des HTTP-Transfers auf. Die Komponente erlaubt es, Seiten auch *nach* der Übertragung mithilfe einer Socket-Verbindung, LiveScript und JavaScript über das DOM im Browser zu manipulieren. Die Möglichkeiten gehen dabei über die von *Ajax* hinaus, da das RAS-System auch einen *Push-Mechanismus* bietet, mit dem sich Kommandos zu beliebigen Zeitpunkten *vom Intermediary* zum Browser senden lassen.
- Der *Scone Robot* ist eine Basis für persönliche Web-Crawler unterschiedlichster Anforderungen. Die Parametrisierbarkeit von Aufträgen an den Crawler, die regelbasierte Steuerbarkeit und das *Classifier-Filter-Konzept* für Dokumente und Hyperlinks machen die Flexibilität des Systems aus (s. Abschnitt 8.6.2).
- Die *Scone NetObjects* sind eine leichtgewichtige Komponente für die Persistierung von Web-Ressourcen, Strukturinformationen und Benutzungsdaten. Sie ist aufgrund des Caching-Konzepts sehr performant und dennoch leicht an die Anforderungen neuer Plug-ins anpassbar (s. Abschnitt 8.6.3).
- Mit dem *Scone AccessTracking* lassen sich die Navigationsaktionen von Benutzern *entfernt* detailliert erfassen, ohne dass zusätzliche Software auf dem PC des Anwenders installiert

[337] Das vorgestellte Framework *Scone* wurde vom Autor dieser Arbeit konzipiert und in wesentlichen Teilen auch implementiert. An der Implementation waren weiterhin folgende Studenten beteiligt: Volkert Jürgens, Torsten Haß, Frank Wollenweber, Björn Stephan. Wichtiges Feedback für die Weiterentwicklung des Frameworks kam von Hartmut Obendorf, Matthias Mayer, Eelco Herder und einigen weiteren Anwendern.

werden muss (s. Abschnitt 8.6.4). Die verwendete Technik ist nahezu plattformunabhängig und an die Möglichkeiten und Eigenheiten weiterer Browser anpassbar. Einige Projekte verwenden ähnliche Techniken, um spezifische Informationen über Benutzer und ihre Aktionen zu ermitteln (s. Rajamony & Elnozahy 2001; Atterer, Wnuk et al. 2006), ohne aber die Flexibilität und Erweiterbarkeit des Scone *AccessTrackings* zu bieten.

Auch wenn *Scone* bei weitem nicht die Popularität von browserbasierterten Erweiterungskonzepten wie *Greasemonkey* (s. Abschnitt 8.9) erreicht hat, so hat es sich doch in einer Reihe verschiedenartiger Projekte bewährt (s. Abschnitt 8.7). Es bietet einen großen Funktionsumfang, ist gut dokumentiert und läuft stabil. Die Arbeit mit Scone hat zudem direkt und indirekt zu zahlreichen wissenschaftlichen Veröffentlichungen geführt (Weinreich & Lamersdorf 2000; Weinreich, Obendorf et al. 2001; Obendorf & Weinreich 2003; Weinreich, Buchmann et al. 2003; Baier, Weinreich et al. 2004; Obendorf, Weinreich et al. 2004; Weinreich, Obendorf et al. 2004; Weinreich, Obendorf et al. 2006a; Weinreich, Obendorf et al. 2006b). Insgesamt wurden die somit am Anfang der Entwicklung des Frameworks *Scone* gesetzten Ziele mehr als erreicht.

9 Resümee, Diskussion und Ausblick

Als um die 1990er Jahre die Grundsteine des Webs gelegt wurden, war weder absehbar, welch universelle Plattform es einmal darstellen, noch welch zentrale Bedeutung es in unserer Gesellschaft haben würde. Viele Konzepte des Webs waren daher nicht für die heutigen Anwendungsgebiete ausgelegt. Das wichtigste Schnittstellenelement, der Hyperlink, ist als einfacher assoziativer Verweis auf andere inhaltlich verwandte Ressourcen konzipiert worden. Heute erlauben komplexe Skripte vielfältige Interaktionsformen auf Webseiten, und Hyperlinks werden für alle möglichen Arten von Verknüpfungen und Aktionen verwendet. Die visuelle Darstellung von Hyperlinks für den Benutzer blieb dennoch in Browsern seit den Anfängen des Webs nahezu unverändert. Das daraus folgende Informationsdefizit der Anwender bei der Link-Navigation führt häufig zu Missverständnissen und Problemen bei der Web-Nutzung.

In dieser Arbeit wurden Erkenntnisse aus den Bereichen Web-Usability und Hypertext herangezogen, um die wesentlichen Faktoren für die Benutzbarkeit von verteilten Informationssystemen mit assoziativen Verknüpfungen zu identifizieren. Auf Basis systematischer Klassifikationen der wesentlichen Aspekte der Link-Benutzungsschnittstelle wurden neue Konzepte zur Reduktion des Informationsdefizits bei der Orientierung und Navigation in verteilten Informationssystemen entwickelt. Die für die Nutzung im Web am besten geeigneten Konzepte sind zum Zwecke der Benutzbarkeitsevaluation prototypisch implementiert worden. Die durchgeführten Studien zur Benutzbarkeit von Hyperlinks erlaubten die Optimierung der entwickelten Konzepte und lieferten neue, zum Teil grundlegende Erkenntnisse über die Herausforderungen der Link-Navigation in verteilten Informationssystemen und die Möglichkeiten ihrer Reduktion.

Der praktische Einsatz der entwickelten Konzepte im Web erfordert Protokoll- und Sprachерweiterungen, die eine effiziente Übertragung der zusätzlich benötigten Link-Daten gewährleisten. Die entwickelten Erweiterungen sind plattformunabhängig und für viele unterschiedliche Konzepte zur Reduktion von Benutzbarkeitsbarrieren bei der Link-Navigation geeignet.

Die Implementation von Web-Erweiterungen, wie die im Rahmen dieser Arbeit realisierten Konzepte, stellt zahlreiche Herausforderungen an die Entwickler. Ein neu gestaltetes Web-Prototyping-Framework vereinfacht die Implementation entsprechender Erweiterungen für beliebige Webbrowser und Server. Es eignet sich für die Realisierung vielfältiger Konzepte zur Unterstützung der Orientierung und Navigation im Web und dient ebenfalls zur Umsetzung und Evaluation der in dieser Arbeit entwickelten Ansätze.

9.1 Beiträge der Arbeit

Die wissenschaftlichen Beiträge dieser Arbeit bauen aufeinander auf und lassen sich so mehreren Ebenen zuordnen. Auf **konzeptioneller Ebene** reicht das Spektrum von Analysen der Benutzbarkeitsfaktoren bei der Navigation in verteilten Hypertext-Informationssystemen über Untersuchungen zur Entwicklung und Anwendung der Link-Benutzungsschnittstelle bis hin zu konkreten Konzepten für die Erweiterung dieser Schnittstelle zur Reduktion ihrer Benutzbarkeitsdefizite. Beiträge auf **praktischer Ebene** sind unter anderem die prototypische Realisierung und die Benutzbarkeitsevaluation der entwickelten Konzepte. Sie gestatten neue Einblicke in das Link-Navigationsverhalten von Benutzern in verteilten Informationssystemen. Lösungen für die praktische Einführung der neuen Konzepte im Web sowie ein flexibles Prototyping-Framework für Web-Erweiterungen wurden auf **softwaretechnischer Ebene** eingebracht.

Einige wesentliche wissenschaftliche Beiträge dieser Arbeit sind im Detail:

- Ausgehend von Analysen zur Benutzbarkeit von verteilten Hypertext-Informationssystemen und einer Langzeitstudie zur Benutzung des Webs wurden grundlegende Probleme beim Umgang mit großen verteilten Informationssystemen und die besondere **Rolle der Benutzungsschnittstelle von Hyperlinks** für die Ergonomie dieser Systeme ermittelt (siehe Kapitel 3).
- Es wurde eine **Spezifikation der implizit vorhandenen**, aber in der Regel für die Benutzer nicht zugänglichen **Link-Informationen im Web entwickelt** (Kapitel 4.5). Dazu sind Attribute und Charakteristika von Ressourcen und Links im Web analysiert worden. Die gewonnene Spezifikation erklärt systematisch alle für die Link-Navigation in verteilten Informationssystemen relevanten – und dennoch bisher oft vernachlässigten – Informationen und kann als Ausgangspunkt für die Entwicklung neuer und besserer Navigationswerkzeuge dienen.
- Auf Basis früherer Resultate aus den Bereichen der Hypertext- und Web-Forschung erfolgte erstmals eine **systematische Klassifikation** der Visualisierungstechniken für Link-Anker (Kapitel 5.2) und der Darstellungsmöglichkeiten für (ergänzende) Link-Informationen (Kapitel 5.3). Diese Klassifikationen geben ein Gesamtbild der verfügbaren **Schnittstellenkonzepte für Hyperlinks** und ihrer jeweiligen Stärken und Schwächen. Die Ergebnisse lassen sich nicht nur für grafische Webbrowser auf Desktop-Oberflächen, sondern auch für andere Schnittstellenkonzepte heranziehen (s. Kapitel 9.3.2).
- Die durchgeführten Studien waren die Basis für die Entwicklung von Konzepten, mit denen sich **automatisch** wesentliche, bisher meist unzugängliche **Informationen** über die dargestellten Links und Zielobjekte **auf konsistente Weise darstellen** lassen. Die für grafische Webbrowser in Desktop-Oberflächen am besten geeigneten Konzepte wurden in einem System mit dem Namen *HyperScout* zusammengefasst, prototypisch realisiert und systematisch evaluiert (Kapitel 5.5ff). Die Konzepte tragen zur verbesserten Benutzbarkeit

und Konsistenz von Hyperlinks im Web bei und gewährleisten, dass Browser bei der Darstellung von Link-Ankern in Webseiten erstmals wesentliche Kriterien der ISO-9241-151:2008-Norm zur Benutzbarkeit von webbasierten Systemen erfüllen (siehe Kapitel 5.1.4 und 9.2; vergl. ISO9241-151 2008, Sektion 8 und 9).

- Zwei Benutzbarkeitsstudien mithilfe von Prototypen erlaubten eine Bewertung und Optimierung der entwickelten Konzepte und lieferten neue **Einblicke in die Problematik der Link-Navigation** (s. Kapitel 6). Die erzielten Ergebnisse können dazu beitragen, die Navigation im Web und in verteilten Informationssystemen zu vereinfachen.
- Eine **Erweiterung des HTTP-Protokolls** und ein neues **XHTML-Modul** dienen zur effizienten Übertragung zusätzlicher Daten zu allen Links einer Webseite. Die Konzepte sind kompatibel mit dem existierenden Web und gut skalierbar. Aufgrund des generischen Ansatzes und der Plattformunabhängigkeit sind beide Erweiterungen für viele andere Konzepte zur Navigationsunterstützung im Web prädestiniert (s. Kapitel 7.3). Die zusätzlich bereitgestellten Link-Daten können gleichzeitig zur Entwicklung neuer barrierefreier Web-Clients beitragen (Kapitel 9.3.3).
- Schließlich wurden neue Konzepte zur **prototypischen Implementation von Web-Erweiterungen** entwickelt (Kapitel 8). Die Konzepte wurden in Form eines Frameworks realisiert, das sich aufgrund seiner Offenheit, Flexibilität und komponentenbasierten Architektur für die Implementation vieler unterschiedlicher Erweiterungen des Webs eignet. Die ausführliche Dokumentation und die Offenlegung des Quellcodes unter www.scone.de machen es allgemein zugänglich, sodass es auch im Rahmen zahlreicher anderer wissenschaftlicher Projekte eingesetzt werden kann und bereits mehrfach verwendet wurde (s. Kapitel 8.7).

Nicht zuletzt führte diese Arbeit zu zahlreichen wissenschaftlichen Publikationen in den Bereichen Hypertext, Web-Technologien und Web-Usability. Hervorzuheben sind neben den Veröffentlichungen zu *HyperScout* (Weinreich & Lamersdorf 2000; Weinreich, Obendorf et al. 2003; Weinreich, Obendorf et al. 2004) und *Scone* (Weinreich, Buchmann et al. 2003; Obendorf, Weinreich et al. 2004; Weinreich, Obendorf et al. 2006a) die Publikationen der umfangreichen Ergebnisse einer mehrwöchigen Studie zur täglichen Anwendung des Webs mit 25 Teilnehmern, in der mithilfe des *Scone-Frameworks* die Web-Benutzeraktionen aufgezeichnet wurden (u. a. Weinreich, Obendorf et al. 2006a; Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007; Weinreich, Obendorf et al. 2008). Die Ergebnisse untermauern gleichzeitig die Bedeutung von Hyperlinks für die Benutzung des Webs (vergl. Kapitel 2.2.4 und 3.3.1).

Darüber hinaus erhielten vier Veröffentlichungen des Autors **Auszeichnungen:**

- (Weinreich, Obendorf et al. 2006b): „*Best Student Paper Award*“ bei der *15th International World Wide Web Conference 2006*, 13 % Annahmequote bei der Konferenz,

- (Weinreich, Obendorf et al. 2006c): „*Best Paper Award*“ bei der Konferenz *Mensch und Computer 2006*,
- (Obendorf, Weinreich et al. 2007): „*Honorable Mention Paper*“ bei der *25th Conference on Human Factors in Computing Systems (CHI 2007)*,
- (Weinreich, Obendorf et al. 2003): „*Herausragender Beitrag*“ bei der Konferenz *Mensch und Computer 2003*.

In internationalen **Journals** wurden zwei weitere Berichte aufgenommen:

- (Weinreich & Lamersdorf 2000) in „*International Journal of Computer and Telecommunications Networking (6)2000*“,
- (Weinreich, Obendorf et al. 2008) in „*ACM Transactions on the Web (2)2008*“.

9.2 Diskussion

Im Folgenden werden die aktuellen Entwicklungen in Bezug auf die Benutzungsschnittstelle und Benutzbarkeit des Webs diskutiert. Die meisten Neuerungen der letzten Jahre beziehen sich auf die Möglichkeiten für interaktive Web-Anwendungen, es gibt jedoch nur wenige Verbesserungen bezüglich der Benutzbarkeit der Browser und keine wesentlichen Änderungen an der Benutzungsschnittstelle von Hyperlinks (s. Abschnitt 9.2.1). Abschnitt 9.2.2 befasst sich mit der Frage, warum scheinbar der Fortschritt in diesem Bereich stagniert und stellt Möglichkeiten zur kurzfristigen praktischen Einführung der in dieser Arbeit entwickelten Konzepte vor. In Abschnitt 9.2.3 wird anhand aktueller Browser-Add-Ons erörtert, welche Teile der entwickelten Konzepte bereits in der Praxis einsetzbar sind und welche Defizite noch bestehen.

9.2.1 Diskussion aktueller Entwicklungen

Seit der Publikation des ersten populären Vertreters „Mosaic“ (s. Anhang B.15) sind Webbrowser wesentlich komplexer und leistungsfähiger geworden, und zahlreiche neue Konzepte haben Einzug in das Web gehalten. Wenn man von den vielen neuen Anwendungsfeldern durch Online-Dienste (s. Kapitel 2.2.3ff) absieht und die Entwicklung der Browser-Software analysiert, so fällt auf, dass die meisten Neuerungen bei den technischen Möglichkeiten der Browser zu finden sind.

Die Browser-Hersteller haben in den letzten Jahren intensiv an der Ausführungsgeschwindigkeit und dem Funktionsumfang der in Webseiten eingesetzten interpretierten Programmiersprache *JavaScript* gearbeitet. Damit noch ausgereiftere interaktive Online-Anwendungen im Browser realisiert werden können, wurde beispielsweise die Manipulierbarkeit der angezeigten Seiten über das DOM erweitert und beschleunigt. Zudem werden mit jeder neuen Browser-Version zusätzliche Möglichkeiten in JavaScript eingeführt. Hierzu gehört eine bessere Unterstützung von Mobilgeräten mit berührungsempfindlichem Bildschirm

durch neue Sprachelemente für Multi-Touch-Gesten,[338] ein vereinfachter Zugriff auf die History des Browsers, wovon insbesondere Ajax-Anwendungen (s. Kapitel 2.2.5) profitieren, und neue Funktionen zur Ausgabe von 2D- und 3D-Grafiken im Browser-Fenster.

Eine weitere Neuerung ist HTML5 (s. Abschnitt 7.5.2). Obwohl es sich noch in der Entwicklung[339] befindet, wird es bereits in weiten Teilen von aktuellen Browsern unterstützt. HTML5 eignet sich ebenfalls besser für interaktive Webseiten, da es unter anderem den Zugriff auf die Elemente der Seite über das DOM genauer spezifiziert und vereinheitlicht.[340] Es erlaubt zudem das Speichern von Daten in einer lokalen Browser-Datenbank, wodurch sich der Umgang mit datenintensiven Online-Applikationen beschleunigen lässt (Benson, Marcus et al. 2010). HTML5 bietet darüber hinaus neue Ausdrücke zur Formular-Definition, die zur Vereinfachung bei der Dateneingabe (beispielsweise von E-Mail-Adressen oder numerischen Werten) beitragen (Münz & Gull 2010). Eine bereits regelmäßig genutzte Möglichkeit von HTML5 ist die direkte Einbindung von multimedialen Inhalten wie Audio- und Video-Dateien in Webseiten (ohne die Verwendung von Plug-ins wie *Adobe Flash*, siehe: Hickson 2011).

Die Sprachelemente zur Definition von *assoziativen Hyperlinks* mittels des Anker-Elements wurden in HTML5 gegenüber den Vorgängern nur geringfügig verändert. Einige bisher ungenutzte und von keinem verbreiteten Browser unterstützte Anker-Attribute[341] fallen weg, ergänzt wurde dafür das Attribut „media“ (Münz & Gull 2010), das als *Media Query*[342] eine „Empfehlung“ für den Medientyp bzw. die Geräteklassen gibt, für die das Zielobjekt optimiert wurde, z. B. für ein Smartphone oder die Sprachausgabe (s. Abschnitt 4.4.2). Für das Attribut „rel“, das ursprünglich den Link-Typ spezifizierte, findet man in HTML5 zahlreiche neue Vorschläge für gültige Attributwerte, die aber vornehmlich technischer Natur sind und Anweisungen für Browser und Suchmaschinen enthalten.

Die Spezifikation für *Cascading Style Sheets CSS3* weist eine Reihe von Ausdrücken für die Gestaltung von Link-Markern auf (s. Abschnitt 4.4.4). Die neuen Möglichkeiten werden bisher aber nur in Teilen von aktuellen Browsern unterstützt, und das obwohl die letzte Version des *CSS3 Hyperlink Presentation Module* bereits 2004 veröffentlicht wurde (Çelik, Bos et al. 2004).

---

[338] Die Entwicklung der sogenannten „Touch-Events“ wird aktuell insbesondere von Apple mit dem Safari-Browser auf dem iPhone und dem iPad vorangetrieben. Eine Übersicht der aktuellen Möglichkeiten bietet: http://quirksmode.org/mobile/tableTouch.html.

[339] Der endgültige Status „Recommendation“ für HTML5 und damit verbundene Sprachen ist für Dezember 2014 vorgesehen. Siehe: http://www.w3.org/2007/03/HTML-WG-charter.html.

[340] Insbesondere bei fehlerhaftem HTML-Code, der im Web eher die Regel als die Ausnahme ist (s. Kapitel 8.6.5.2), verhalten sich Browser oft in unvorhersehbarer und individueller Weise. Diese Problematik soll durch HTML5 vermieden werden, um die Entwicklung von interaktiven Web-Anwendungen zu vereinfachen.

[341] Dies sind die Attribute „`charset`“, „`coords`“, „`name`“, „`rev`“ und „`shape`“ (Münz & Gull 2010).

[342] *Media Queries* wurden im Zusammenhang mit CSS entwickelt, um medien- und geräteabhängige Stile definieren zu können (Lie, Çelik et al. 2010).

In Bezug auf die Benutzungsschnittstelle der Browser lassen sich in den letzten Jahren zwei wesentliche Tendenzen beobachten: Erstens wird die Schnittstelle sukzessive minimiert und rückt gegenüber dem Dokumentbereich zunehmend in den Hintergrund. Beispielsweise zeigen die meisten modernen Browser oben keine Pull-down-Menüs mehr an (s. Abb. 162 oben), alle Bedienelemente werden in einer Zeile mit der aktuellen Adresse dargestellt, und statt einer Statusleiste findet man den URI des Link-Zieles in einem kleinen Zusatzbereich, der bei Bedarf eingeblendet wird (s. Abb. 162, unten links). Zweitens werden Web-Dienste zunehmend fest in Browser integriert. So kann man bei den meisten Browsern Begriffe für die globale Suche direkt in die Adresszeile eingeben, und einige Browser wie Google Chrome setzen für die Darstellung von anderen Formaten als HTML, wie RSS-Feeds, PDF- und Office-Dokumente, Online-Dienste ein. Dies führt zu einer engeren Verzahnung des Webs mit Desktop-Anwendungen. Gerade die Firma Google treibt diese Entwicklung voran, um so die eigenen Angebote einfacher zugänglich und attraktiver zu machen. Beim *Google Chrome OS*[343] sollen letztendlich alle Office-Anwendungen im Browser laufen und Daten in der *Cloud*[344] gespeichert werden, sodass Anwender kaum noch lokal installierte Programme benötigen. Auf diese Weise entwickelt sich im Browser parallel zur Desktop-Oberfläche eine eigene Plattform mit spezifischen Potenzialen und Grenzen.

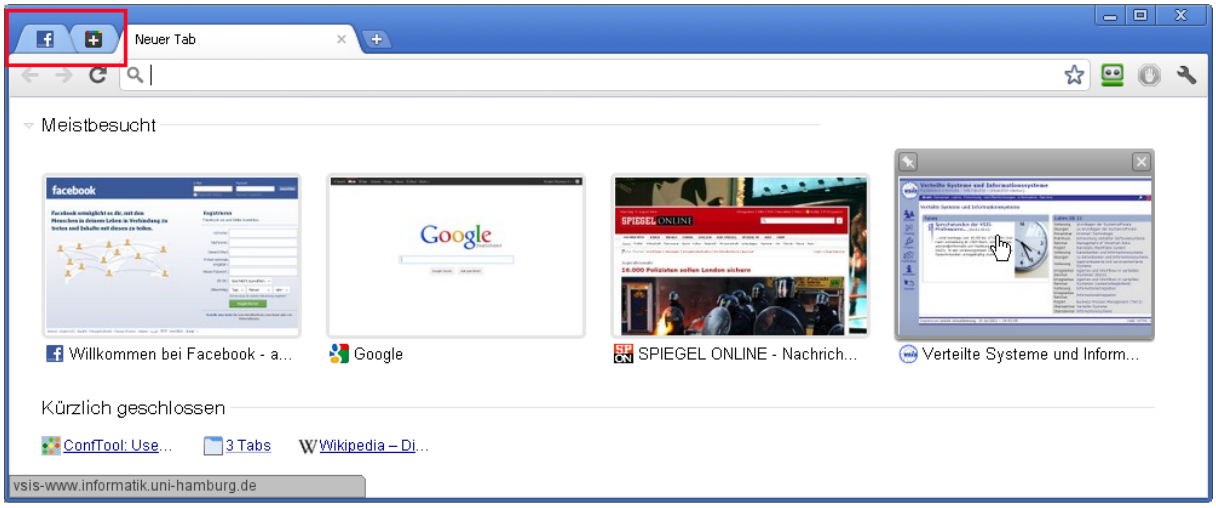


Abb. 162: „App Tabs“ und die Übersicht der meistbesuchten Seiten in Google Chrome 14.

Betrachtet man die Änderungen an Browsern in Bezug auf die Nutzung des Webs als *Informationssystem*, so sind die Verbesserungen überschaubar. Signifikate Auswirkungen auf das Navigationsverhalten der Anwender können die seit einigen Jahren in allen grafischen Browsern verfügbaren Browser-Tabs (Reiter) haben, wie mehrere Studien zeigen (Obendorf,

[343] Das Google Chrome OS (Projektname: „Chromium OS“) wurde Ende 2009 als Open Source Projekt von Google freigegeben. Mehr Informationen findet man unter: http://www.chromium.org/chromium-os.

[344] Der Begriff *Cloud Computing* bezeichnet IT-Infrastrukturen, bei denen Daten und Dienste online angeboten werden und bei Bedarf dynamisch den Benutzern zur Verfügung stehen. Auf diese Weise soll Speicherplatz und Rechenkapazität optimal ausgelastet werden. Abgerechnet werden diese Dienste in der Regel nach Nutzungsdauer und -intensität. Das Ganze geschieht für den Anwender auf transparente Weise, wie in einer „Wolke“ verhüllt (Armbrust, Fox et al. 2009).

Weinreich et al. 2007; Dubroy & Balakrishnan 2010; Huang & White 2010; Zhang & Zhao 2011), weil sie beispielsweise für den Wiederbesuch von Seiten bei der „Hub-and-Spoke"-Navigation statt des Zurück-Buttons eingesetzt werden. Neu sind die Gruppierung von Tabs (ab Opera 11 und Firefox 4) und die „App Tabs" (Google Chrome und Firefox ab Version 4), als vom Benutzer ausgewählte *permanente* Reiter, die der Browser beim Start automatisch öffnet, nur durch ein Icon repräsentiert werden und farblich auf Änderungen des Inhalts hinweisen (s. Abb. 162, Kasten links oben). Sie sind damit für Online-Applikationen wie *Google Mail* und *Facebook* prädestiniert; solche Werkzeuge eignen sich aber auch gut für die Beobachtung häufig aktualisierter Informationsseiten, beispielsweise von Nachrichtenportalen (vergl. Kellar, Watters et al. 2007).

Eine weitere für die Benutzbarkeit positive Entwicklung ist der verbesserte „History-Support" (s. Abschnitt 3.1.7) aktueller Browser, der auch die Kritik von (Herder, Weinreich et al. 2006) und (Obendorf, Weinreich et al. 2007) aufgreift. Aktuelle Versionen von Chrome, Safari und Opera zeigen in neu geöffneten Tabs eine Übersicht der am häufigsten besuchten Seiten als Thumbnails an (s. Abb. 162 Mitte) und vereinfachen so den direkten Zugriff auf diese Ressourcen. Eine „Leseliste" in Safari 5.1 dient zum Vormerken von URIs für das spätere Lesen, die nach dem Seitenaufruf automatisch gelöscht werden.

Die Darstellung von Hyperlinks bleibt hingegen seit über 15 Jahren fast unverändert und gleicht immer noch der des ersten populären Browsers Mosaic von 1993 (s. Abschnitt B.15). Die einzigen wesentlichen Änderungen sind, dass sich die Hervorhebung von textlichen Link-Ankern sowohl durch Web-Autoren als auch durch Benutzer anpassen lässt (s. Abschnitt 5.1.3) und dass man mithilfe eines Attributs zusätzliche Textinformationen zum Link in einem kleinen Tooltip anbieten kann (s. Abschnitt 5.3.7). Beide Änderungen gehen aber schon auf die späten 1990er Jahre zurück. Die programmatischen Möglichkeiten für Links sind hingegen in den letzten Jahren durch den Einsatz von JavaScript und Ajax immer umfangreicher geworden, sodass sich inzwischen nahezu beliebige Interaktionsformen mit Links realisieren lassen (s. Abschnitt 4.5.1.3). Des Weiteren führen zahlreiche Dienste zu neuen Herausforderungen bei der Link-Navigation. Ein Beispiel sind „Link-Verkürzer" wie *Bit.ly* und *tr.im*, mit deren Hilfe sich URIs in Kurzform repräsentieren lassen; so führt die Adresse „http://bit.ly/MRIkY" zur Login-Seite von Facebook. Bei einer (nicht-repräsentativen) Studie von Ende 2010 mit über 1900 Teilnehmern folgten 97% der Personen unbedacht Links mit solchen kurzen URIs, offenbar in Unwissenheit, dass sie auch direkt zu Seiten mit Schadcode führen können (BitDefender 2010). Der Anwender erhält durch Webbrowser dennoch weiterhin keine verlässlichen Informationen darüber, welche Aktion ein Link auslöst und zu welcher Ressource er führt. Das potenzielle Informationsdefizit (s. Kapitel 3.4 und 4.1ff) nimmt folglich eher zu als ab.

Aktuelle Browser tragen gleichzeitig der ISO 9241-151 keine Rechnung, die seit 2008 zahlreiche Forderungen für die Link-Schnittstelle im Web stellt, und festlegt, dass „besondere" Links kenntlich zu machen sind. Beispielsweise sollen Links zu besonderen Dateitypen

(ISO9241-151 2008, 9.4.9), Links, die neue Fenster öffnen (ISO9241-151 2008, 9.4.10) und „Transaktions-Links“ (Links, die Aktionen ausführen, siehe: ISO9241-151 2008, 9.4.4) für den Anwender eindeutig als solche erkennbar sein. Bis heute bleibt eine derartige Link-Kennzeichnung im Web alleine den Web-Autoren überlassen, aber nur wenige Anbieter berücksichtigen die Forderungen des Standards. Sofern besondere Links doch einmal kenntlich gemacht werden, geschieht dies in individueller und somit inkonsistenter Weise, was dem Grundsatz der Erwartungskonformität von benutzbarer Software widerspricht (s. ISO9241-110 2006). Eine weitere Forderung der ISO 9241-151 ist, dass alle Links einfach identifizierbar sind (ISO9241-151 2008, 9.4.2). Auch dies ist bei heutigen Webbrowsern nicht sichergestellt, da Web-Autoren die Hervorhebung von Link-Ankern selbst festlegen können und sich inzwischen beliebige Objekte mit Verknüpfungen versehen lassen (s. Abschnitt 4.4.1).

Die in dieser Arbeit vorgestellten Konzepte zur Erweiterung der Link-Benutzungsschnittstelle berücksichtigen die in der ISO 9241-151 aufgeführten Kriterien für die Ergonomie von Hyperlinks im Web: Es werden sowohl die eindeutige Hervorhebung von Links durch die *Link-Overlays-on-Demand* als auch eine Kennzeichnung besonderer Links durch *Overlay-Farben*, *multiple Maus-Icons* und *Tooltips* automatisch vorgenommen (s. Abschnitt 6.4). Die Konzepte von HyperScout II machen somit Webbrowser bezüglich der Link-Schnittstelle erstmals konform zum ISO-Standard, auch wenn die Anbieter einer Website den Standard nicht berücksichtigen.

9.2.2 Zur Einführung erweiterter Link-Benutzungsschnittstellen in der Praxis

Die aktuell unbefriedigende Situation bezüglich der Benutzbarkeit vieler Hyperlinks und die insgesamt sehr ermutigenden Evaluationsergebnisse und Bewertungen der Studienteilnehmer der HyperScout-Prototypen werfen die Frage auf, warum ein Konzept wie *HyperScout II* bisher nicht in aktuellen Webbrowsern integriert wurde. Diese Frage ist nicht einfach zu beantworten, aber einige potenzielle Faktoren lassen sich als Hypothese aufstellen.

Zum einen ist es gut möglich, dass die Probleme bei der Navigation im Web von Browser-Herstellern bisher nicht ausreichend erkannt wurden. Die Schwerpunkte werden momentan primär bei einer besseren Unterstützung multimedialer Inhalte und interaktiver Web-Anwendungen gesetzt (s. Abschnitt 9.2).

Zweitens kann man von einem „Henne-Ei-Problem“ sprechen, da die volle Funktionalität von HyperScout einerseits eine Erweiterung der Webserver erfordert, sodass sie die benötigten Daten zur Verfügung stellen, und andererseits eine Anpassung der Browser notwendig ist, damit sie diese Informationen den Benutzeranforderungen gemäß darstellen. Die für das HyperScout-Konzept entwickelten Protokoll- und Spracherweiterungen (s. Kapitel 7.5) stellen zwar sicher, dass es keine technischen Kompatibilitätsprobleme mit dem existierenden Web gibt, dennoch wird eine solche Erweiterung für die meisten Anbieter erst interessant, wenn sie damit viele ihrer Benutzer erreichen. Für Benutzer hingegen wird das System erst dann zu einer verlässlichen Hilfe, wenn es von vielen Websites unterstützt wird.

Ein dritter Grund können Sorgen bezüglich der Kompatibilität entsprechend erweiterter Browser mit der *Benutzungsoberfläche* heutiger Websites sein; HTML5, JavaScript und CSS setzen kaum Grenzen bei der Gestaltung von Web-Angeboten, sodass durchaus ungewollte Effekte bei bestimmten Konstellationen mit den Konzepten von HyperScout II denkbar sind: So könnten sich die Tooltips von HyperScout II mit JavaScript-Tooltips einer Web-Anwendung überlappen oder sich die Link-Overlays störend auf die Darstellung der Seite auswirken. Verhindern ließen sich solche unerwünschten Wechselwirkungen allerdings auf einfache Weise, beispielsweise durch das Einbetten eines zusätzlichen Meta-Elements in den HTML-Code (s. Abschnitt 4.5.2.6), das die HyperScout-Erweiterung für die aktuelle Seite einschränkt.

Aus anderer Perspektive betrachtet werden heute bereits mehrere Teilkonzepte von HyperScout durch zahlreiche Browser-Erweiterungen realisiert. Alleine für den Browser Firefox sind Ende 2011 über 5000 Erweiterungen verfügbar, von denen mehrere Dutzend versuchen, die Navigation im Web, den Zugriff auf früher besuchte Seiten, die Sicherheit beim Browsen im Web und den Umgang mit Tabs zu vereinfachen. Einige Ideen dieser Arbeit findet man in einzelnen Browser-Erweiterungen wieder, beispielsweise bei *WOT* zur Kennzeichnung von Links zu nicht vertrauenswürdigen Websites und beim *LinkChecker* zur farblichen Markierung defekter Links (s. folgender Abschnitt 9.2.3).

Trotz der geschilderten Hindernisse sollten sich die Konzepte von *HyperScout II* mittelfristig vielen Benutzern des Webs zugänglich machen lassen. Eine Browser-Extension, z. B. für Firefox oder Google Chrome, die alle Möglichkeiten der Benutzungsschnittstelle von *HyperScout II* integriert, ließe sich auf Basis der Ergebnisse dieser Arbeit innerhalb weniger Wochen entwickeln. Ein größeres Problem stellen die entsprechenden Erweiterungen für Webserver dar, da idealerweise *alle* Betreiber ihre Server entsprechend aktualisieren müssten. Als Übergangslösung könnte ein Metadaten-Dienst dienen, der unabhängig von Webservern arbeitet und global für die meisten Websites einen Großteil der benötigten erweiterten Link-Informationen bereitstellt. Eine technische Grundlage für die Übertragung zusätzlicher Link-Daten zu Webseiten basierend auf RDF (Berners-Lee, Hendler et al. 2001) und REST (Fielding 2000) wird in Kapitel 7.4.1 kurz vorgestellt. Ein solcher zentraler Dienst wäre voraussichtlich innerhalb einiger Monate einsetzbar, allerdings erfordert er Partner mit ausreichend Kapital, um die benötigte Bandbreiten- und Rechenkapazität zur Verfügung zu stellen. Bewerkstelligen ließe sich dies problemlos durch Firmen wie Google oder Microsoft, die zum einen das Web bereits für ihre Suchmaschinen indexieren und zum anderen über die notwendigen Geschäftsmodelle für derartige „kostenfreie“ Angebote verfügen. Ein zentraler Dienst für erweiterte Link-Daten kann kommerziell lukrativ sein, da sich gleichzeitig Informationen über die Interaktion der Benutzer mit dem Web erfassen lassen. Dies zeigen Erweiterungen wie *WOT* (s. folgender Abschnitt) oder der *McAfee Site Advisor* (s. Abschnitt 3.3.3).

9.2.3 Browser-Erweiterungen zur Vereinfachung der Link-Navigation

Die Einführung von *Firefox 1.0* im November 2004 hat die Entwicklung von Browser-Erweiterungen wesentlich vereinfacht, da Firefox eine flexible Schnittstelle für „Add-Ons“ eingeführt hat, die sich bedeutend leichter als die vorheriger Browser programmieren lässt (s. Abschnitt 8.3.1). Dem Beispiel folgend bieten inzwischen andere Browser wie *Google Chrome*, *Apple Safari* und *Opera* vergleichbare, wenn auch inkompatible Erweiterungsschnittstellen an. Diese Schnittstellen lassen sich zwar weder konzeptionell noch funktional mit dem in dieser Arbeit vorgestellten Framework *Scone* vergleichen (s. Kapitel 8.6), bieten aber die Vorteile der direkten Integration in den Browser und der komfortablen Installation von Add-Ons aus einem globalen Repository, wodurch sie schnell sehr populär geworden sind.

Von den mehreren tausend verfügbaren Browser-Add-Ons versuchen zahlreiche auch die Benutzbarkeit der Browser zu verbessern, und einige beziehen sich konkret auf die Defizite der Link-Navigation. Im Folgenden werden drei Beispiele diskutiert, die einen Bezug zu den in dieser Arbeit entwickelten Konzepten aufweisen und die gegenwärtigen *Möglichkeiten* und *Grenzen* in diesem Bereich aufzeigen.

WOT (für „Web of Trust“) gehört zu den populärsten Browser-Erweiterungen und ist für die meisten Desktop-Browser erhältlich.[345] *WOT* bindet in die Seiten zahlreicher Suchmaschinen und E-Mail-Dienste (wie *Google Mail* und *Microsoft Hotmail*) hinter allen Link-Ankern zu externen Websites ein kreisförmiges Icon ein, das in einer von fünf Farben die „Reputation“ der Website repräsentiert. Das Spektrum reicht von dunkelgrün („exzellente Reputation“) über gelb bis dunkelrot („sehr niedrige Reputation / potenziell gefährliche Website“). Wenn der Anwender den Mauszeiger über das Icon bewegt, erscheint ein Tooltip mit detaillierten Informationen zur Reputation (s. Abb. 163). Benutzer haben zudem die Möglichkeit, Websites selbst zu bewerten und so zum „Web of Trust“ beizutragen (vergleiche das Projekt *Co-Internet*, Abschnitt 8.7). *WOT* nutzt eine REST-Schnittstelle zur Abfrage der Reputationsdaten von den Servern des Anbieters der Erweiterung (s. Abschnitt 7.4.1). Es wird eine XML-Datei zurückgeliefert, die Informationen über alle externen Links der aktuellen Seite enthält. Allerdings ist *WOT* auf die Websites von populären Suchmaschinen und Webmail-Anbietern beschränkt.

[345] Die Erweiterung *WOT* wird von der finnischen Firma „*WOT Services Oy*“ angeboten. Die Angaben zur Anzahl der Downloads stammen vom Hersteller, Stand: November 2011. Siehe: http://www.mywot.com/.

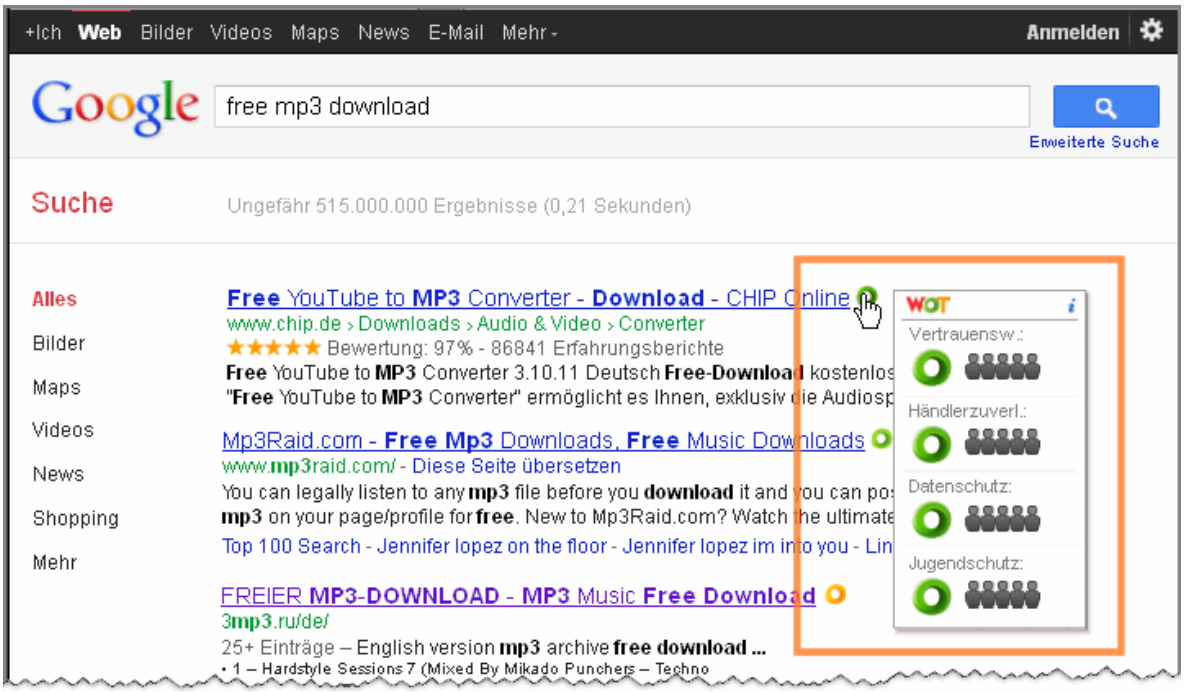


Abb. 163: Die Erweiterung *WOT* zeigt mithilfe von Icons und Tooltips die Reputation von Websites an. Der erste Link der Liste wird als sehr vertrauenswürdig bewertet, der dritte Treffer hingegen nicht.

Die Firefox-Erweiterung *LinkChecker*[346] überprüft sämtliche Links einer Seite und zeigt den „Status" der Links mittels eines farbigen Link-Overlays an (s. Abschnitt 6.4.1). Das Werkzeug lässt sich aus einem Menü des Browsers aktivieren, woraufhin nacheinander die Ziel-Adressen aller Links der aktuellen Seite per Prefetching angefragt werden. Grün gekennzeichnete Links sind intakt (s. Abschnitt 6.4.1 und Tabelle 2), rote Links führen zu einer Fehlermeldung und bei Gelb wird der Link umgeleitet, der Server ist gegenwärtig nicht erreichbar oder der Zugriff auf das Dokument wird verwehrt (s. Abb. 164 links). Die Erweiterung ist insbesondere für Autoren von Webseiten nützlich, da sie einen Überblick über fehlerhafte Links verschafft. Aufgrund des sequenziellen Prefetchings aller Link-Ziele dauert die Überprüfung einer Seite jedoch häufig recht lange.

[346] LinkChecker wurde von Kevin A. Freitas entwickelt und ist unter folgender Adresse zu finden: https://addons.mozilla.org/de/firefox/addon/linkchecker/developers.



Abb. 164 links: Die Erweiterung *LinkChecker* kennzeichnet den Status aller Link-Ziele durch farbige Overlays. Rechts: *Link Alert* hebt besondere Links durch Icons in einem Tooltip hervor. Das Beispiel zeigt einen Link zu einer anderen Website, der ein neues Fenster öffnet.

Link Alert für Firefox zeigt zusätzliche Link-Informationen mithilfe von Icons an, die unterhalb des Mauszeigers in einer Art Tooltip erscheinen. Das System analysiert hierfür die Attribute des Anker-Elements und zeigt Icons für Links zu besonderen Dateitypen, mit anderen Protokollen als HTTP und Links, die neue Fenster öffnen oder zu anderen Websites führen (s. Abb. 164 rechts). Die maximal dargestellte Anzahl von Icons kann eingestellt werden, alle Icons lassen sich in der Konfiguration einzeln deaktivieren, und der Benutzer hat die Möglichkeit, zusätzliche Icons für weitere Dateitypen zu ergänzen. Die Konfigurationsmöglichkeiten sind damit flexibler als bei *HyperScout II*, jedoch kann *Link Alert* systembedingt viele Link-Attribute nicht berücksichtigen, da es lediglich die Attribute des Anker-Elements auswertet.

Alle drei Erweiterungen weisen einzelne Aspekte der in dieser Arbeit entwickelten und im Prototypen *HyperScout II* realisierten Konzepte auf. *WOT* demonstriert, wie sich schnell serverseitig Informationen für alle Links einer Seite bereitstellen und als Icons und Tooltips einbinden lassen. Dies gelingt, obwohl ein zentralistischer Ansatz verfolgt wird. WOT beschränkt sich zwar auf Kriterien der Reputation und Qualität bei der Link-Navigation und ist nur für wenige (wenn auch bedeutende) Websites verfügbar, jedoch ließe sich dieses Konzept für weitere Websites und andere Attribute erweitern. *LinkChecker* verwendet Link-Overlays und Prefetching, um einen Überblick über den Status aller Links einer Seite zu geben, und *Link Alert* zeigt, wie man wichtige Link-Attribute mithilfe von Icons beim Mauszeiger darstellt, ohne dass zusätzliche Daten übertragen werden müssen. Allerdings bleiben alle verfügbaren Erweiterungen auf bestimmte Aspekte der Link-Navigation beschränkt und lassen viele wichtige Informationen unberücksichtigt. Zudem sind keine Evaluationsergebnisse zu den existierenden Erweiterungen verfügbar, und es ist daher nicht bekannt, welche Potenziale und Grenzen diese spezifischen Lösungen jeweils bezüglich der Link-Navigation aufweisen.

Für Endanwender führt die Vielzahl von Browser-Erweiterungen dazu, dass sie sie kaum überblicken können und sich die jeweiligen Vor- und Nachteile nur schwer erfassen lassen.

Zusammenfassend betrachtet fehlt bisher ein konsistentes Gesamtkonzept für Browser-Erweiterungen, das alle wichtigen Aspekte der Link-Navigation berücksichtigt, so wie es in dieser Arbeit vorgestellt wurde.

9.3 Ausblick

Das Web wird sich aller Voraussicht nach auch zukünftig rasch weiterentwickeln. Neben dem bereits aufgeführten Siegeszug von Web-Applikationen (s. Abschnitt 2.2.5 und 9.2) ist auch beim *Zugriff* auf das Web eine Entwicklung zu beobachten, die Auswirkungen auf die Anforderungen der Benutzer bei der Web-Interaktion hat. Zum einen nimmt der *mobile* Zugang zum Internet über Funktechniken wie UMTS und LTE stetig zu (UMTS-Report 2007; Fennah & Botterill 2010). Hier spielen Datenvolumen und Performanz zumeist eine kritischere Rolle als bei nicht-mobilen Zugängen. Diese hiermit verbundenen Einschränkungen haben Auswirkungen auf das Navigationsverhalten der Benutzer und die Herausforderungen bei der Navigation (s. Kapitel 9.3.1).

Des Weiteren werden Geräte mit Benutzungsschnittstellen, die nicht der Desktop-Metapher folgen und eine WIMP-Schnittstelle bieten, immer populärer (Goasduff & Pettey 2010). Hervorzuheben sind dabei insbesondere Systeme mit berührungsempfindlichem Bildschirm wie Smartphones und Tablet PCs (z. B. das *iPad*), die inzwischen Hunderte Millionen von Anwendern haben (Pettey & Stevens 2011). Solche Geräte erfordern aufgrund neuer Interaktionsparadigmen andere Konzepte zur Erweiterung der Link-Benutzungsschnittstelle als Desktop-Systeme (s. Kapitel 9.3.2). Die Vielfalt der neuen Geräteklassen und Oberflächenkonzepte erhöht gleichzeitig die Bedeutung der Barrierefreiheit im Web, wozu die in dieser Arbeit entwickelten Techniken beitragen können (s. Kapitel 9.3.3). Zur prototypischen Realisierung und Evaluation von Erweiterungen zur Navigationsvereinfachung im Web für die neuen mobilen Geräteklassen bieten sich die in Kapitel 8 vorgestellten intermediary-basierten Konzepte an, da sie keine spezielle Erweiterungsschnittstelle für Browser voraussetzen (s. Kapitel 9.3.4).

9.3.1 Bedeutung der Ergebnisse der Arbeit für den mobilen Zugriff auf das Web

Der mobile Zugriff auf das Web (UMTS-Report 2007) und die Verwendung von Smartphones können einen steten Zuwachs verzeichnen (Pettey & Stevens 2011). Bei einer repräsentativen Umfrage von Internetnutzern innerhalb der EU gaben bereits Ende 2009 26 % der Teilnehmer an, regelmäßig mit Mobilgeräten auf das Internet zuzugreifen (Fennah & Botterill 2010). In immer mehr Schwellenländern (wie Nigeria und Indien[347]) spielt der mobile Internetzugriff mit Smartphones aufgrund der verfügbaren Infrastruktur und aus Kostengründen eine ähnliche Rolle wie der Zugriff über Desktop-PCs und Festnetzanschlüsse (Donner 2010; Donovan & Donner 2010).

[347] Aktuelle Statistiken zur Nutzung von Mobilgeräten und Browsern findet man bei den Firmen StatCounter (http://gs.statcounter.com/) und Net Applications Inc. (http://www.netmarketshare.com/).

Übertragungsstandards wie LTE und UMTS mit HSPA bieten beim mobilen Internetzugang Transferraten von mehreren MBit/s, aber dennoch treten bei der Übertragung von Webseiten oft wesentlich größere Wartezeiten auf als bei der Nutzung von DSL und Standleitungen (s. Abschnitt 3.3.1). Zusätzlich dauert die Interpretation der Webseiten auf kleinen Mobilgeräten wie Smartphones infolge der relativ leistungsschwachen Prozessoren meist einige Sekunden; dies führt zu weiteren Verzögerungen bei der Navigation (Nielsen 2009a; Meyerovich & Bodík 2010). Darüber hinaus sind beim mobilen Internetzugriff Onlinezeit und Datenvolumen häufig mit Kosten verbunden oder monatlich begrenzt.

Für Anwender können aus diesen Gründen daher fehlerhafte Navigationsschritte im Web noch störender sein als bei schnellen Festnetzanbindungen, wo neue Seiten oft bereits nach ein bis zwei Sekunden erscheinen (s. Abschnitt 3.3.1). In Studien hat sich gezeigt, dass Benutzer bei langen Antwortzeiten und begrenztem Transfervolumen anders im Web navigieren und sich Navigationsschritte genauer überlegen als wenn derartige Einschränkungen fehlen (Chen, Subramanian et al. 2009; Chetty, Banks et al. 2011). Zusätzliche Link-Hinweise, beispielsweise auf die zu erwartende Übertragungszeit und die Dateigröße, werden in solchen Fällen als besonders hilfreich erachtet (s. Kapitel 6.3ff; Chen, Subramanian et al. 2009). Die erweiterte Link-Schnittstelle von HyperScout stellt solche und weitere Zusatzinformationen zur Verfügung und ist daher bei Mobilgeräten wahrscheinlich in noch mehr Situationen nützlich als bei den in dieser Arbeit im Fokus stehenden Desktop-Systemen.

Die Anzeige der Link-Hinweise erfordert zwar die Übertragung zusätzlicher Daten, jedoch sind diese für gewöhnlich im Verhältnis zum Gesamtdokument von geringem Umfang, als textliche Parameter in den Code der Seite eingebettet und somit gut komprimierbar (s. Abschnitt 3.3.1). Eine Verlängerung der Übertragungszeit wird auf diese Weise minimiert. Eine Vereinfachung der Navigation, die möglicherweise geringere Anzahl übertragener Webseiten und die ersparte Arbeitszeit der Anwender, als eines ihrer wertvollsten Güter, stehen dem als potenzielle Vorteile gegenüber (siehe auch Abschnitte 7.6 und 9.3.4).

9.3.2 Erweiterung der Link-Schnittstelle für neue Geräteklassen

Die ersten Konzepte für „Multi-Touch"-Bedienoberflächen wurden bereits Anfang der 1980er entwickelt;[348] entsprechende Geräte sind aber erst seit der Einführung des *iPhones* im Jahr 2007 einer breiten Masse zugänglich, wobei sie seitdem stetig an Popularität gewonnen haben (Goasduff & Pettey 2010). Bei dieser Geräteklasse interagiert der Anwender direkt mit den Fingern auf einem kapazitiv-berührungsempfindlichen Bildschirm und steuert so auch den Webbrowser. Diese sogenannten *„Natural User Interfaces"* (auch *NUIs*, siehe: Wigdor & Wixon 2011) sind bisher vor allem bei Smartphones und Tablet-PCs mit speziellen Betriebssystemen für Mobilgeräte (wie *Apple iOS* oder *Google Android*) zu finden, jedoch soll ab Ende des Jahres 2012 *Windows 8* ebenfalls eine entsprechende Oberfläche mit dem Namen *Metro*

[348] Eine gute Übersicht zur Entwicklung von Multi-Touch-Oberflächen bietet Bill Buxton (siehe auch: Lee, Buxton et al. 1985) von Microsoft Research unter: http://www.billbuxton.com/multitouchOverview.html.

bieten.[349] NUIs sollen Benutzern das Gefühl geben, auf direktere und natürlichere Weise mit einem System zu interagieren. Sie werden herkömmliche grafische Schnittstellen (GUIs) voraussichtlich nicht verdrängen, sich aber als eigener Bereich in der Computertechnik etablieren (Wigdor & Wixon 2011: 18).

Die vergleichsweise kleinen Bildschirme und die direktere Interaktion als bei Desktop-Systemen mit WIMP-Oberfläche bringen zahlreiche neue Herausforderungen mit sich (Nielsen 2009b). So verdeckt beispielsweise der Finger Teile der Oberfläche, die Bedienelemente müssen größer[350] als bei Maussteuerung sein, da kein Zeigewerkzeug eingesetzt wird, und es steht in der Regel nur eine virtuelle statt einer physischen Tastatur zur Verfügung (Wigdor & Wixon 2011).

Die für HyperScout II (s. Kapitel 6.4ff) vorgeschlagenen und evaluierten Erweiterungen der Link-Schnittstelle sind für diese Geräteklasse daher nur begrenzt einsetzbar: Tooltips und Maus-Icons eignen sich in der bei HyperScout II evaluierten Interaktionsform nicht, da es keinen Mauszeiger gibt, den man zur Einblendung der Tooltips über aktive Elemente führen könnte, und für die Aktivierung von Link-Overlays gibt es in der Regel keine frei belegbaren Tasten.

Gleichzeitig tritt die Problematik der oft nicht eindeutigen Identifizierbarkeit von Link-Ankern bei dieser Geräteklasse noch deutlicher zutage, weil Benutzer mangels Mauszeigers nicht nach aktiven Bereichen auf Webseiten suchen können (s. Kapitel 5.2.7). Führt der Benutzer den Finger über den Bildschirm, wird in der Regel stattdessen der Seiteninhalt verschoben („gescrollt"); tippt er irrtümlich auf einen nicht-aktiven Bereich statt auf einen Link, so erhält er zumeist gar keine Rückmeldung vom Browser oder der entsprechende Seitenbereich wird gezoomt. *Link-Overlays-On-Demand* als Mittel zur eindeutigen Kennzeichnung von Link-Ankern sind daher bei diesen Systemen mindestens so zweckmäßig wie bei Browsern für herkömmliche grafische Benutzungsoberflächen. Aufgrund der (zumeist) fehlenden Tastatur ist bei NUIs eine andere Benutzeraktion zur Aktivierung der Overlays erforderlich als bei HyperScout II (hier wurde die STRG-Taste verwendet, s. Abschnitt 6.4.1), die sich nicht mit existierenden Eingabekommandos oder Gesten überschneidet. Zwei Lösungsvorschläge sind ein reservierter Bereich auf dem Bildschirm (eine Art virtuelle Taste), der bei Anwahl die Overlays einblendet und die Aktivierung der Overlays bei längerer Berührung des Displays ohne Scrollbewegung (s. Kapitel 9.3.4).

Die Problematik der oft ungenügenden Link-Informationen (s. Abschnitt 3.4 und 4.1ff) ist bei den gegenwärtigen Browsern für Systeme mit Multi-Touch-Oberfläche ebenfalls potenziell noch bedeutender als bei Desktop-Browsern: Für Benutzer ist weder der URI des Link-Zieles

---

[349] Entwicklerversionen von Windows 8 sind seit Sept. 2011 verfügbar. Siehe: http://windows.microsoft.com/en-US/windows-8/preview.

[350] Dies ist als "Fat Finger"-Problem bekannt geworden. Da ein Finger sich lange nicht so präzise auf dem Bildschirm positionieren lässt wie ein Mauszeiger, wird bei NUIs als Mindestgröße von Bedienelementen meist 0,9 cm bis 1,5 cm angesehen gegenüber ca. 4 mm bei Maussteuerung (Wigdor & Wixon 2011, S. 73ff).

verfügbar (zumal NUI-Browser keine Statuszeile bieten) noch gibt es Tooltips, die sonst für das „`title`“-Attribut von Link-Ankern eingesetzt werden (s. Abschnitt 5.3.7).

Zusätzliche Informationen zu den Links ließen sich bei Browsern in NUIs – ebenso wie bei Desktop-Systemen – durch verschiedenfarbige Link-Overlays visualisieren. Des Weiteren könnten verschiedene Schrifttypen oder auch Icons eingesetzt werden, die beispielsweise hinter allen Link-Ankern erscheinen (s. Abschnitte 5.3ff). Für das Anzeigen detaillierterer, textlicher Informationen zu ausgewählten Links sind aber die für HyperScout II vorgeschlagenen Tooltips und Maus-Icons nur eingeschränkt geeignet, da NUIs keinen Mauszeiger zur Einblendung dieser Elemente bieten. Zudem ist die Bildschirmoberfläche von Mobilgeräten in der Regel kleiner als die von Desktop-PCs, sodass größere Tooltips potenziell zu viel verdecken und die zusätzlich angezeigten Link-Informationen auf das Wesentliche zu beschränken sind. Eine für NUIs angepasste Variante von Tooltips könnte so aussehen, dass ein kleines Fensterelement 1–2 cm oberhalb des Fingers erscheint (und nicht wie sonst üblich vorwiegend rechts unterhalb des Mauszeigers). So ließe sich ein Verdecken des Elements durch die Hand des Anwenders vermeiden. Zum Aufrufen des Tooltips könnte der Benutzer z. B. mit dem Finger einen Link-Anker etwas länger berühren, da Browser bei Multi-Touch-Oberflächen einen Link erst öffnen, nachdem sich der Finger wieder vom Bildschirm löst. Die hier vorgeschlagenen Konzepte können als Anregung dienen und müssten ebenfalls mit Benutzern evaluiert werden, um die Eignung für Webbrowser in NUIs mit Multi-Touch-Oberfläche zu verifizieren (s. Kapitel 9.3.4).

Einen Überblick über die verfügbaren Schnittstellenkonzepte für die Darstellung von Link-Ankern und von erweiterten Link-Informationen und die jeweiligen Stärken und Schwächen der Methoden findet man in den Kapiteln 5.2 und 5.3. Die Ergebnisse dieser Klassifikationen lassen sich auch für die Gestaltung von Navigationshilfen bei NUIs nutzen, wobei als zusätzliche Kriterien der verfügbare Platz und die Möglichkeiten zur Aktivierung der Link-Informationen zu berücksichtigen wären.

9.3.3 Barrierefreiheit und erweiterte Link-Schnittstellen im Web

Die in den vorherigen Abschnitten beschriebenen Entwicklungen des Webs zeigen gleichzeitig, dass die Barrierefreiheit („Accessibility“) eine immer bedeutendere Rolle spielt. Die wachsende Vielfalt der Anwendungsgebiete (s. Abschnitt 9.2), die zunehmenden Erfordernisse bezüglich der Flexibilität beim Zugriff auf das Web (s. Abschnitt 9.3.1) und die neuen Geräteklassen (s. Abschnitt 9.3.2) sind wesentliche Ursachen hierfür. Hinzu kommt, dass Menschen in immer mehr Bereichen auf das Web angewiesen sind; dies gilt ebenso für Personen mit physischen Einschränkungen.

Die Erfordernisse der Barrierefreiheit spiegeln sich in zahlreichen Normen, Regelungen und Standards wider. Beispielsweise fordert die ISO-Norm 9241-151 (ISO9241-151 2008, 10.9), dass Websites unabhängig von der Geräteklasse und dem Eingabegerät funktionieren müssen. In Deutschland legt die Rechtsverordnung *BITV 2* (BITV2 2011) fest, dass Einrich-

tungen des öffentlichen Rechts ihre Internetangebote so zu gestalten haben, dass sie für Menschen mit Behinderungen im Sinne des *Behindertengleichstellungsgesetzes* nutzbar sind. Die bedeutendsten Standards für barrierefreie Systeme sind die *Web Content Accessibility Guidelines (WCAG)* in Version 1.0 (Chisholm, Vanderheiden et al. 1999) und Version 2.0 (Caldwell, Cooper et al. 2008) von der *Web Accessibility Initiative (WAI)* des W3C. Sie beschreiben zahlreiche Kriterien für die Gestaltung barrierefreier Websites, unter anderem auch zur Gestaltung barrierefreier Hyperlinks.

Studien zeigen allerdings, dass nur ein Bruchteil der Websites die Standards der WAI berücksichtigen (Hackett, Parmanto et al. 2004; Loiacono & McCoy 2006; Loiacono, Jr. et al. 2009). Die Problematik nimmt aufgrund des steigenden Anteils von Websites mit interaktiven Angeboten tendenziell zu, weil interaktive Seiten besondere Herausforderungen an die Barrierefreiheit stellen und Online-Applikationen von den Standards nur begrenzt berücksichtigt werden (Lazar, Dudley-Sponaugle et al. 2004; Loiacono, Jr. et al. 2009). Beispielsweise haben Screenreader regelmäßig Probleme mit Webseiten, deren Inhalt sich per Ajax (s. Abschnitt 2.2.5) bei der Nutzung ändert, da Screenreader die Seiten beim Laden interpretieren und sequenziell vorlesen. Werden in Seiten dynamisch Informationen verändert, so lässt sich dies schwerlich sequenziell wiedergeben.

Erfüllt eine Website die Kriterien der Barrierefreiheit nicht, so fehlen Benutzern mit Einschränkungen bei der Navigation potenziell noch häufiger Informationen über die Bedeutung von Links als Anwendern ohne Beeinträchtigungen. Zwei Gründe hierfür sind, dass nur wenige Autoren Link-Titel definieren (s. Abschnitte 4.3 und 4.4.2) und auch Alternativtexte[351] zu eingebetteten Objekten häufig fehlen (Google 2006). Ein Beispiel für ein regelmäßig auftretendes Problem der Barrierefreiheit bei der Navigation sind grafische Link-Anker, die Screenreader naturgemäß nicht vorlesen können. Die (leicht unterschätzbare) Komplexität barrierefreier Gestaltung rührt daher, dass Anwender von Computersystemen viele unterschiedliche Arten von Beeinträchtigungen haben können (Siegmund 2008; Piotrowski & Tauber 2009), die jeweils andere Herausforderungen an die Benutzungsschnittstelle stellen.

Links sind mit Abstand das bedeutendste Navigationsmittel im Web (s. Kapitel 2.2.4; Weinreich, Obendorf et al. 2006b; Obendorf, Weinreich et al. 2007), sie müssen daher auch bei der Barrierefreiheit angemessen berücksichtigt werden. Die in dieser Arbeit vorgeschlagenen Konzepte und Erweiterungen des HTTP-Protokolls und der Sprache (X)HTML tragen dem Rechnung, indem mit ihrer Hilfe automatisch zu allen Links textliche Hinweise hinzugefügt werden können, die alle wesentlichen Eigenschaften der Links und Link-Ziele berücksichtigen (s. Kapitel 7.5). Diese Text-Informationen lassen sich einfach verarbeiten und nicht nur optisch darstellen, sondern auch akustisch wiedergeben oder für das Filtern von

[351] Das Bereitstellen von äquivalenten Alternativen für Audio- und visuelle Inhalte wird als erster Punkt der WCAG 1.0 und 2.0 aufgeführt (Chisholm, Vanderheiden et al. 1999; Caldwell, Cooper et al. 2008).

Links verwenden (Noirhomme-Fraiture & Serpe 1998). Beispielsweise kann ein Screenreader dem Benutzer den automatisch bereitgestellten Titel des Link-Ziels vorlesen, wenn der Link-Anker eine Grafik ist und weitere Informationen fehlen.[352] Bei anderen Anforderungen, wie dem mobilen Zugriff per Smartphone, kann hingegen auf den Dateityp oder die Größe des Zielobjekts hingewiesen werden (Chen, Subramanian et al. 2009).[353] Je nach Situation, Art des Links und Benutzerprofil lassen sich die jeweils relevanten Informationen auswählen und auf geeignete Weise mitteilen.

9.3.4 Zukünftige Schritte

Diese Arbeit hat zahlreiche Ergebnisse zur Reduktion der Probleme bei der Navigation im Web geliefert und gleichzeitig Möglichkeiten für weitere Forschungsarbeiten aufgezeigt. Die präsentierten Konzepte und Ergebnisse können als Grundlage für zukünftige Forschungs- und Entwicklungsprojekte dienen und haben dies auch bereits getan, wie die auf den Projekten *Scone* und *HyperScout* aufbauenden Publikationen zeigen (s. Kapitel 8.7).

Die Studienergebnisse zur erweiterten Link-Benutzungsschnittstelle von HyperScout II legen weitere Schritte nahe, um die Konzepte zu optimieren und zusätzliche Erkenntnisse zu gewinnen. Beispielsweise wäre es sinnvoll, einen auch für Endanwender leicht zu installierenden Prototypen (z. B. als Firefox-Extension) zu entwickeln, der unter anderem ein Hilfesystem bietet und flexible und dennoch einfach bedienbare Konfigurationsmöglichkeiten für die Link-Benutzungsschnittstelle aufweist (s. Kapitel 6.2.6 und 6.5.6). Eine vereinfachte Installation würde es erlauben, das System auch im Rahmen einer *größeren Langzeitstudie* auf Desktop-Systemen zu evaluieren, die weitere Ergebnisse zur Nutzung der erweiterten Link-Schnittstelle liefern könnte. Für eine solche Studie wäre neben einer Online-Teilnehmerbefragung per Fragebogen (vergl. Abschnitte 6.1.2 und 6.3; Root & Draper 1983) eine freiwillige Protokollierung der Nutzung der Schnittstellenerweiterung sinnvoll, sodass man empirische Ergebnisse zum Benutzerverhalten und den Vorlieben bei der Konfiguration erhält (s. Kapitel 6.1.2 und 8.6.4.1). Damit sich die Daten interpretieren lassen, wäre es notwendig, neben den besuchten Websites auch die Aufgaben der Anwender, die verwendeten Plattformen und die verfügbare Bandbreite zu berücksichtigen. Auf diese Weise ließen sich zusätzliche und voraussichtlich auch statistisch signifikante Erkenntnisse zu den Anforderungen und Verhaltensweisen bei der Link-Navigation gewinnen.

Ein verwandtes Forschungsfeld, das in dieser Arbeit bewusst unberücksichtigt blieb, sind Web-Applikationen und die Möglichkeiten zur Optimierung der Benutzbarkeit und Barrierefreiheit von Links in solchen Systemen. Bei Web-Applikationen spielen zwar Formu-

---

[352] Eine mögliche Umsetzung für das Vorlesen solcher Zusatzinformationen in Webseiten wird in (Sato, Zhu et al. 2011) vorgeschlagen und evaluiert.

[353] Ein Beispiel für die Möglichkeiten solcher Techniken zeigt Apples *VoiceOver*. Es erlaubt sehbehinderten Benutzern bei Touchscreens (z. B. beim *iPhone*) mit dem Finger den Bildschirm abzutasten, wobei ihnen vorgelesen wird, was sich gerade unter ihrem Finger befindet. Mehr Informationen zu *VoiceOver* für Geräte mit iOS und OS-X findet man unter: http://www.apple.com/de/accessibility/voiceover/.

lare eine größere Rolle als bei der Nutzung des Webs als Hypertext-Informationssystem (s. Abschnitt 2.2.4), dennoch bleiben Links unverzichtbar und dienen neben der Navigation im System dazu, Aktionen auszulösen. Beispielsweise ändern Links den Zustand des Systems oder laden nur Teile einer Seite neu; das Paradigma der Webseite als Navigationseinheit geht auf diese Weise verloren. Auch wenn HTML5, JavaScript und Ajax dazu beitragen, dass sich viele Aufgaben wesentlich effizienter mit dem Webbrowser erledigen lassen, bringen sie neue Probleme mit sich; so funktionieren viele wichtige Browser-Funktionen nicht mehr auf gewohnte Weise, wie der Back-Button, Bookmarks oder das Speichern und Drucken von Seiten (s. Abschnitt 3.1.7). Da Web-Applikationen und der Einsatz von JavaScript und Ajax eine immer bedeutendere Rolle spielen, sollten zukünftige Forschungsprojekte die entsprechenden Möglichkeiten der hier verwendeten „Links“ explizit berücksichtigen.

Nicht zuletzt führen die rasanten Entwicklungen beim mobilen Zugriff auf das Web und die neuen Schnittstellenkonzepte von Multi-Touch-Oberflächen zu neuen Herausforderungen bei der Link-Navigation, die potenziell noch gravierender sind als die bei der Nutzung des Webs auf Desktop-Systemen (s. Abschnitte 9.3.1 und 9.3.2). Aus diesen Gründen wäre es angebracht, für *Mobilgeräte* und *Natural User Interfaces* angepasste Konzepte zur Reduktion des Informationsdefizits bei der Link-Navigation zu entwickeln und zu evaluieren, die den spezifischen Herausforderungen dieser Systeme gerecht werden. Für die Realisierung entsprechender Prototypen sind die in dieser Arbeit vorgestellten Intermediary-Konzepte (s. Kapitel 8.4.3) prädestiniert, da Browser von Smartphones und Tablet-PCs in der Regel keine Programmschnittstellen für Browser-Erweiterungen bieten, wie man sie inzwischen bei den meisten Browsern für Desktop-Systeme findet. Der intermediary-basierte Ansatz von Scone eignet sich auch für Browser ohne Erweiterungsschnittstelle und erlaubt so die prototypische Realisierung vieler Arten von Navigationshilfen. Die neuen Möglichkeiten von HTML5 (wie Websockets, s. Kapitel 8.6.5.1) und Ajax können die Entwicklung entsprechender Prototypen wesentlich vereinfachen. Dieser kurze Ausblick vermittelt einen Eindruck davon, dass die Benutzbarkeit des Webs und von Hyperlinks auch zukünftig ein interessantes Forschungsfeld bleibt, bei dem sich stets neue Herausforderungen ergeben.

10 Literaturverzeichnis

Anmerkung: Alle URIs in dieser Arbeit und diesem Verzeichnis wurden Ende Oktober 2011 oder später auf Gültigkeit überprüft. Es wird daher nicht gesondert für jede Adresse der Zeitpunkt des letzten Zugriffs aufgeführt.

Bei Ressourcen, die inzwischen nicht mehr zugreifbar sind, wird zusätzlich die entsprechenden Adresse des „Internet Archive“ aufgeführt.

Abrams, David, Ron Baecker & Mark Chignell (1998):
Information Archiving with Bookmarks: Personal Web Space Construction and Organization.
Conference on Human Factors in Computer Systems (CHI'98), Los Angeles, USA. ACM Press: 41-48.

Acharya, Sanjay (2010): **The World in 2010: ICT Facts and Figures.**
International Telecommunication Union (ITU) (Geneva, Switzerland).
http://www.itu.int/net/pressoffice/press_releases/2010/39.aspx

Adar, Eytan, Jaime Teevan, Susan T. Dumais & Jonathan L. Elsa (2009):
The Web Changes Everything: Understanding the Dynamics of Web Content.
Second ACM International Conference on Web Search and Data Mining (WSDM 2009), Barcelona, Spanien. ACM Press: 282-291.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.1136

Adida, Ben, Mark Birbeck, Shane McCarron & Steven Pemberton (2008):
RDFa in XHTML: Syntax and Processing.
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/rdfa-syntax/

Adkisson, Heidi (2002): **Identifying De-Facto Standards for E-Commerce Web Sites**.
Master's Thesis. Technical Communication Department, University of Washington (Washington, DC, USA): 77 S.
http://www.hpadkisson.com/papers/index.html

Adya, Atul (1999): **Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions**. *Doctoral Thesis*. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (Cambridge, USA): 198 S.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.1744

Ahlberg, Christopher, Christopher Williamson & Ben Shneiderman (1992):
Dynamic Queries for Information Exploration: An Implementation and Evaluation.
Conference on Human Factors in Computing Systems (CHI'92), Monterey, USA. ACM Press: 619-626.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.8828

Akscyn, Robert M., Douglas L. McCracken & Elise Yoder (1988, Juli):
KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations.
Communications of the ACM, 31(7): 820-835.
http://dl.acm.org/citation.cfm?doid=48511.48513

Albers, Michael C. & Eric Bergman (1995): **The Audible Web: Auditory Enhancements for Mosaic**.
Computer Human Interaction CHI'95, Palo Alto, California, USA. ACM Press: 318-319.

Altheim, Murray, Frank Boumphrey, Sam Dooley, Shane McCarron, et al. (2001):
Modularization of XHTML.
W3C Recommendation 10 April 2001. World Wide Web Consortium.
http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/

Altheim, Murray, Frank Boumphrey, Sam Dooley, Shane McCarron, et al. (2010):
XHTML™ Modularization 1.1 - Second Edition.
W3C Recommendation 29 July 2010. World Wide Web Consortium.
http://www.w3.org/TR/xhtml-modularization/

Alvestrand, Harald Tveit (2001, Januar):
RFC 3066 - Tags for the Identification of Languages.
Network Working Group.
http://www.faqs.org/rfcs/rfc3066.html

Amento, Brian, Loren Terveen & Will Hill (2000):
Does "Authority" Mean Quality? Predicting Expert Quality Ratings of Web Documents.
ACM SIGIR conference on Research and development in information retrieval, Athens, Greece.
ACM Press: 296-303.
http://dl.acm.org/citation.cfm?doid=345508.345603

Amitay, Einat, David Carmel, Adam Darlow, Ronny Lempel & Aya Soffer (2003):
The Connectivity Sonar: Detecting Site Functionality by Structural Patterns.
Hypertext 2003, Nottingham, UK. ACM Press: 38-47.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.108.2624

Anderson, Kenneth M. (1997): **Integrating Open Hypermedia Systems with the World Wide Web**.
Hypertext'97, Southampton, UK. ACM Press: 157-166.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.4591

Andrews, Keith (1996): **Browsing, Building, and Beholding Cyberspace**.
Ph.D. Thesis. Institute for Information processing and Computer supported new Media (IICM), Graz University of Technology (Austria): 156 S.
http://ftp.iicm.tugraz.at/pub/keith/phd/andrews-1996-phd.pdf

Ansel Suter, Bettina (1995):
Hyperlinguistics. Hypertext-Lernumgebungen im akademischen Kontext: Eine Fallstudie.
Dissertation. Zentralstelle der Studentenschaft (Zürich, Schweiz): 212 S.

Apple Computer, Inc. (2011): **Apple Human Interface Guidelines.**
Apple Computer, Inc. (Cupertino, California, USA).
http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html

Ard, Scott & Steven Musil (2001, 27. Juni): **Microsoft Clips Windows XP Smart Tags.**
CNET Networks, Inc. (San Francisco, USA).
http://news.com.com/2100-1001_3-269167.html

Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, et al. (2009, 10 Februar):
Above the Clouds: A Berkeley View of Cloud Computing.
Technical Report. Electrical Engineering and Computer Sciences, University of California (Berkeley, CA, USA). 23 S.

ATIS (2011): **ATIS-0100523.2011: American National Standard.**
Alliance for Telecommunications Industry Solutions (ATIS).
http://www.atis.org/glossary/definition.aspx?id=4529

Atterer, Richard, Monica Wnuk & Albrecht Schmidt (2006): **Knowing the User's Every Move - User Activity Tracking for Website Usability Evaluation and Implicit Interaction**.
15th International World Wide Web Conference, Edinburgh, UK. ACM Press: 203-212.
http://dl.acm.org/citation.cfm?doid=1135777.1135811

Axelsson, Jonny, Beth Epperson, Masayasu Ishikawa, Shane McCarron, et al. (2010):
XHTML™ 2.0: W3C Working Group Note.
W3C Working Group Note. World Wide Web Consortium.
http://www.w3.org/TR/xhtml2

Ayers, Eric Z. & John T. Stasko (1995): **Using Graphic History in Browsing the World Wide Web**.
Fourth International World Wide Web Conference, Boston, USA: 259-270.
http://www.w3.org/Conferences/WWW4/Papers2/270/

Bacon, Jean (1998): **Concurrent Systems: Operating Systems, Database and Distributed Systems - An Integrated Approach**. Addison Wesley (Harlow, England). 752 S.

Bagemihl, Sven, Alexandra Kühte, Christian Herp & Harald Fritzsch (2010):
Generation Netzwerk – Die Jugendstudie von VZnet und iq digital.
VZnet Netzwerke Ltd., iq Digital Media Marketing GmbH (Berlin, Düsseldorf).
http://static.pe.meinvz.net/media/de/newsletter/2010_Generation_Netzwerk.pdf

Baier, Tobias, Harald Weinreich & Frank Wollenweber (2004):
Verbesserung von Social Navigation durch Identitätsmanagement.
Mensch & Computer 2004: Allgegenwärtige Interaktion, Paderborn.
Oldenbourg Wissenschaftsverlag: 189-198.
http://vsis-www.informatik.uni-hamburg.de/publications/viewpub.phtml/205

Baier, Toby (2005): **Persönliches digitales Identitätsmanagement**.
Dissertation. Verteilte Systeme und Informationssysteme, Fachbereich Informatik, Universität Hamburg, Deutschland.
http://vsis-www.informatik.uni-hamburg.de/publications/viewThesis.php/298

Bannon, Liam J. & Kaj Grønbæk (1989):
Hypermedia: Support for a more natural information organization.
7th Nordic Conference for Information and Documentation, Lyngby, Denmark: 15 S.

Bannon, Liam J. & Claire O'Malley (1984):
Problems in Evaluation of Human-Computer Interfaces: A Case Study.
Human-Computer Interaction - INTERACT '84. Elsevier Science Publishers.

Bardini, Thierry (2000):
Bootstrapping: Douglas Engelbart, Coevolution, and the Origins of Personal Computing.
Stanford University Press: 312 S.

Baron, Lisa, Jean Tague-Sutcliffe, Mark T. Kinnucan & Tom Carey (1996):
Labeled, Typed Links as Cues when Reading Hypertext Documents.
Journal of the American Society for Information Science, 47(12): 896-908.

Barrett, Rob, Paul Maglio, Jörg Meyer, Steve Ihde & Stephen Farrell (2000):
WBI Development Kit for Java. *(Java Software)*
IBM Almaden Research Centre, USA.
http://www.alphaworks.ibm.com/tech/wbidk , (seit Mai 2008 im Internet Archive verfügbar:
http://web.archive.org/web/20080514102342/http://www.alphaworks.ibm.com/tech/wbidk)

Barrett, Rob & Paul P. Maglio (1998):
Intermediaries: New Places for Producing and Manipulating Web Content.
Computer Networks and ISDN Systems, 30(1-7): 509-518.
http://www.almaden.ibm.com/cs/wbi/papers/www7/intermediaries.html

Barrett, Rob, Paul P. Maglio & Daniel C. Kellem (1997): **How to Personalize the Web**.
Conference on Human Factors in Computing Systems (CHI'97), Atlanta, Georgia, United States. ACM Press: 75 - 82.
http://dl.acm.org/citation.cfm?doid=258549.258595

Beckett, Dave (2004): **RDF/XML Syntax Specification (Revised).**
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/REC-rdf-syntax/

Bederson, Benjamin B., James D. Hollan, Jason Stewart, David Rogers, et al. (1996):
A Zooming Web Browser.
SPIE Multimedia Computing and Networking Conference, San Jose, USA: 260-271.
http://www.cs.umd.edu/hcil/pad++/papers/spie-96-webbrowser/spie-96-webbrowser.pdf

Beeman, Wiliam O., Kenneth T. Anderson, Gail Bader, James Larkin, et al. (1987):
Hypertext and Pluralism: From Lineal to Non-lineal Thinking.
Conference on Hypertext and Hypermedia, Chapel Hill, USA. ACM Press: 67-88.
http://dl.acm.org/citation.cfm?doid=317426.317434

Belkin, Nicholas J. & W. Bruce Croft (1987): **Retrieval Techniques**.
Annual Review of Information Science and Technology, 22: 109-145.

Benson, Edward, Adam Marcus, David Karger & Samuel Madden (2010):
Sync Kit: A Persistent Client-Side Database Caching Toolkit for Data Intensive Websites.
World Wide Web Conference 2010, Raleigh, North Carolina, USA. ACM Press: 121-130.
http://db.csail.mit.edu/pubs/sync-kit.pdf

Berghel, Hal (1997, Februar): **Cyberspace 2000: Dealing with Information Overload**.
Communications of the ACM, 40(2): 19-24.

Bergman, Michael K. (2001): **The Deep Web: Surfacing Hidden Value**.
The Journal of Electronic Publishing, 7(1).
http://quod.lib.umich.edu/cgi/t/text/text-idx?c=jep;view=text;rgn=main;idno=3336451.0007.104

Bernard, Michael L. (2002):
Examining User Expectations for the Location of Common E-Commerce Web Objects.
Human Factors and Ergonomics Society Annual Meeting, Baltimore, Maryland, USA. Human Factors and Ergonomics Society.

Bernard, Rory, Richard Crowder, Ian Heath & Wendy Hall (1995):
Authoring a Large Scale Industrial Hypermedia Application: A Case Study.
International Workshop on Hypermedia Design, Montpellier, France. Springer: 30-31.

Berners-Lee, Tim (1989, März): **Information Management: A Proposal.**
CERN Internal Communication.
http://www.w3.org/History/1989/proposal.html

Berners-Lee, Tim (1996, Juli): **The Web Maestro: An Interview with Tim Berners-Lee**.
MIT Technology Review, 99(5): 32-40.

Berners-Lee, Tim (1998a): **Style Guide for Online Hypertext.**
World Wide Web Consortium.
http://www.w3.org/Provider/Style/

Berners-Lee, Tim (1998b, 7. Mai): **The World Wide Web: A Very Short Personal History.**
World Wide Web Consortium.
http://www.w3.org/People/Berners-Lee/ShortHistory.html

Berners-Lee, Tim (2006, 28.August): **Tim Berners-Lee, originator of the Web and director of the World Wide Web Consortium, talks about where we've come, and about the challenges and opportunities ahead.**
Podcast-Interview (Text Transkription).
http://www-128.ibm.com/developerworks/podcast/dwi/cm-int082206.txt

Berners-Lee, Tim (2010): **Tim Berners-Lee: Personal Homepage and Biography.**
World Wide Web Consortium.
http://www.w3.org/People/Berners-Lee/

Berners-Lee, Tim & Dan Connolly (1995): **HTML 2.0 Specification.**
World Wide Web Consortium.
http://www.w3.org/MarkUp/html-spec/

Berners-Lee, Tim, Roy T. Fielding & Larry Masinter (2005, Januar):
RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax.
Network Working Group.
http://tools.ietf.org/rfc/rfc3986.txt

Berners-Lee, Tim & Mark Fischetti (2000):
Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web.
Harper Collins (New York, USA). 246 S.

Berners-Lee, Tim, James Hendler & Ora Lassila (2001, Mai): **The Semantic Web**.
Scientific American, 284(5): 34-43.
http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

Bernstein, Mark (1996): **HypertextNow: Showing Links.**
Eastgate Systems Inc. (Watertown, MA, USA).
http://www.eastgate.com/HypertextNow/archives/ShowingLinks.html

Bernstein, Mark (1997): **HypertextNow - Link Types: A Second Look.**
Eastgate Systems Inc. (Watertown, MA, USA).
http://www.eastgate.com/HypertextNow/archives/Trigg.html

Bernstein, Mark, Jay David Bolter, Michael Joyce & Elli Mylonas (1991):
Architectures for Volatile Hypertext.
Hypertext '91, San Antonio, Texas, USA. ACM Press: 243-260.

Bevan, Nigel, Carol Barnum, Gilbert Cockton, Jakob Nielsen, et al. (2003):
The "magic number 5": is it enough for web testing?
CHI '03 Conference on Human Factors in Computing Systems (Extended Abstracts), Ft. Lauderdale, Florida, USA. ACM Press: 698 - 699.
http://dl.acm.org/citation.cfm?doid=765891.765936

Bharat, Krishna, Andrei Broder, Monika Henzinger, Puneet Kumar & Suresh Venkatasubramanian (1998): **The Connectivity Server: fast access to linkage information on the Web**.
Computer Networks and ISDN Systems, 30(1-7): 469-477.

Bieber, Michael, Fabio Vitali, Helen Ashman, V. Balasubramanian & Harri Oinas-Kukkonen (1997a):
Fourth Generation Hypermedia: Some Missing Links for the World Wide Web.
Int. Journal on Human-Computer Studies, World Wide Web Usability Special Issue, 47(1): 31-65.

Bieber, Michael, Fabio Vitali, Helen Ashman, V. Balasubramanian & Harri Oinas-Kukkonen (1997b):
Some Hypermedia Ideas for the WWW.
30th HICSS Conf., Wailea (HA). IEEE Computer Society Press: 309-319.

Biron, Paul V., Kaiser Permanente & Ashok Malhotra (2004):
XML Schema Part 2: Datatypes Second Edition.
World Wide Web Consortium.
http://www.w3.org/TR/xmlschema-2/

BITCOM (2011, 31. März): **Besonders ältere Menschen fühlen sich von Informationen überflutet.**
Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (Berlin).
http://www.bitkom.org/de/presse/8477_67508.aspx

BitDefender (2010, 7. Dezember): **BitDefender Study Finds Majority of Social Network Users Do Not Check Shared Links for Malware** *Pressemitteilung*. BitDefender SRL.
http://news.bitdefender.com/NW1922-en--BitDefender-Study-Finds-Majority-of-Social-Network-Users-Do-Not-Check-Shared-Links-for-Malware.html

BITV2 (2011, 12. September):
Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie Informationstechnik-Verordnung - BITV 2).
Beauftragter der Bundesregierung für die Belange behinderter Menschen.
Bundesanzeiger Verlag (Bonn): 1843-1859.
http://www.bgbl.de/Xaver/start.xav?startbk=Bundesanzeiger_BGBl

Blackmon, Marilyn Hughes, Peter G. Polson, Muneo Kitajima & Clayton Lewis (2002):
Cognitive Walkthrough for the Web.
Conf. on Human Factors in Computing Systems, Minneapolis, Minnesota, USA. ACM Press: 463-470.
http://dl.acm.org/citation.cfm?doid=503376.503459

Blair, David C. & M.E. Maron (1985, März):
An Evaluation of Retrieval Effectiveness for a Full-Text Document Retrieval System.
Communications of the ACM, 28(3): 289-299.
http://dl.acm.org/citation.cfm?doid=3166.3197.

Blair, David C. & M.E. Maron (1990):
Full-Text Information Retrieval: Further Analysis and Carification.
Information Processing and Management, 26(3): 437-447.
http://dx.doi.org/10.1016/0306-4573(90)90102-8

Bleich, Holger (2005, 18. April): **Web-Fehler**. *c't - Magazin für Computertechnik*(9): 92-93.

Boag, Scott, Don Chamberlin, Mary F. Fernández, Daniela Florescu, et al. (2011, Januar):
XQuery 1.0: An XML Query Language (Second Edition).
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/xquery/

Boiko, Bob (2001): **Content Management Bible**.
Wiley & Sons Inc. (Indianapolis). 816 S.

Bortz, Jürgen & Nicola Döring (1995): **Forschungsmethoden und Evaluation**.
Springer (Berlin, Heidelberg). 768 S.

Bos, Bert (2011): **Cascading Style Sheets - Home Page.**
World Wide Web Consortium.
http://www.w3.org/Style/CSS/

Botafogo, Rodrigo A. & Ben Shneiderman (1991):
Identifying Aggregates in Hypertext Structures. *Hypertext*. ACM Press: 63-74.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.7481

Bouthors, Vincent & Olivier Dedieu (1999):
Pharos, a Collaborative Infrastructure for Web Knowledge Sharing.
3rd European Conf. on Research and Advanced Technology for Digital Libaries, ECDL'99.
Springer Verlag: 215-233.
ftp://ftp.inria.fr/INRIA/publication/RR/RR-3679.pdf

Bouvin, Niels Olof (1999): **Unifying Strategies for Web Augmentation**. *Hypertext '99*, Darmstadt.
ACM Press: 91-100.

Bouvin, Niels Olof (2000): **Augmenting the Web through Open Hypermedia**. *Ph.D. Thesis*.
Department of Computer Science, Aarhus University (Aarhus): 150 S.
http://dx.doi.org/10.1080/13614560208914733

Bowman, C. Mic, Peter B. Danzig, Darren R. Hardy, Udi Manber & Michael F. Schwartz (1995):
The Harvest Information Discovery and Access System.
Computer Networks and ISDN Systems, 28(1-2): 119-125.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.1842

Boyns, Mark R. (1998): **Design and Implementation of a World Wide Web Filtering System**. *Master Thesis*. Computer Science, San Diego State University (San Diego, USA): 80 S.
http://muffin.doit.org/

Braden, Roberts A. (1996): **Visual Literacy**, in David H. Jonassen (ed.): "*Handbook of Research on Educational Communications and Technology*". Simon & Schuster Macmillian (New York): 491-520.

Bradner, Scott (1997, März): **RFC 2119: Key Words for Use in RFCs to Indicate Requirement Levels.**
Network Working Group / Harvard University.
http://www.rfc-editor.org/rfc/rfc2119.txt

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, et al. (2006):
Extensible Markup Language (XML) 1.1. *W3C Recommendation*. World Wide Web Consortium.
http://www.w3.org/TR/xml11/

Brewington, Brian E. & George Cybenko (2000): **How Dynamic is the Web?**
Computer Networks, 33(1-6): 257-276.
http://www9.org/w9cdrom/264/264.html

Brin, Sergey & Lawrence Page (1998):
The Anatomy of a Large-Scale Hypertextual Web Search Engine.
Computer Networks and ISDN Systems, 30(1-7): 107-117.
http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Brooks, Charles, Murray S. Mazer, Scott Meeks & Jim Miller (1995):
Application-Specific Proxy Servers as HTTP Stream Transducers.
Fourth International World Wide Web Conference, Boston, USA: 539-548.
http://www.w3.org/pub/Conferences/WWW4/Papers/56

Brown, Peter J. (1987): **Turning Ideas into Products: The Guide System**.
Hypertext '87, Chapel Hill, NC. ACM Press: 34-45.
http://dl.acm.org/citation.cfm?doid=317426.317430

Brownell, David (2002): **SAX2**.
O'Reilly (Sebastopol, California, USA). 240 S.

Brumen, Bostjan, Tomaz Domajnko, Matjaz B. Juric & Ivan Rozman (1999):
How to store Java Objects. *Java Report*, 4(4): 33-45.

Brusilovski, Peter (1996):
Adaptive Hypermedia: an Attempt to Analyze and Generalize,
in "*Multimedia, Hypermedia, and Virtual Reality.*" Springer-Verlag (Berlin), Vol. 1077: 288-304.

Bry, François & Michael Eckert (2005): **Processing link structures and linkbases in the web's open world linking**. *Conference on Hypertext and Hypermedia*, Salzburg, Austria. ACM Press: 134-144.
http://dl.acm.org/citation.cfm?doid=1083356.1083383

BSI (2011): **Die Lage der IT-Sicherheit in Deutschland 2011.**
Bundesamt für Sicherheit in der Informationstechnik - BSI (Bonn).
https://www.bsi.bund.de/DE/Publikationen/Lageberichte/lageberichte_node.html

Buchmann, Volkert (2002): **Multiple Links im World-Wide Web: Entwicklung eines Client/ Server-Navigationswerkzeuges mit dem Framework Scone**.
Studienarbeit. Fachbereich Informatik, Universität Hamburg: 59 S.
http://vsis-www.informatik.uni-hamburg.de/getDoc.php/thesis/23/Studienarbeit_final_print.pdf

Bulterman, Dick, Jack Jansen, Sjoerd Mullender, Marisa DeMeglio, et al. (2008):
Synchronized Multimedia Integration Language (SMIL 3.0).
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/SMIL3/

Bumann, Sigrid & Sandi Bilinski (1998): "**Lost in History" - Entwicklung von Strategien zur Minimierung der Orientierungs- und Navigationsschwierigkeiten in besuchten WWW-Seiten**.
Diplomarbeit. Fachbereich Informatik, Universität Hamburg.

Bundesnetzagentur (2010): **Zuordnung ersteigerter Frequenzblöcke.** Bundesnetzagentur (Bonn).
http://www.bundesnetzagentur.de/cln_1911/SharedDocs/Pressemitteilungen/DE/2010/100830_VerlosungErsteigerteFrequBloecke.html?nn=65116

Burbules, Nicholas C. (1997): **Rhetorics of the Web: Hyperreading and Critical Literacy**, in Ilana Snyder (ed.): "*Page to Screen: Taking Literacy Into the Electronic Era*".
Allen and Unwin (New South Wales, Australia): 102-122.
http://faculty.ed.uiuc.edu/burbules/papers/rhetorics.html

Burg, Frederick Murray & Max S. Schoeffler (2002): **Method And System For Graphic Display Of Link Actions**. *US Patent Nr. 6,362,840* (United States).

Bush, Vannevar (1945a): **As We May Think**. *The Atlantic Monthly*, July 1945;
reprinted in: Interactions, (III)2, 1996: 35-46.

Bush, Vannevar (1945b): **As We May Think: A Top U.S. Scientist Forsees a Possible Future World in Which Man-made Machines Will Start to Think**. *Life Magazine*, 19(11): 112-124.

Butler, Mark H., Johan Hjelm & Kazuhiro Kitagawa (2004): **CC/PP Information Page.**
World Wide Web Consortium.
http://www.w3.org/Mobile/CCPP/

Buur, Jacob & Kirsten Bagger (1999, Mai 1999): **Replacing Usability Testing with User Dialogue**.
Communications of the ACM, 42(5): 63-66.
http://dl.acm.org/citation.cfm?doid=301353.301417.

Buzan, Tony & Barry Buzan (2002): **Das Mind-Map-Buch: Die beste Methode zur Steigerung ihres geistigen Potentials**. Moderne Verlagsgesellschaft Mvg.: 309 S.

Byrd, Donald (1999): **A Scrollbar-based Visualization for Document Navigation**.
Digital Libraries '99, Berkley, CA. ACM Press: 122-129.

Cailliau, Robert & Helen Ashman (2000): **Hypertext in the Web - a History**.
ACM Computing Surveys, 31(4).
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/62.html

Caldwell, Ben, Michael Cooper, Loretta Guarino Reid & Gregg Vanderheiden (2008):
Web Content Accessibility Guidelines (WCAG) 2.0.
W3C Recommendation 11 December 2008. W3C.
http://www.w3.org/TR/2008/REC-WCAG20-20081211/

Campbell, Brad & Joseph M. Goodman (1988, Juli): **HAM: A General Purpose Hypertext Abstract Machine**. *Communications of the ACM*, 31(7): 856-861.
http://dl.acm.org/citation.cfm?doid=317426.317429

Campbell, Christopher & Paul Maglio (1999): **Facilitating Navigation in Information Spaces: Road-Signs on the World Wide Web**. International Journal of Human-Computer Studies, 50: 309-327.

Cao, Pei & Chengjie Liu (1998): **Maintaining Strong Cache Consistency in the World-Wide Web**.
IEEE Transactions on Computers, 47(4): 445-457.

Capurro, Rafael (1978): **Information: Ein Beitrag zur etymologischen und ideengeschichtlichen Begründung des Informationsbegriffs**. *Doktorarbeit*, Universität Düsseldorf: 320 S.
http://www.capurro.de/info.html

Capurro, Rafael & Birger Hjørland (2003): **The Concept of Information**.
Annual Review of Information Science and Technology, 37: 343-411.
http://www.capurro.de/infoconcept.html

Card, Stuart K., Jock D. Mackinlay & Ben Shneiderman. (1999):
Readings in Information Visualization: Using Vision to Think.
Morgan Kaufmann Publishers (San Francisco, USA). 686 S.

Card, Stuart K., Allan Newell & Thomas P. Moran (1983):
The Psychology of Human-Computer Interaction.
Lawrence Erlbaum Associates (Mahwah, New Jersey, USA). 496 S.

Carr, Leslie A., David De Roure, Gary Hill & Wendy Hall (1996): **Web Links as User Artefacts**.
Technical Report. University of Southampton (Southampton, UK). 11 S.

Carr, Leslie A., David DeRoure, Wendy Hall & Gary Hill (1995):
The Distributed Link Service: A Tool for Publishers, Authors and Readers.
Fourth International World Wide Web Conference (WWW4): The Web Revolution, Boston, MA, USA.
O'Reilly & Associates: 647-656.
http://eprints.ecs.soton.ac.uk/archive/00000739/

Carr, Leslie A., David DeRoure, Wendy Hall & Gary Hill (1998):
Implementing an open link service for the World Wide Web.
World Wide Web Journal, 1(2): 61-71.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.3264

Catledge, Lara D. & James E. Pitkow (1995): **Characterizing Browsing Strategies in the World-Wide Web**. Computer Networks and ISDN Systems, 27(6): 1065-1073.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.8958

Catlin, Timothy, Paulette Bush & Nicole Yankelovich (1989):
Internote: Extending a hypermedia framework to support annotative collaboration.
Hypertext '89, Pittsburgh, PA, USA. ACM Press: 365-378.

Cattell, R. G. G., Douglas K. Barry, Mark Berler, Jeff Eastman, et al. (2000): **The Object Data Standard ODMG 3.0**. Morgan Kaufman Publishers (San Francisco, CA, USA). 292 S.

Çelik, Tantek, Bert Bos & Daniel Glazman (2004): **CSS3 Hyperlink Presentation Module.** World Wide Web Consortium. http://www.w3.org/TR/css3-hyperlinks/

Çelik, Tantek, Matthew Mullenweg & Eric Meyer (2006): **XFN 1.1: XHTML Friends Network.** Global Multimedia Protocols Group. http://gmpg.org/xfn/

CERN (2008): **http://info.cern.ch: The Website of the World's First-Ever Web Server.** CERN - European Organization for Nuclear Research (Geneva, Switzerland). http://info.cern.ch/

Chakrabarti, Soumen, David Gibson & Kevin McCurley (1999): **Surfing the Web Backwards**. Computer Networks and ISDN Systems (8th International WWW Conference, Toronto, Canada), 31(11-16): 1679-1693. http://www8.org/w8-papers/5b-hypertext-media/surfing/surfing.html

Chan, Sharon (2010): **MIX10: The Bing blue color that's worth $80 million.** The Seattle Times (Seattle, Washington, USA). http://seattletimes.nwsource.com/html/microsoftpri0/2011361493_mix10_the_bing_blue_color_thats_worth_80_million.html

Chappell, David (1996): **Understanding ActiveX and OLE**. Microsoft Press (Redmond, Washington). 328 S.

Chartier, Roger & Guglielmo Cavallo (1999): **Die Welt des Lesens. Von der Schriftrolle zum Bildschirm**. Campus Verlag (Frankfurt / New York). 668 S.

Charzinski, Joachim (2010): **Traffic Properties, Client Side Cachability and CDN Usage of Popular Web Sites**. *15th International GI/ITG Conference, MMB&DFT 2010*, Essen, Deutschland. Springer Verlag: 136-150. http://www.springerlink.com/content/p50n774244236p77/

Chen, Jay, Lakshminarayanan Subramanian & Kentaro Toyama (2009): **Web Search and Browsing Behavior under Poor Connectivity**. *27th International Conference Human Factors in Computing Systems - CHI 2009*, Boston, MA, USA. ACM Press: 3473-3478. http://dl.acm.org/citation.cfm?doid=1520340.1520505

Chetty, Marshini, Richard Banks, A. J. Bernheim Brush, Jonathan Donner & Rebecca E. Grinter (2011, März & April): **While the Meter is Running: Computing in a Capped World**. *Interactions*, 18(2): 72-75. http://dl.acm.org/citation.cfm?doid=1925820.1925836

Chisholm, Wendy, Gregg Vanderheiden & Ian Jacobs (1999, 5. Mai): **Web Content Accessibility Guidelines 1.0.** *W3C Recommendation*. World Wide Web Consortium. http://www.w3.org/TR/WCAG10/

Christen, Michael (2005, 8. Dezember): **YaCy - Peer-to-Peer Web-Suchmaschine**. *Die Datenschleuder*, 86: 54-57. http://chaosradio.ccc.de/media/ds/ds086.pdf.

Christensen, Bent Guldbjerg & Frank Allan Hansen (2002): **XLink - Linking the Web and Open Hypermedia**. *International Workshop on Open Hypermedia Systems Core Concepts and Research Directions at HT'02*, University of Maryland, USA: 9-18. http://www.daimi.au.dk/~bentor/papers/XLink-Hypertext02.pdf

Christensen, Bent Guldbjerg, Frank Allan Hansen & Niels Olof Bouvin (2003):
Xspect: Bridging Open Hypermedia and XLink.
12th International World Wide Web Conference: WWW 2003, Budapest, Ungarn. ACM Press: 490-499.
http://dl.acm.org/citation.cfm?doid=775152.775222

Christensen, Erik, Francisco Curbera, Greg Meredith & Sanjiva Weerawarana (2001):
Web Services Description Language (WSDL) 1.1. World Wide Web Consortium.
http://www.w3.org/TR/wsdl

Christian, Eliot (2000): **Global Information Locator Service (GILS).**
GILS Maintenance Agency of the U.S. Geological Survey.
http://www.gils.net/

Chu, Heting & Marilyn Rosenthal (1996): **Search Engines for the World Wide Web: A Comparative Study and Evaluation Methodology**. *ASIS 1996 Annual Conference*, Baltimore, USA.
American Society for Information Science: 127-135.
http://www.asis.org/annual-96/ElectronicProceedings/chu.html

Ciancarini, Paolo, Federico Folli, Davide Rossi & Fabio Vitali (2002):
XLinkProxy: External Linkbases with XLink.
ACM Symposium on Document Engineering, McLean, Virginia, USA. ACM Press: 57-65.
http://dl.acm.org/citation.cfm?doid=585058.585070

Cisco (2010): **Third annual broadband study shows global broadband quality improves by 24% in one year.** Saïd Business School, Oxford University & Cisco Systems Inc. (London, UK).
http://newsroom.cisco.com/dlls/2010/prod_101710.html

Cisco (2011): **Cisco Visual Networking Index: Forecast and Methodology, 2010-2015.**
White Paper. Cisco Systems Inc. (San Jose, USA).
http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

Clark, James (1999, 16 November): **XSL Transformations (XSLT) - Version 1.0.**
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/xslt

Clark, James & Steve DeRose (1999, 16. November): **XML Path Language (XPath) - Version 1.0.**
World Wide Web Consortium.
http://www.w3.org/TR/xpath

Cleary, Chip & Ray Bareiss (1996): **Practical Methods for Automatically Generating Typed Links**.
Hypertext '96, Washington, DC. ACM Press: 31-41.

Clement, Luc, Andrew Hatley, Claus von Riegen & Tony Rogers (2004, 19. Oktober):
UDDI Version 3.0.2: Technical Committee Draft. OASIS Open.
http://www.uddi.org/pubs/uddi-v3.0.2-20041019.pdf

Cockburn, Andy & Saul Greenberg (1999): **Issues of Page Representation and Organisation in Web Browser's Navigation Tools**.
OzCHI'99 Australian Conference on Human Computer Interaction, Wagga Wagga Australia: 7-14.
http://www.cosc.canterbury.ac.nz/~andy/publs.html

Cockburn, Andy, Saul Greenberg, Steve Jones, Bruce McKenzie & Michael Moyle (2003):
Improving Web Page Revisitation: Analysis, Design, and Evaluation.
IT & Society, 1(3): 159-183.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.1451

Cockburn, Andy, Saul Greenberg, Bruce McKenzie, Michael Jasonsmith & Shaun Kaasten (1999): **WebView: A Graphical Aid for Revisiting Web Pages**. *OzCHI'99 Australian Conference on Human Computer Interaction*, Wagga Wagga Australia: 15-22. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.6167

Cockburn, Andy & Steve Jones (1997): **Design Issues for World Wide Web Navigation Visualisation Tools**. *Fifth Conference on Computer-Assisted Research of Information (RIAO'97)*, McGill University, Montreal, Quebec, Canada: 55-74. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.3590

Cockburn, Andy & Bruce McKenzie (2001): **What Do Web Users Do? An Empirical Analysis of Web Use**. International Journal of Human-Computer Studies, 54(6): 903-922. http://www.cosc.canterbury.ac.nz/andrew.cockburn/papers/ijhcsAnalysis.pdf

Cole, Jeffrey I. & et al. (2003, Januar): **The UCLA Internet Report - „Surveying the Digital Future"**. *Technical Report*. UCLA Center for Communication Policy (Los Angeles, USA). 89 S.

Collins, Allan M. & M. Ross Quillian (1969): **Retrieval Time from Semantic Memory**. Verbal Learning and Verbal Behavior, 8: 240-248.

comScore (2009): **Microsoft Sites Captures Largest Share of Time Spent Online Worldwide** comScore, Inc. (Reston, VA, USA). http://comscore.com/ger/Press_Events/Press_Releases/2009/11/Microsoft_Sites_Captures_Largest_Share_of_Time_Spent_Online_Worldwide

Conklin, Jeff (1987): **Hypertext: An Introduction and Survey**. IEEE Computer, 20(9): 17-41.

Conklin, Jeff & Michael L. Begeman (1987): **gIBIS: A Hypertext Tool for Exploratory Policy Discussion**. ACM Transactions on Information Systems (TOIS), 6(4): 303-331. http://dl.acm.org/citation.cfm?doid=58566.59297

Connolly, Dan, Robert Cailliau & Tim Berners-Lee (2000): **A Little History of the World Wide Web.** World Wide Web Consortium. http://www.w3.org/History.html

Consortium, TEI (2004): **TEI - Text Encoding Initiative Website.** http://www.tei-c.org/

Coulouris, George F., Jean Dollimore & Tim Kindberg (2005): **Distributed Systems: Concepts and Design**. Addison Wesley (Harlow, England). 4. Ausgabe: 927 S.

Cox, Donald A., Jasdeep S. Chugh, Carl Gutwin & Saul Greenberg (1998): **The Usability of Transparent Overview Layers**. *CHI'98 Conference on Human Factors in Computing Systems, Companion Proceedings*, Los Angeles, USA. ACM Press: 301-302. http://dl.acm.org/citation.cfm?doid=286498.286777

Crane, Dave, Eric Pascarello & Darren James (2005): **Ajax in Action**. Manning Publications (Greenwich, Conneticut, USA). 650 S.

Crocker, David H. (1982): **RFC 822: Standard for the format of ARPA Internet text messages.** Dept. of Electrical Engineering, University of Delaware, USA. http://www.faqs.org/rfcs/rfc822.html

Cugini, John & Jean Scholtz (1999): **VISVIP: 3D Visualization of Paths through Web Sites**. *International Workshop on Web-Based Information Visualization (WebVis'99)*, Florence, Italy. IEEE Computer Society: 259-263. http://www.itl.nist.gov/iaui/vvrg/cugini/webmet/visvip/webvis-paper.html

Dalitz, Wolfgang & Gernot Heyer (1995): **Hyper-G – Das Internet-Informationssystem der 2. Generation**. d-punkt Verlag (Heidelberg, Deutschland).

Date, CJ (2003): **An Introduction to Database Systems**.
Addison-Wesley (Reading, Massachusetts, USA). 1024 S.

Daviel, Andrew (1996): **A Dictionary of HTML META Tags.**
Vancouver Webpages Inc.
http://vancouver-webpages.com/META/

Davis, Hugh (1999): **Hypertext Link Integrity**.
ACM Computing Surveys (CSUR), 31(4es): Artikel Nr. 28.
http://dl.acm.org/citation.cfm?doid=345966.346026

Davis, Hugh C, D. E. Millard, S. Reich, N. Bouvin, et al. (1999): **Interoperability between Hypermedia Systems: The Standardisation Work of the OHSWG**.
Hypertext 1999, Darmstadt, Deutschland. ACM Press: 201-202.
http://eprints.ecs.soton.ac.uk/7419/01/ohstb.pdf

Davis, Hugh C. (1995, August): **To Embed or Not to Embed**.
Communications of the ACM, 38(8): 108-109.

Davis, Hugh C., Simon J. Knight & Wendy Hall (1994): **Light hypermedia services: A study of third party application integration**. *Hypertext 1994*, Edinburgh, Scotland. ACM Press: 41-50.
http://eprints.ecs.soton.ac.uk/723/

Davis, Hugh, Wendy Hall, Ian Heath & Gary Hill (1992): **Towards an Integrated Information Environment With Open Hypermedia Systems**. *ECHT Conference*, Milano, Italy. ACM Press: 181-190.

Davis, Hugh, Sigi Reich & David Millard (1997):
A Proposal for a Common Navigational Hypertext Protocol.
Open Hypermedia Systems Working Group.
http://www.ecs.soton.ac.uk/~hcd/ohp/ohp35.htm

December, John (1994, October): **Challenges for Web Information Providers**.
Computer-Mediated Communication Magazine, 1(6): 8-14.
http://www.ibiblio.org/cmc/mag/1994/oct/webip.html.

Deering, Steve & Robert Hinden (1998):
Internet Protocol, Version 6 (IPv6) Specification - RFC 2460. The Internet Society.
http://www.faqs.org/rfcs/rfc2460.html

Delisle, Norman M. & Mayer D. Schwartz (1987, April): **Contexts - A Partitioning Concept for Hypertext**. *Transactions on Office Information Systems*, 5(2): 168-186.

Demunter, Christophe (2005): **Die digitale Kluft in Europa.**
European Statistical Data Support (EuroStat) (Luxemburg, Luxemburg).
http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-NP-05-038/DE/KS-NP-05-038-DE.PDF

DeRose, Steve, Eve Maler, David Orchard & Norman Walsh (2010):
XML Linking Language (XLink) Version 1.1.
W3C Recommendation, 6 May 2010. World Wide Web Consortium.
http://www.w3.org/TR/xlink11/

DeRose, Steven J. (1989): **Expanding the Notion of Links**. *Hypertext '89*, Pittsburgh, PA, USA.
ACM Press: 249-257.

DeRose, Steven J. (1998, 28. März): **FRESS: The File Retrieval and Editing SyStem.** *White Paper*.
http://www.derose.net/steve/writings/whitepapers/fress.html

DeRose, Steven J. & Andries van Dam (1999): **Document Structure and Markup in the FRESS Hypertext System [Alias: 'The Lost Books of Hypertext']**.
Markup Languages: Theory and Practice, 1(1): 7-32.

DeRose, Steven J., Ron Daniel, Paul Grosso, Eve Maler, et al. (2002, 16. August):
XML Pointer Language (XPointer). World Wide Web Consortium.
http://www.w3.org/TR/xptr/

DeRose, Steven J. & David J. Durand (1994): **Making Hypermedia Work: A User's Guide to HyTime**.
Kluwer Academic Publishers (Norwell, Massachusetts).

DeRose, Steven J., Eve Maler, David Orchard & Ben Trafford (2001, 27. Juni):
XML Linking Language (XLink) Version 1.0.
World Wide Web Consortium XLink Working Group.
http://www.w3.org/TR/xlink

Destatis (2009): **73% der privaten Haushalte haben einen Internetzugang.**
Statistisches Bundesamt (Wiesbaden, Deutschland).
http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2009/12/PD09__464__IKT,templateId=renderPrint.psml

Dieberger, Andreas, Paul Dourish, Kristina Höök, Paul Resnick & Alan Wexelblat (2000):
Social Navigation - Techniques for Building More Usable Systems.
Interactions of the ACM, 7(6): 36-45.
http://dl.acm.org/citation.cfm?doid=352580.352587.

Dierks, Tim & Christopher Allen (1999): **RFC 2246 - The TLS Protocol Version 1.0.** *RFC*.
Network Working Group. http://www.faqs.org/rfcs/rfc2246.html

Dijk, Teun Adrianus van & Walter Kintsch (1983): **Strategies of Discourse Comprehension**.
Academic Press (New York, USA). 418 S.

Dijkstra, Edsger W. (1968, März): **Go To Statement Considered Harmful**.
Communications of the ACM, 11(3): 147-148.
http://dl.acm.org/citation.cfm?doid=362929.362947.

Dikaiakos, Marios D. & Demetris Zeinalipour-Yazti (2001):
A Distributed Middleware Infrastructure for Personalized Services.
Technical Report. Dept. of Computer Science, University of Cyprus (Lefkosia, Republik Zypern).

Dix, Alan, Janet Finlay, Greory Abowd & Russell Beale (1993): **Human-Computer Interaction**.
Prentice Hall (New York, USA).

Donner, Jonathan (2010): **Framing M4D: The Utility of Continuity and the Dual Heritage of "Mobiles and Development"**.
The Electronic Journal of Information Systems in Developing Countries, 44(3): 1-16.
http://www.ejisdc.org/ojs2/index.php/ejisdc/article/viewFile/746/342

Donovan, Kevin & Jonathan Donner (2010):
A Note on the Availability (and Importance) of Pre-Paid Mobile Data in Africa.
2nd International Conference on Mobile Communication Technology for Development (M4D2010),
Kampala, Uganda. Karlstad University Press: 263-267.
http://www.jonathandonner.com/prepaydata_M4D.pdf

Doorenbos, Robert B., Oren Etzioni & Daniel S. Weld (1997):
A Scalable Comparison-Shopping Agent for the World-Wide Web.
First International Conference on Autonomous Agents, Marina del Rey, USA. ACM Press: 39-48.
http://dl.acm.org/citation.cfm?doid=267658.267666

Dorsch, Friedrich (1994): **Psychologisches Wörterbuch**. Huber (Bern, Schweitz).

Dourish, Paul & Matthew Chalmers (1994): **Running Out of Space: Models of Information Navigation**. *People and Computers IX, BCS HCI '94*, Glasgow. Cambridge University Press. http://www.dcs.gla.ac.uk/~matthew/papers/hci94.pdf

Draper, Stephen W. (2001): **A Note on the Hawthorne Effect**. Department of Psychology (Glasgow, UK).

Dubroy, Patrick & Ravin Balakrishnan (2010): **A Study of Tabbed Browsing Among Mozilla Firefox Users**. *28th International Conference on Human Factors in Computing Systems*, Atlanta, Georgia, USA. ACM Press: 673-682. http://dl.acm.org/citation.cfm?doid=1753326.1753426

DuCharme, Bob (2002a, 23. Oktober): **HTML A Attributes: What People Use.** *Web-Dokument*. http://www.snee.com/xml/a-href.html

DuCharme, Bob (2002b, 13. März): **XLink: Who Cares?** O'Reilly Media Inc. http://www.xml.com/pub/a/2002/03/13/xlink.html

DuCharme, Bob (2003, 1. Mai): **A Nineteenth-Century Linking Application.** O'Reilly Media Inc. http://www.oreillynet.com/pub/wlg/3153

Dumais, Joseph S. & Janice C. Redish (1993): **A Practical Guide to Usability Testing**. Intellect Ltd. (Westport, CT, USA). 412 S.

Duncker, Karl (1935): **Zur Psychologie des produktiven Denkens**. Springer (Berlin). 135 S.

Duncker, Karl (1945): **On problem-solving**. The American Psychological Association Inc. (Washington, DC), Vol. 58: 114 S.

Dunning, Ted (1994, 10. März): **Statistical Identification of Language**. *Technical Report*. Computing Research Lab (CRL), New Mexico State University (Las Cruces, New Mexico, USA). 31 S.

Eberleh, Edmund, Horst Oberquelle & Reinhard Opperman (1994): **Einführung in die Software-Ergonomie. Gestaltung graphisch-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen.** Walter de Gruyter (Berlin).

ECMA (1999): **Standard ECMA-262: ECMAScript Language Specification.** ECMA International. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

eco (2011, 31. August): **Safer Internet DE SIC: Annual Public Report (Geschäftsbericht).** LMK, LfM, eco, FSM, jugendschutz.net, NgK. eco - Verband der deutschen Internetwirtschaft e.V. (Köln). http://www.internet-beschwerdestelle.de/D1_4_1_Annual_Public_Report_T12.pdf

Edwards, Deborah M. & Lynda Hardman (1999): **Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment**, in Ray McAleese (ed.): "*Hypertext: Theory into Practice*". Intellect Books (Exeter, UK). 2nd Edition: 90-105.

Eichmann, David (1994): **The RBSE Spider - Balancing Effective Search Against Web Load**. *First International Conference on the World-Wide Web*, CERN, Genf, Schweiz. Elsevier Publishing: 113-120. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.7487

Eichmann, David (1995): **Ethical Web Agents**. Computer Networks and ISDN Systems, 28(1-2): 127-136. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.3585

El-Saddik, Abdulmotaleb, Carsten Griwodz & Ralf Steinmetz (1998): **Exploiting User Behaviour in Prefetching WWW Documents**. *Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, Oslo, Norway. Springer: 302-311. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/EGS98-1.html

Engelbart, Douglas Carl (1984): **Authorship provisions in Augment**. *Compcon*, San Francisco, California, USA. IEEE. http://www.dougengelbart.org/pubs/oad-2250.html

Engelbart, Douglas Carl & et al. (1968): **The NLS Demonstration on the Fall Joint Computer Conference held at the Convention Center in San Francisco.** *Real Media Files*. Augmentation Research Center at Stanford Research Institute. http://sloan.stanford.edu/mousesite/1968Demo.html

Ericsson, K. Anders & Herbert A. Simon (1993): **Protocol Analysis: Verbal Reports as Data - Rev'd Edition**. MIT Press (Cambridge, MA).

Etgen, Michael & Judy Cantor (1999): **What does getting WET (Web Event-logging Tool) Mean for Web Usability?** *5. Human Factors and the Web Conference*, Gaithersburg, Maryland, USA. http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/

Eurostat (2010): **Internetzugang und Internetnutzung im Jahr 2010.** Eurostat (Luxemburg). http://epp.eurostat.ec.europa.eu/cache/ITY_PUBLIC/4-14122010-BP/DE/4-14122010-BP-DE.PDF

Evenson, Shelley, John Rheinfrank, Fitch RichardsonSmith & Wendie Wulff (1989): **Towards a Design Language for Representing Hypermedia Cues**. *Hypertext 1989*, Pittsburgh, Pennsylvania. ACM Press: 83-92. http://dl.acm.org/citation.cfm?doid=74224.74231

Farhoomand, Ali F. & Don H. Drury (2002, Oktober): **Managerial Information Overload**. *Communications of the ACM*, 45(10): 127-131. http://dl.acm.org/citation.cfm?doid=570907.570909.

Felfoldi, Sophie (2005): **Digital Libraries: Metadata Resources.** International Federation of Library Associations and Institutions IFLA. http://www.ifla.org/II/metadata.htm

Fennah, Alison & Sarah Botterill (2010): **EIAA Mediascope Europe 2010.** European Interactive Advertising Association (EIAA) (Braybrooke Northants, United Kingdom). http://advertising.microsoft.com/schweiz/WWDocs/User/de-ch/ForPublishers/EIAA_Mediascope%202010_Schweiz.pdf

Ferber, Reginald (2003): **Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web**. dpunkt Verlag (Heidelberg, Deutschland). 352 S.

Fetterly, Dennis, Mark Manasse, Marc Najork & Janet Wiener (2003): **A Large-Scale Study of the Evolution of Web Pages**. *12th International World Wide Web Conference*, Budapest, Hungary. http://research.microsoft.com/pubs/73808/p97-fetterly.pdf

Fielding, Roy T. (1994): **Maintaining Distributed Hypertext Infostructures: Welcome to MOMspider's Web**. Computer Networks and ISDN Systems, 27(2): 193-204. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.9090

Fielding, Roy T. (2000): **Architectural Styles and the Design of Network-based Software Architectures**. *Doctor of Philosophy*. Information and Computer Science, University of California (Irvine, USA). http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Fielding, Roy T., J. Gettys, J. Mogul, H. Frystyk, et al. (1999, Juni):
RFC 2616: Hypertext Transfer Protocol - HTTP/1.1.
World Wide Web Consortium / Network Working Group.
http://www.w3.org/Protocols/rfc2616/rfc2616.html

Finck, Matthias, Michael Janneck, Monique Janneck & Hartmut Obendorf (2005): **Kooperative Wissensnetze**. *Mensch & Computer*, Linz, Österreich. Oldenbourg Wissenschaftsverlag: 133-142.
http://mc.informatik.uni-hamburg.de/konferenzbaende/mc2005/konferenzband/muc2005_12_finck_etal.pdf

Fisher, Darin (2002): **Link Prefetching FAQ.** The Mozilla Organization.
http://www.mozilla.org/projects/netlib/Link_Prefetching_FAQ.html

Fletcher, Dan (2010, 20. Mai): **Friends without borders: Facebook and how it's redefining privacy**. *TIME*, 175(21): 18-24.

Fogg, BJ & Jakob Nielsen (2000): **Probabilistic Web Link Viability Marker And Web Page Ratings**. *Patent Number: 6,163,778* (United States).

Ford, Nigel & Sherry Y. Chen (2000): **Individual Differences, Hypermedia Navigation, and Learning: An Empirical Study**. Journal of Educational Multimedia and Hypermedia, 9(4): 281-311.
http://www.editlib.org/index.cfm?fuseaction=Reader.ViewAbstract&paper_id=9546

Frécon, Emmanuel & Gareth Smith (1998): **WebPath - A Three Dimensional Web History**. *IEEE Symposium on Information Visualization (InfoVis '98)*, NC, USA. IEEE: 3-10.

Freed, Ned & Nathaniel S. Borenstein (2006): **RFC 2045 und RFC 2046: MIME Media Types.**
IANA - Internet Assigned Numbers Authority (Marina del Rey, USA).
http://www.iana.org/assignments/media-types/

Freedman, Alan (2001): **Computer Desktop Encyclopedia**.
Osborne/McGraw-Hill (New York, USA). 1124 S.

Frei, Hans-Peter & Daniel Stieger (1992): **Making use of hypertext links when retrieving information**. *ECHT*, Milan, Italy. ACM Press: 102 - 111.
http://dl.acm.org/citation.cfm?doid=168466.168502

Gamma, Erich, Richard Helm, Ralph Johnson & John Vlissides (1994):
Design Patterns: Elements of Reusable Object-Oriented Software.
Addison Wesley Longman (Reading, Massachussetts). 395 S.

Garrett, Jesse James (2005, 18. Februar): **Ajax: A New Approach to Web Applications.**
Adaptive Path LLC (San Francisco).
http://www.adaptivepath.com/publications/essays/archives/000385.php

Garrett, L. Nancy, Karen L. Smith & Norman Meyrowitz (1986): **Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System**. *Conference on Computer-Supported Cooperative Work (CSCW'86)*, Austin, Texas, US. ACM Press: 163-174.

Gedikli, Fatih, Mouzhi Ge & Dietmar Jannach (2011):
Explaining Online Recommendations Using Personalized Tag Clouds.
i-com - Zeitschrift für interaktive und kooperative Medien, 10(1/2011): 3-10.

Gerhards, Maria & Annette Mende (2003, August):
Offliner 2003: Stabile Vorbehalte gegenüber dem Internet. *Media-Perspektiven*: 359-373.
http://www.ard-zdf-onlinestudie.de/fileadmin/Online03/Online03_Offline.pdf

Gerhardt-Powals, Jill (1996):
Cognitive Engineering Principles for Enhancing Human-Computer Performance.
International Journal of Human-Computer Interaction, 8(2): 189-211.

Gertz, Michael, M. Tamer Özsu, Gunter Saake & Kai-Uwe Sattler (2004):
Report on the Dagstuhl Seminar: "Data Quality on the Web".
ACM SIGMOD Record, 33(1): 127-132.
http://dl.acm.org/citation.cfm?doid=974121.974144

Gibson, David, Jon M. Kleinberg & Prabhakar Raghavan (1998):
Inferring Web Communities from Link Topology.
9th ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA. ACM Press: 225-234.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.1187

Gillies, James & Robert Cailliau (2001): **Die Wiege des Web. Die spannende Geschichte des WWW**.
Dpunkt Verlag (Heidelberg, Deutschland). 410 S.

Glassman, Steven (1994): **A Caching Relay for the World Wide Web**.
Computer Networks and ISDN Systems, 27(2): 165-173.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.9707

Gloor, Peter A. (1990): **Hypermedia-Anwendungsentwicklung: Eine Einführung mit HyperCard-Beispielen**. B.G. Teubner (Stuttgart, Deutschland).

Goasduff, Laurence & Christy Pettey (2010): **Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down.**
Gartner, Inc. (Egham, UK).
http://www.gartner.com/it/page.jsp?id=1421013

Goldfarb, Charles F. (1997): **ISO/IEC 10744:1997: Information technology - Hypermedia/Time-based Structuring Language (HyTime)**. International Organization for Standardization.
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29303

Goldfarb, Charles F., Steven R. Newcomb, W. Eliot Kimber & Peter J. Newcomb (1997):
A Reader's Guide to the HyTime Standard. HyTime Users' Group.
http://www.hytime.org/papers/htguide.html

Goldfarb, Charles F. & Yuri Rubinsky (1990): **The SGML Handbook (Including ISO 8879 Standard)**.
Oxford University Press (Oxford). 609 S.

Golding, Richard Andrew (1992): **Weak-Consistency Group Communication and Membership**.
Dissertation. Computer & Information Sciences, University of California (Santa Cruz, USA): 165 S.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.195

Goldman-Segall, Ricki, David Jonesson & Hermann Maurer (1996): **DynamIcons as Dynamic Graphic Interfaces:Interpreting the Meaning of a Visual Representation**.
Intelligent Tutoring Media, 6(3-4): 149-158.
http://www.tandfonline.com/doi/abs/10.1080/14626269609408872?journalCode=ndcr19

Goodman, Danny (1987): **The Complete HyperCard Handbook**. Bantam Books (Toronto). 720 S.

Google (2006): **Web Authoring Statistics.** Google Inc. (Mountain View, CA, USA).
http://code.google.com/webstats/

Goose, Stuart, Wendy Hall & Siegfried Reich (2000):
Microcosm TNG: A Framework for Distributed Open Hypermedia.
IEEE MultiMedia, 7(1): 52-60.

Gorfein, David S. & Robert R. Hoffmann (1987):
Memory and Learning - The Ebbinghaus Centennial Conference.
Erlbaum Associates (New York, USA).

Graff, Martin G. (2005): **Individual Differences in Hypertext Browsing Strategies**. Behaviour & Information Technology, 24(2): 93-99. http://www.elsinnet.org.uk/research/mgg_files/BIT.pdf

Gräther, Wolfgang & Wolfgang Prinz (2001): **The Social Web Cockpit: Support for Virtual Communities**. *SIGGroup Conference on Supporting Group Work*, Boulder, Colorado, USA. ACM Press: 252-259. http://dl.acm.org/citation.cfm?doid=500286.500323

Gray, Matthew (1996, 20. Juni): **Internet Statistics - Growth and Usage of the Web and the Internet.** Massachusetts Institute of Technology (Cambridge, MA, USA). http://www.mit.edu/people/mkgray/net/

Greenberg, Saul & Andy Cockburn (1999): **Getting Back to Back: Alternate Behaviors for a Web Browser's Back Button**. *5th Annual Human Factors and the Web Conference*, NIST; Gaithersburg, USA. http://www.cosc.canterbury.ac.nz/~andy/papers/hfweb.pdf

Griffel, Frank (1998): **Componentware**. dpunkt Verlag (Heidelberg, Deutschland). 645 S.

Grønbæk, Kaj, Lennert Sloth & Niels Olof Bouvin (2000): **Open Hypermedia as User Controlled Meta Data for the Web**. *Ninth International World Wide Web Conference WWW9*, Amsterdam, The Netherlands. Elsevier: 553-566.

Grønbæk, Kaj, Lennert Sloth & Peter Ørbæk (1999): **Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW**. *8th World Wide Web Conference*, Toronto, Canada. Elsevier: 253-267. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.8773

Grønbæk, Kaj & Randall H. Trigg (1994, Februar): **Design Issues for a Dexter-Based Hypermedia System**. *Communications of the ACM*, 37(2): 40-49. http://dl.acm.org/citation.cfm?doid=175235.175238

Grønbæk, Kaj & Randall H. Trigg (1999): **From Web to Workplace**. MIT Press (Massachusetts, USA).

GSA (2008): **GSM/3G Network Update.** GSA: Global mobile Suppliers Association. http://www.gsacom.com/gsm_3g/market_update.php4

Gudgin, Martin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, et al. (2007): **SOAP Version 1.2.** World Wide Web Consortium. http://www.w3.org/TR/soap/

Guy, Marieke & Emma Tonkin (2006): **Folksonomies - Tidying up Tags?** D-Lib Magazine, 12(1). http://www.dlib.org/dlib/january06/guy/01guy.html

Haake, Jörg M., Jörg Hannemann & Manfred Thüring (1991): **Ein Ansatz zur Organisation von Hyperdokumenten**. *Hypertext & Hypermedia '91*, Graz, Austria. Springer-Verlag: 117-134.

Haan, Bernard J., Paul Kahn, Victor A. Riley, James H. Coombs & Norman K. Meyrowitz (1992, Januar): **IRIS Hypermedia Services**. *Communications of the ACM*, 35(1): 36-51.

Haas, Stephanie W. & Erika S. Grams (1998a): **A Link Taxonomy for Web Pages**. *61st Annual Meeting of the Information Access in the Global Information Economy (ASIS'98)*, Orlando, Florida, USA: 485-495.

Haas, Stephanie W. & Erika S. Grams (1998b): **Page and Link Classifications: Connecting Diverse Resources**. *3rd ACM International Conference on Digital Libraries*, Pittsburgh, USA. ACM Press: 99-107.

Hackett, Stephanie, Bambang Parmanto & Xiaoming Zeng (2004): **Accessibility of Internet Websites through Time**. *6th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS'04*, Atlanta, Georgia, USA. ACM Press: 32-39. http://dl.acm.org/citation.cfm?doid=1028630.1028638

Halasz, Frank G. (1988, Juli): **Reflections on Notecards: Seven Issues for the Next Generation of Hypertext Systems**. *Communications of the ACM*, 31(7): 836-852.

Halasz, Frank G., Thomas P. Moran & Randall H. Trigg (1987): **Notecards in a Nutshell**. *Conference on Human Factors and Computing Systems*, Toronto, Canada. ACM Press. http://dl.acm.org/citation.cfm?doid=29933.30859

Halasz, Frank G. & Mayer D. Schwartz (1994, Februar): **The Dexter Hypertext Reference Model**. *Communications of the ACM*, 37(2): 30-39.

Hall, Wendy (1997): **The History of the Microcosm Project.** University of Southampton, UK. http://www.mmrg.ecs.soton.ac.uk/projects/microcosm.html, (seit Aug. 2008 nur noch im Internet Archive verfügbar: http://web.archive.org/web/20080804131650/http://www.mmrg.ecs.soton.ac.uk/projects/microcosm.html).

Halverson, Tim & Anthony J. Hornof (2004): **Link Colors Guide a Search**. *Human Factors In Computing Systems (CHI'2004)*, Wien, Österreich. ACM Press: 1367-1370. http://dl.acm.org/citation.cfm?doid=985921.986066

Hamilton, Martin, Alex Rousskov & Duane Wessels (1998, Dezember): **SQUID Cache Digest Specification.** http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt

Hammond, Nick & Lesley Allinson (1989): **Extending Hypertext for Learning: An Investigation of Access and Guidance Tools**. *HCI'89: People and Computers V*, Nottingham, GB. Cambridge University Press: 293-304.

Hammwöhner, Rainer (1993): **Kognitive Plausibilität: Vom Netz im (Hyper-)Text zum Netz im Kopf.** Nachrichten für Dokumentation: Zeitschrift für Informationswissenschaft und –praxis, 44(1): 23-28. http://www-nw.uni-regensburg.de/~har16557/Literatur/nfd_1993.pdf

Hannemann, Jorg, Manfred Thüring & Norbert Friedrich (1992): **Hyperdocuments as User Interfaces: Exploring and Browsing Semantic for Coherent Hyperdocuments**. *Hypertext und Hypermedia 1992*. Springer: 87-102.

Hansen, Klaus Marius, Christian Yndigegn & Kaj Grønbæk (1999): **Dynamic Use of Digital Library Material Supporting Users with Typed Links in Open Hypermedia**. *European Conference on Digital Libraries*, Paris, France. Springer: 254-273. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.3650

Hardman, Lynda (1988): **Hypertext Tips: Experiences in Developing a Hypertext Tutorial**. *British Computer Society Human-Computer Interaction Specialist Group*, Manchester, UK. Cambridge University Press: 437 - 451.

Hardman, Lynda (1998): **Modelling and Authoring Hypermedia Documents**. *Ph.D. Thesis*, University of Amsterdam (Amsterdam, The Netherlands): 248 S. http://homepages.cwi.nl/~lynda/thesis/

Hardman, Lynda, Dick C. A. Bulterman & Guido van Rossum (1994, Februar): **The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model**. *Communications of the ACM*, 37(2): 50-62.

Harold, Elliotte Rusty & W. Scott Means (2002): **XML in a Nutshell**. O'Reilly & Associates (Sebastopol, Kalifornien, USA). 640 S.

Harper, Simon, Carole Goble, Robert Stevens & Yeliz Yesilada (2004):
Middleware to Expand Context and Preview in Hypertext.
ACM SIGACCESS Conference on Assistive Technologies (ASSETS), Atlanta, USA. ACM Press: 63-70.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.357

Harper, Simon, Yeliz Yesilada, Carole Goble & Robert Stevens (2004): **How Much is Too Much in a Hypertext Link? Investigating Context and Preview - A Formative Evaluation**.
Conference on Hypertext and Hypermedia, Santa Cruz, Kalifornien, USA. ACM Press: 116-125.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.3470

Harrison, Beverly L., Gordon Kurtenbach & Kim J. Vicente (1995):
An Experimental Evaluation of Transparent User Interface Tools and Information Content.
User Interface Software and Technologies (UIST'95), Pittsburgh, PA, USA. ACM Press: 81-90.
http://dl.acm.org/citation.cfm?doid=238386.238583

Harrison, Beverly L. & Kim J. Vicente (1996): **An Experimental Evaluation of Transparent Menu Usage**. *Human Factors in Computing Systems (CHI'96)*, New York, NY, USA. ACM Press: 391-398.

Hartson, H. Rex, José C. Castillo, John Kelso & Wayne C. Neale (1996):
Remote Evaluation: The Network as an Extension of the Usability Laboratory. *Conference on Human Factors in Computing Systems (CHI'96)*, Vancouver, Canada. ACM Press: 228-235.
http://dl.acm.org/citation.cfm?doid=238386.238511

Haß, Torsten (2004): **Konzeption und Implementation eines Werkzeuges zur Unterstützung partizipativer Benutzbarkeitstests von Websites**.
Diplomarbeit. Fachbereich Informatik, Universität Hamburg: 131 S.
http://vsis-www.informatik.uni-hamburg.de/publications/viewThesis.php/315

Hayden, Michael V. (2006): **National Information Assurance Glossary**.
Committee National Security Systems: 86 S.
http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf

Hégaret, Philippe Le, Ray Whitmer & Lauren Wood (2005): **Document Object Model (DOM).**
World Wide Web Consortium.
http://www.w3.org/DOM/

Hegner, Marcus (2003, Mai): **Methoden zur Evaluation von Software**.
InformationsZentrum Sozialwissenschaften (Bonn). 98 S.

Heinecke, Andreas M. (1992): **Ergonomische Gestaltung der Benutzungsschnittstellen von CAD-Systemen.** Gesellschaft für Informatik, Fachausschuss 4.2, Fachgruppe 4.2.1, Arbeitskreis AK 2.
http://www.hi-soft.de/amh/cad/

Hendler, James, Tim Berners-Lee & Eric Miller (2002): **Integrating Applications on the Semantic Web**. Journal of the Institute of Electrical Engineers of Japan, 122(10): 676-680.
http://www.w3.org/2002/07/swint

Hendley, Robert J., Nick S. Drew, Andrew M. Wood & Russell Beale (1995):
Narcissus: Visualising Information. *InfoVis: IEEE Symp. on Information Visualization*: 90-96.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.1407

Herder, Eelco (2003): **Modeling User Navigation**.
User Modeling 2003, Johnstown, USA. Springer: 417-419.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.2628

Herder, Eelco (2006): **Forward, Back and Home Again - Analyzing User Behavior on the Web**.
PhD Thesis, University of Twente (Enschede, Niederlande): 229 S.
http://www.l3s.de/~herder/thesis/index.php

Herder, Eelco, Harald Weinreich, Hartmut Obendorf & Matthias Mayer (2006): **Much to Know About History**. *Adaptive Hypermedia 2006*, Dublin, Irland. Springer: 283-287. http://vsis-www.informatik.uni-hamburg.de/publications/view.php/270

Herman, Ivon, Sandro Hawke, Eric Prud'hommeaux & Ralph Swick (2011): **W3C Semantic Web Activity.** World Wide Web Consortium. http://www.w3.org/2001/sw/

Hickson, Ian (2011): **HTML 5: A vocabulary and associated APIs for HTML and XHTML.** *W3C Working Draft*. World Wide Web Consortium. http://www.w3.org/TR/html5/

Hightower, Ron R., Laura T. Ring, Jonathan I. Helfman, Benjamin B. Bederson & James D. Hollan (1998): **Graphical Multiscale Web Histories: A Study of PadPrints**. *Hypertext'98*, Pittsburg, USA. ACM Press: 58-65. http://www.cs.umd.edu/hcil/pad++/papers/hypertext-98-padprints/hypertext-98-padprints.pdf

Hilbert, David M. & David F. Redmiles (2000): **Extracting Usability Information from User Interface Events**. ACM Computing Surveys, 32(4): 384-421. http://www.ics.uci.edu/~redmiles/publications/J006-HR00.pdf

Hill, Gary, Wendy Hall, Dave De Roure & Les Carr (1995): **Applying Open Hypertext Principles to the WWW**. *International Workshop on Hypermedia Design '95*, Montpellier, France. Springer Verlag: 174ff. http://eprints.ecs.soton.ac.uk/704/5/html/index.html

Hiltz, Starr R. & Murray Turoff (1985, Juli): **Structuring Computer-Mediated Communication Systems to Avoid Information Overload**. *Communications of the ACM*, 28(7): 680-689. http://dl.acm.org/citation.cfm?doid=3894.3895

Hofstätter, Peter Robert (1974): **Das Fischer-Lexikon: 6. Psychologie**. Fischer Taschenbuch-Verlag 552.-566. Tausend: 366 S.

Holm, Ronnie (2002, May): **A Trip Down Hypermedia Lane**. *Linux Gazette*(78). http://linuxgazette.net/issue78/holm.html

Hong, Jason I., Jeffrey Heer, Sarah Waterson & James A. Landay (2001): **WebQuilt: A Proxy-based Approach to Remote Web Usability Testing**. ACM Transactions on Information Systems (TOIS), 19(3): 263-285. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.755

Höök, Kristina , Marie Sjölinder & Nils Dahlbäck (1996): **Individual Differences and Navigation in Hypermedia**. Swedish Institute of Computer Science.

Horton, Wiliam (1994): **Das Icon Buch: Entwurf und Gestaltung visueller Symbole und Zeichen.** Addison Wesley.

Horton, William (1990): **Designing and Writing Online Documentation: Help Files to Hypertext**. John Wiley & Sons (New York).

Hotchkiss, Gord, Marina Garrison & Steve Jensen (2004, April): **Search Engine Usage in North America**. Enquiro Search Solutions. 59 S.

Huang, Jeff & Ryen W. White (2010): **Parallel Browsing Behavior on the Web**. *21st ACM Conference on Hypertext and Hypermedia*, Toronto, Canada. ACM Press: 13-17. http://dl.acm.org/citation.cfm?doid=1810617.1810622

Hussein, Tim & Jürgen Ziegler (2011): **Situationsgerechtes Recommending**. Informatik-Spektrum, 34(2/2011): 143-152.

IANA (2004): **Internet Official Names for Character Sets.**
IANA - Internet Assigned Numbers Authority (Marina del Rey, USA).
http://www.iana.org/assignments/character-sets

Ihde, Steve C., Paul P. Maglio, Jörg Meyer & Rob Barrett (2001):
Intermediary-based Transcoding Framework. IBM Systems Journal, 40(1): 179-192.
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5386962

Ingham, David B., Steve J. Caughey & Mark C. Little (1996): **Fixing the "Broken-link" Problem: The W3Objects Approach**. Computer Networks and ISDN Systems, 28(7-11): 1255-1268.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.6639

Irler, Wolfgang J. & Gilberto Barbieri (1991): **Farbmarkierungen im Hypertext als Orientierungs- und Lernhilfe**. *Hypertext/Hypermedia'91*, Graz, Österreich. Springer: 135-144.

ISO8613-11 (1995): **ISO/IEC 8613-11:1995 - Information technology - Open Document Architecture (ODA) and interchange format: Tabular structures and tabular layout.**
ISO - International Organization for Standardization.
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=23323

ISO9186 (2001): **ISO 9186: Graphical Symbols - Test Methods for Judged Comprehensibility and for Comprehension - Test Methods - Part 1: Methods for Testing Comprehensibility**.
ISO - International Organization for Standardization (Genf, Schweiz). 56 S.

ISO9241-8 (1998): **ISO 9241-8: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 8: Anforderungen an Farbdarstellungen**.
Beuth-Verlag (Berlin). DIN-Taschenbuch 354 Software-Ergonomie.

ISO9241-11 (1999): **DIN EN ISO 9241-11:1998: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 11: Anforderungen an die Gebrauchstauglichkeit; Leitsätze**.
Beuth-Verlag (Berlin). 28 S.

ISO9241-14 (1997): **ISO 9241-14:1997: Ergonomics of human-system interaction - Part 14: Menu dialogues**. ISO - International Organization for Standardization (Genf, Schweiz).

ISO9241-110 (2006): **ISO 9241-110:2006: Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung**. Beuth-Verlag (Berlin). 28 S.

ISO9241-151 (2008): **ISO 9241-151:2008: Ergonomics of human-system interaction - Part 151: Guidance on World Wide Web user interfaces**.
ISO - International Organization for Standardization (Genf, Schweiz).

Ivory, Melody Y. & Marti A. Hearst (2001):
The State of the Art in Automating Usability Evaluation of User Interfaces.
ACM Computing Surveys, 33(4): 470-516.

Ivory, Melody Y. & Rodrick Megraw (2005): **Evolution of Web Site Design Patterns**.
ACM Transactions on Information Systems (TOIS), 23(4): Oktober 2005.
http://dl.acm.org/citation.cfm?doid=1095872.1095876

Ivory, Melody Y., Rashmi R. Sinha & Marti A. Hearst (2001):
Empirically Validated Web Page Design Metrics.
Human Factors and Computing Systems (CHI'01), Seattle, WA, USA. ACM Press: 52-60.
http://dl.acm.org/citation.cfm?doid=365024.365035

Jackson, Collin, Daniel R. Simon, Desney S. Tan & Adam Barth (2007):
An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks.
1st International Conference on Usable Security (USEC'07), Lowlands, Scarborough, Trinidad and Tobago. Springer Verlag: 281-293.
http://dx.doi.org/10.1007/978-3-540-77366-5_27

Jander, Mary (2002): **Internet Growth Slows, Report Says.**
Light Reading Newsletter from United Business Media LLC (New York, USA).
http://www.lightreading.com/document.asp?doc_id=22798

Jansen, Bernard J. & Amanda Spink (2006): **How are we Searching the World Wide Web? A Comparison of Nine Search Engine Transaction Logs**.
Information Processing and Management, 42(2006): 248-263.
http://ist.psu.edu/faculty_pages/jjansen/academic/pubs/jansen_searching_the_web.pdf

Jeffries, Robin, James R. Miller, Cathleen Wharton & Kathy M. Uyeda (1991):
User Interface Evaluation in the Real World: A Comparison of Four Techniques.
Human factors in computing systems: Reaching through technology, New Orleans, Louisiana, USA. ACM Press: 119-124.
http://dl.acm.org/citation.cfm?doid=108844.108862

Jerroudi, Zoulfa El, Jürgen Ziegler, Stephan Meissner & Axel Philipsenburg (2005):
E-Quest: Ein Online-Befragungswerkzeug für Web Usability.
Mensch & Computer 2005: Kunst und Wissenschaft – Grenzüberschreitungen der interaktiven ART, Paderborn, Deutschland. Oldenbourg Wissenschaftsverlag.
http://mc.informatik.uni-hamburg.de/konferenzbaende/mc2005/konferenzband/muc2005_30_jerrodi_etal.pdf

John, Bonnie E. & David E. Kieras (1996):
The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast.
ACM Transactions on Computer-Human Interaction (TOCHI), 3(4): 320-351.
http://dl.acm.org/citation.cfm?doid=235833.236054

Johnson, Chris (1997a): **What is the Web Worth?**
Time and the Web, Staffordshire, Großbritannien.
http://www.hiraeth.com/conf/web97/papers/johnson.html

Johnson, Ralph E. (1997b): **Components, Frameworks, Patterns**.
SIGSOFT Symposium on Software Reusability. ACM Press: 10-17.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.6157

Joho, Hideo & Joemon M. Jose (2006):
A Comparative Study of the Effectiveness of Search Result Presentation on the Web
Lecture Notes in Computer Science: Advances in Information Retrieval, ECIR 2006 / LNCS 3936: 302-313.
http://www.springerlink.com/content/t54821704181874t/

Jonassen, David H. (1993): **Effects of Semantically Structured Hypertext Knowledge Bases on Users' Knowledge Structures**, in C. McKnight, A. Dillon & J. Richardson (ed.): "*Hypertext - A Psychological Perspective*". Ellis Horwood Limited (Chichester, West Sussex): 153-166.

Jones, William, Harry Bruce & Susan Dumais (2001):
Keeping Found Things Found on the Web.
Conference on Information Knowledge Management, Atlanta, USA. ACM Press: 119-126.
http://kftf.ischool.washington.edu/docs/KFTF_Web.pdf

Jones, William, Susan Dumais & Harry Bruce (2002): **Once Found, What Then?: A Study of "Keeping" Behaviors in Personal Use of Web Information**.
ASIST 2002 : 65th ASIST Annual Meeting, Philadelphia, USA. ASIST: 391-402.
http://kftf.ischool.washington.edu/docs/ASIST2002.pdf

Jordan, David & Craig Russell (2005): **Java Data Objects**.
O'Reilly (Sebastopol, CA, USA). 384 S.

Jørgensen, Anker Helms (1989): **Using the Thinking-Aloud Method in System Development**.
Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Boston, MA, USA. Elsevier Science Publishers: 743-750.

Joyce, Rosemary A., Carolyn Guyer & Michael Joyce (2000): **Sister Stories.**
NYU Press (New York).
http://www.nyupress.org/sisterstories/index.html,
(seit Juli 2008 nur noch im Internet Archive verfügbar:
http://web.archive.org/web/20080731193801/http://www.nyupress.org/sisterstories/index.html).

Kaasten, Scaun & Saul Greenberg (2001):
Integrating Back, History and Bookmarks in Web Browsers.
Conference of Human Factors in Computing Systems (CHI'01), Seattle, USA. ACM Press: 379-380.
http://grouplab.cpsc.ucalgary.ca/papers/

Kaasten, Shaun, Saul Greenberg & Christopher Edwards (2001):
How People Recognize Previously Seen Web Pages from Titles, URLs and Thumbnails.
Department of Computer Science, University of Calgary (Calgary, Alberta, Canada). 8 S.

Kacmar, Charles J. & Jane M. Carey (1991):
Assessing the Usability of Icons in User Interfaces.
Behaviour and Information Technology, 10(6): 443-457.

Kaczmirek, Lars (2003): **Gebrauchstauglichkeit der Ergebnisseiten von Suchmaschinen**.
Mensch & Computer 2003: Interaktion in Bewegung, Stuttgart. B.G. Teuber Verlag: 337-347.
http://mc.informatik.uni-hamburg.de/konferenzbaende/mc2003/konferenzband/muc2003-33-kaczmirek.pdf

Kahan, José & Marja-Riitta Koivunen (2001): **Annotea: An Open RDF Infrastructure for Shared Web Annotations**. *World Wide Web Conference*, Hong Kong. ACM Press: 623-632.
http://www.www10.org/cdrom/papers/pdf/p488.pdf

Kahn, Paul (1995, August): **Visual Cues for Local and Global Coherence in the WWW**.
Communications of the ACM, 38(8): 67-69.

Kahn, Paul, Ronnie Peters & George P. Landow (1995):
Three Fundamental Elements of Visual Rhetoric in Hypertext,
in Wolfgang Schuler, Jörg Hannemann & Norbert Streitz (ed.): "*ESPRIT Research Report, Project 6532*". Springer (Berlin).
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.327

Kalbach, James & Tim Bosenick (2003): **Web Page Layout: A Comparison Between Left- and Right-justified Site Navigation Menus**. JoDI: Journal of Digital Information, 4(1): Article 153.
http://journals.tdl.org/jodi/article/viewArticle/94

Kamba, Tomonari, Krishna Bharat & Michael C. Albers (1995):
The Krakatoa Chronicle - An Interactive, Personalized, Newspaper on the Web.
4th World Wide Web Conference, Boston, USA. Elsevier: 159-170.
http://www.w3.org/Conferences/WWW4/Papers/93/

Kaminski, Chris (2001, 22. Juni): **Much Ado About Smart Tags**.
A List Apart Magazine(115).
http://www.alistapart.com/articles/smarttags/

Kangas, Steve (2001): **Layout and Content of Popular Sites.**
Atlas NetConversions (Seattle, WA, USA).
http://www.atlassolutions.com/pdf/Layout_popular_sites_new.pdf

Kantor, Brian & Phil Lapsley (1986): **RFC 977 - NNTP: Network News Transfer Protocol.**
Network Working Group.
http://www.ietf.org/rfc/rfc977.txt

Kappe, Frank (1991): **Spezielle Eigenschaften großer Hypermedia-Systeme**.
Hypertext & Hypermedia'91, Graz, Austria. Springer: 164-173.

Kappe, Frank (1995): **A Scalable Architecture for Maintaining Referential Integrity in Distributed Information Systems**. J.UCS: Journal of Universal Computer Science, 1(2): 84-104.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.4547

Karat, Clare-Marie, Robert Campbell & Tarra Fiegel (1992):
Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation.
CHI '92, New York, NY, USA. ACM Press: 397-404.

Katerattanakul, Pairin & Keng Siau (1999):
Measuring Information Quality of Web Sites: Development of an Instrument.
20th Int. Conference on Information Systems, Charlotte, North Carolina, USA. ACM Press: 279 - 285.

Kay, Russell (2004): **Phishing**.
Computerworld (Magazine), Online Quick Study.
http://www.computerworld.com/securitytopics/security/story/0,10801,89096,00.html.

KDE (2011): **KDE Human Interface Guidelines (HIG).**
KDE Usability Project, KDE e.V. (Tübingen, Deutschland).
http://techbase.kde.org/Projects/Usability/HIG

Keep, Christopher, Tim McLaughlin & Robin Parmar (1993): **The Electronic Labyrinth.**
http://eserver.org/elab/

Kehoe, Colleen, Jim Pitkow, Kate Sutton, Gaurav Aggarwal & Juan D. Rogers (1999):
10th WWW User Survey.
GVU: Graphic, Visualization and Usability Center, College of Computing, Georgia Institute of Technology (Atlanta, USA).
http://www.gvu.gatech.edu/user_surveys/survey-1998-10/report.pdf

Kellar, Melanie, Carolyn Watters & Kori M. Inkpen (2007):
An Exploration of Web-based Monitoring: Implications for Design.
CHI '07 - SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, USA. ACM Press: 377-386.
http://dl.acm.org/citation.cfm?doid=1240624.1240686

Kibby, M. R. & J. T. Mayes (1989): **Towards intelligent hypertext**.
Hypertext Theory into Practice, Norwood, USA. Ablex: 164-172.

Kıcıman, Emre & Benjamin Livshits (2007): **AjaxScope: A Platform for Remotely Monitoring the Client-Side Behavior of Web 2.0 Applications**.
ACM SIGOPS Operating Systems Review, 41(6): 17-30.
http://dl.acm.org/citation.cfm?doid=1323293.1294264

Kidd, Alison (1994): **The Marks are on the Knowledge Worker**.
Conference on Human Factors in Computing Systems (CHI), Boston, USA. ACM Press: 186-191.
http://dl.acm.org/citation.cfm?doid=191666.191740

Kieras, David (1994): **GOMS modeling of user interfaces using NGOMSL**.
Conference on Human Factors in Computing Systems (CHI'94), Boston, Massachusetts, USA.
ACM Press: 371-372.
http://dl.acm.org/citation.cfm?doid=259963.260467

Kim, Kyung-Sun (2006): **Experienced Web Users' Search Behavior: Effects of Focus and Emotion Control**. Proceedings of the American Society for Information Science and Technology, 42(1).
http://arizona.openrepository.com/arizona/handle/10150/105638

King, Andrew (2011): **Average Web Page Size Septuples Since 2003.** Website Optimization, LLC (Ann Arbor, MI, USA). http://www.websiteoptimization.com/speed/tweak/average-web-page/

King, Andrew B. (2003): **Speed Up Your Site: Web Site Optimization**. New Riders: 528 S.

King, Andrew B. (2006): **The Average Web Page.** Website Optimization, LLC (Ann Arbor, MI, USA).
http://www.optimizationweek.com/reviews/average-web-page/

Kleinberg, Jon M. (1999): **Authoritative Sources in a Hyperlinked Environment**.
Journal of the ACM, 46(5): 604-632.
http://www.cs.cornell.edu/home/kleinber/auth.pdf

Koehler, Wallace (1999): **An Analysis of Web Page and Web Site Constancy and Permanence**.
Journal of the American Society for Information Science, 50(2): 162-180.

Kohut, Andrew, Richard Wike & Juliana Menasce Horowith (2007): **World Publics Welcome Global Trade - But Not Immigration.** Pew Research Center (Washington, DC, USA).
http://pewglobal.org/reports/display.php?ReportID=258

König, Wolfgang (1994): **Profil der Wirtschaftsinformatik. Ausführungen der Wissenschaftlichen Kommission der Wirtschaftsinformatik.** *Wirtschaftsinformatik*, 36(1): 80-81.

Kopak, Richard W. (1999): **Functional Link Typing in Hypertext**.
ACM Computing Surveys, 31(4es): 16.
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/41.html

Kopak, Richard William (2000): **A Taxonomy of Link Types for Use in Hypertext**.
Doctor of Philosophy. Faculty of Information Studies, University of Toronto (Canada): 276 S.
http://hdl.handle.net/1807/14406

Kopetzky, Theodorich & Max Mülhäuser (1999):
Visual Preview for Link Traversal on the World Wide Web.
Journal of Computer Networks and ISDN Systems
(Proc. of the 8th International WWW Conference, Toronto), 31(11-16): 1525-1532.

Korsgaard, Thomas Rune (2007): **Improving Trust in the Wikipedia**. *Master's Thesis*. Informatics and Mathematical Modelling, Technical University of Denmark (Lyngby, Denmark): 217 S.
http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=5397

Kossel, Axel & Urs Mansmann (2011, 11. April): **Breitband-Zensus**. *c't Magazin für Computertechnik*, 2011(9): 96-100.

Koster, Martijn (1993): **Guidelines for Robot Writers.** RobotsTXT.org (Wymondham, UK).
http://www.robotstxt.org/wc/guidelines.html

Koster, Martijn (1994a): **Aliweb - Archie-Like Indexing in the Web**.
1st Int. Conference on the World-Wide Web, CERN, Genf, Schweiz. Elsevier Science BV: 175 - 182.
http://www.iicm.tugraz.at/thesis/cguetl_diss/literatur/Kapitel05/References/Koster_1994/aliweb.pdf

Koster, Martijn (1994b): **A Standard for Robot Exclusion.** RobotsTXT.org (Wymondham, UK).
http://www.robotstxt.org/orig.html

Koved, Larry & Ben Shneiderman (1986): **Embedded menus: selecting items in context.**
Communications of the ACM, 29(4): 312-318.

Krcmar, Helmut (2004): **Informationsmanagement**. Springer Verlag (Berlin). 574 S.

Kreutz, Reinhard, Helmar Conradi & Klaus Spitzer (1998):
Improved Visual Navigation in Web-Documents.
AACE ED-MEDIA / ED-TELECOM 98, Freiburg, Germany.
Association for the Advancement of Computing in Education (AACE): 755-760.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.1599

Kreutz, Reinhard, Brigitte Euler & Klaus Spitzer (1999): **No Longer Lost in WWW-based Hyperspaces**. *ACM Hypertext '99 Conference*, Darmstadt, Germany. ACM Press: 133-134.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.3795

Kristol, David M. & Lou Montulli (2000, Oktober): **RFC 2965: HTTP State Management Mechanism.** Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2965.txt

Kuhlen, Rainer (1991): **Hypertext : Ein nicht-lineares Medium zwischen Buch und Wissensbank**. Springer Verlag (Berlin). 362 S.

Kuhlen, Rainer (1997): **Hypertext**, in Marianne Buder, Werner Rehfeld, Thomas Seeger & Dietmar Strauch (ed.): "*Grundlagen der praktischen Information und Dokumentation*". Saur (München): 1069.

Kupiec, Julian, Jan Pedersen & Francine Chen (1995): **A trainable document summarizer**.
SIGIR Conference on Research and Development in Information Retrieval. ACM Press: 68-73.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1161

Ladd, Brian C., Michael V. Capps, P. David Stotts & Rick Furuta (1994):
Multi-Head Multi-Tail Mosaic: Adding Parallel Automata Semantics to the Web.
4th Annual World Wide Web Conference, Boston, USA. O'Reilly & Associates: 433-440.
http://www.w3.org/Conferences/WWW4/Papers/118/

Lancaster, Frederick W. & Emily G. Fayen (1973): **Information Retrieval On-Line**.
John Wiley & Sons Inc: 597 S.

Landow, George P. (1987): **Relationally Encoded Links and the Rhetoric of Hypertext**.
Hypertext '87, Chapel Hill, NC. ACM Press: 331-344.

Landow, George P. (1997): **Hypertext 2.0 - The Convergence of Contemporary Critical Theory and Technology**. Johns Hopkins University Press (Baltimore).

Landow, George P. (2004): **Hypertext.** National University of Singapore.
http://www.cyberartsweb.org/cpace/ht/htov.html

Landow, George P. & Paul Kahn (1992): **Where's the Hypertext? The Dickens Web as a System-Independent Hypertext**. *Hypertext '92*, Milan, Italy. ACM Press: 149-160.
http://dl.acm.org/citation.cfm?doid=168466.168515

Landsberger, Henry A. (1958): **Hawthorne Revisited: Management and the Worker, Its Critics, and Developments in Human Relations in Industry**.
Cornell University Press (Ithaca, New York). 119 S.

Lang, Bernard & Francois Rouaix (1996a): **The V6 Web Engine - Release 1**
Technical Report. INRIA (Paris Rocquencourt).
ftp://ftp.inria.fr/INRIA/Projects/cristal/Francois.Rouaix/V6/v6.ps.gz

Lang, Bernard & François Rouaix (1996b): **The V6 Engine.**
World Wide Web Conference Workshop Paper (Paris, Frankreich).
http://www.cs.vu.nl/~eliens/WWW5/papers/V6/

Lang, Darice M. & Monica Luketick (1996): **Mining for Gems in an Information Overload**.
14th Conference on Systems Documentation, North Carolina, USA. ACM Press: 167-178.
http://dl.acm.org/citation.cfm?doid=238215.238270

Langer, Stefan (2001): **Natural languages on the Word Wide Web**.
B.U.L.A.G.: N°26, T.A.L. et Internet. Presses Universitaires Franc-Comtoises: 89-100.
http://www.cis.uni-muenchen.de/people/langer/veroeffentlichungen/bulag.pdf

Lansdale, M.W. (1988): **The psychology of personal information management**.
Applied Ergonomics, 19(1): 55-66.

Lawson, Mark (2003): **Web's inventor gets a knighthood.** BBC World News (London).
http://news.bbc.co.uk/2/hi/technology/3357073.stm

Lazar, Jonathan, Alfreda Dudley-Sponaugle & Kisha-Dawn Greenidge (2004):
Improving web accessibility: a study of webmaster perceptions.
Computers in Human Behavior, 20(2): 269-288.
http://www.sciencedirect.com/science/article/pii/S0747563203000906

Lee, SK, William Buxton & Kenneth C. Smith (1985):
A multi-touch three dimensional touch-sensitive tablet.
Human Factors in Computing Systems (CHI'85), San Francisco, California, USA. ACM Press: 21-25.
http://dl.acm.org/citation.cfm?doid=317456.317461

Lee, Wie M. & Brian Jepson (2005): **Programming the dotNET Compact Framework**.
Microsoft Press (Redmonton, Washington, USA). 304 S.

Lempel, Ronny & Shlomo Moran (2000): **The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect**. Computer Networks, 33(1-6): 387-401.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.5859

Leporda, Alex (2005): **VaiWatch 2005 - Untersuchung zum deutschsprachigen Web.**
validome.org (Braunfels). http://www.validome.org/lang/ge/html/valiwatch-web-2005

Leuf, Bo & Ward Cunningham (2001): **The Wiki Way: Collaboration and Sharing on the Web**.
Addison-Wesley (Boston). 435 S.

Levine, Rick (1996): **Guide to Web Style.** Sun Microsystems.
http://www.sun.com/styleguide/ (nicht mehr verfügbar, Kopie unter:
http://vsis-www.informatik.uni-hamburg.de/ergonomie/sun_style/Printing.Version.html).

Levy, David M. (1997): **I Read the News Today, Oh Boy: Reading and Attention in Digital Libraries**.
Digital Libraries, Philadelphia, PA. ACM Press: 202-211.

Lewandowski, Dirk & Nadine Höchstötter (2007): **Qualitätsmessung bei Suchmaschinen - System- und nutzerbezogene Evaluationsmaße**. Informatik-Spektrum, 30(3): 159-169.
http://www.durchdenken.de/lewandowski/doc/IS_2007_Preprint.pdf

Lewandowski, Dirk, Henry Wahlig & Gunnar Meyer-Bautor (2006): **The Freshness of Web Search Engines' Databases**. Journal of Information Science, 32(2): 131-148.
http://www.durchdenken.de/lewandowski/doc/jis_preprint.pdf

Lewis, Clayton (1982, 17. Februar): **Using the "Thinking-aloud" Method in Cognitive Interface Design**. *Report*. IBM Thomas J. Watson Research Center (Yorktown Heights, NY). 6 S.

LexisNexis (2003): **How to Shepardize.** LexisNexis, Reed Elsevier Inc. http://www.lexisnexis.com/infopro/training/reference/Shepards/shepardscompgd.pdf

Lie, Håkon Wium, Tantek Çelik, Daniel Glazman & Anne van Kesteren (2010): **CSS3 Media Queries.** *Editor's Draft*. World Wide Web Consortium. http://dev.w3.org/csswg/css3-mediaqueries/

Lin, Eric, Saul Greenberg, Eileah Trotter, David Ma & John Aycock (2011): **Does Domain Highlighting Help People Identify Phishing Sites?** *CHI 2011*, Vancouver, Canada. ACM Press: 2075-2084. http://dl.acm.org/citation.cfm?doid=1978942.1979244

LinkQuality.com (2011): **The Web's Integrity Benchmark.** Linkalarm Inc. http://linkquality.com/

Liu, Shenwei & Keishi Tajima (2010): **WildThumb: A Web Browser Supporting Efficient Task Management on Wide Displays**. *14th Int. Conference on Intelligent User Interfaces - IUI '10*, Hong Kong, China. ACM Press: 159-168. http://www.dl.soc.i.kyoto-u.ac.jp/~tajima/papers/iui10abs.html

Livshits, Benjamin & Emre Kıcıman (2008): **Doloto: Code Splitting for Network-Bound Web 2.0 Applications**. *SIGSOFT 2008*, Atlanta, Georgia, USA. ACM Press. http://dl.acm.org/citation.cfm?doid=1453101.1453151

Loiacono, Eleanor & Scott McCoy (2006): **Website accessibility: a cross-sector comparison**. Universal Access in the Information Society, 4(4): 393-399. http://www.springerlink.com/content/p61110888gt0555g/

Loiacono, Eleanor T., Nicholas C. Romano Jr. & Scott McCoy (2009, September): **The State of Corporate Website Accessibility**. *Communications of the ACM*, 52(9): 128-132. http://dl.acm.org/citation.cfm?doid=1562164.1562197.

Lorenzen-Schmidt, Olde & Stephan Nufer (2008): **From a Distance: Usability Testing aus der Ferne**. i-com - Zeitschrift für interaktive und kooperative Medien, 7(1/2008): 44-46.

Louden, Kenneth C. (1994): **Programmiersprachen: Grundlagen, Konzepte, Entwurf**. International Thomson Publishing: 810 S.

Luhmann, Niklas (1992): **Kommunikation mit Zettelkästen – Ein Erfahrungsbericht**, in "*Universität als Milieu*". Haux (Bielefeld): 53-61.

Luotonen, Ari & Kevin Altis (1994): **World-Wide Web Proxies**. Computer Networks and ISDN Systems (Special Issue for the 1st WWW Conference), 27(2): 147-154. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.3972

Lyman, Peter, Hal R. Varian & et al. (2003, Oktober): **How Much Information 2003?** School of Information Management & Systems, University of California (Berkeley). http://www.sims.berkeley.edu/research/projects/how-much-info-2003/

Lyman, Peter, Hal R. Varian, James Dunn, Aleksey Strygin & Kirsten Swearingen (2000): **How Much Information 2000?** School of Information Management & Systems, University of California (Berkeley). http://www2.sims.berkeley.edu/research/projects/how-much-info/

Lynch, Patrick J. & Sarah Horton (1997, Juli): **Imprudent Linking Weaves a Tangled Web**. *IEEE Computer*, 30(7): 115-117.

Lynch, Patrick J. & Sarah Horton (2002):
Web Style Guide: Basic Design Principles for Creating Web Sites.
Yale University Press (Yale). 176 S.

Lythgoe, John N. (1979): **The Ecology of Vision**. Oxford University Press (Oxford, UK).

Macheras, Ioannis (2003): **Fortschrittliche Hypertext-Konzepte und ihre Implementierung im Web auf Basis von XML, XLink und XPointer**.
Diplomarbeit. Institut für Informatik, Technische Universität München: 116 S.

Mack, Robert L. & Jill M. Robinson (1992): **When Novices Elicit Knowledge: Question-Asking in Designing, Evaluating and Learning to use Software**, in "*The Psychology of Expertise: Cognitive Research and Empirical AI*". Springer-Verlag (New York): 245-268.

Magel, Ken (1997, December): **Is It Too Late to Put the User Back into HTML?**
Computer, 30(12): 131-132.

Maglio, Paul P. & Rob Barrett (2000): **Intermediaries Personalize Information Streams**.
Communications of the ACM, 43(8): 96-101.
http://dl.acm.org/citation.cfm?doid=345124.345158

Malandrino, Delfina & Vittorio Scarano (2005):
A Taxonomy of Programmable HTTP Proxies for Advanced Edge Services.
1st Int. Conference on Web Information Systems and Technologies, Miami, Florida, USA: 231-238.
http://isis.dia.unisa.it/papers/ProxyWEBIST05.pdf

Malandrino, Delfina & Vittorio Scarano (2006): **Tackling Web Dynamics by Programmable Proxies**.
Computer Networks, 50(10): 1564-1580.
http://isis.dia.unisa.it/papers/CNWebDynamics.pdf

Maloney, Murray & Liam Quin (1995): **Hypertext Links in HTML.**
Draft of the Internet Engineering Task Force.
http://www.w3.org/MarkUp/draft-ietf-html-relrev-00.txt

Marchionini, Gary, Gary Geisler & Ben Brunk (2000):
AgileViews: A Human-Centered Framework for Interfaces to Information Spaces.
Annual Conference of the American Society for Information Science, Chicago, USA: 271-280.
http://ils.unc.edu/~march/agileviews/Agileviews.pdf

Marchionini, Gary & Ben Shneiderman (1988):
Finding Facts vs. Browsing Knowledge in Hypertext Systems.
IEEE Computer, 21(1): 70-80.

Marcu, Daniel (1997): **From discourse structures to text summaries**.
14th National Conference on Artificial Intelligence (AAAI-97), Providence, Rhode Island.
AAAI Press / MIT Press: 629-636.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.6671

Markwell, John & David W. Brooks (2003): **"Link Rot" Limits the Usefulness of Web-based Educational Materials in Biochemistry and Molecular Biology**.
Biochemistry and Molecular Biology Education, 31(1): 69-72.
http://www.bambed.org/cgi/content/full/31/1/69

Marshall, Catherine C., Frank M. Shipman III & James H. Coombs (1994):
VIKI: Spatial Hypertext Supporting Emergent Structure.
European conference on Hypermedia technology, Edinburgh. ACM Press: 13-23.
http://dl.acm.org/citation.cfm?doid=192757.192759

Marshall, Catherine C. & Russell A. Rogers (1992):
Two Years before the Mist: Experiences with Aquanet.
ECHT '92: European Conference on Hypertext Technology, Milano, Italy. ACM Press: 53-62.

Marshall, Catherine C. & Frank M. Shipman III (1995, August): **Spatial Hypertext: Designing for Change**. *Communications of the ACM*, 38(8): 88-97.
http://dl.acm.org/citation.cfm?doid=208344.208350

Marshall, Catherine C. & Frank M. Shipman III (2003): **Which Semantic Web?**
Hypertext and Hypermedia 2003, Nottingham, UK. ACM Press: 57-66.
http://dl.acm.org/citation.cfm?doid=900051.900063

Mattsson, Michael, Jan Bosch & Mohamed E. Fayad (1999, October):
Framework Intergration - Problems, Causes, Solutions.
Communications of the ACM, 42(10): 81-87.

Mauldin, Alan (2011): **Global Bandwidth Research Service - Executive Summary.**
TeleGeography, Inc. (Washington D.C., USA).
http://www.telegeography.com/page_attachments/products/website/research-services/global-bandwidth-research-service/0002/2017/gb11-exec-sum.pdf

Maurer, Hermann (1996): **HyperWave - The Next Generation Web Solution**.
Addison Wesley: 635 S.

Mayer, Matthias (2008): **Visualizing Web Sessions: Improving Web Browser History by a Better Understanding of Web Page Revisitation and a New Session- and Task-Based, Visual Web History Approach**. *Dissertation*. Department of Informatics, University of Hamburg: 319 S.

Mayer, Matthias (2009): **Web History Tools and Revisitation Support: A Survey of Existing Approaches and Directions**.
Foundations and Trends® in Human–Computer Interaction, 2(3): 173-278.
http:/dx.doi.org/10.1561/1100000011

Mayer, Matthias & Benjamin B. Bederson (2001):
Browsing Icons: A Task-Based Approach for a Visual Web History.
HCI Lab, University of Maryland (Maryland, USA).

McCarron, Shane & Masayasu Ishikawa (2010):
XHTML™ 1.1 - Module-based XHTML - Second Edition.
W3C Recommendation, 23 November 2010. World Wide Web Consortium.
http://www.w3.org/TR/xhtml11/

McGuinness, Deborah L. & Frank van Harmelen (2004): **OWL Web Ontology Language.**
World Wide Web Consortium.
http://www.w3.org/TR/owl-features/

McIntyre, Donald (2002): **Colour Blindness: Causes and Effects**.
Dalton Publishing (Chester, Vereinigtes Königreich). 164 S.

McKendree, Jean, Will Reader & Nick Hammon (1995, July):
The "Homeopathic Fallacy" in Learning from Hypertext. *Interactions*, 2(3): 74-82.
http://dl.acm.org/citation.cfm?doid=208666.208687.

McKnight, Cliff, Andrew Dillon & John Richardson (1991): **Hypertext In Context**.
Cambridge University Press (Cambridge, UK). 166 S.

Medjahed, Brahim, Boualem Benatallah, Athman Bouguettaya, Anne H. H. Ngu & Ahmed K. Elmagarmid (2003): **Business-to-Business Interactions: Issues and Enabling Technologies**. International Journal on Very Large Data Bases, 12(1): 59.85. http://www.springerlink.com/content/659ll5tlgretcpe1/

Menezes, Alfred J., Paul C. van Oorschot & Scott A. Vanstone (1996): **Handbook of Applied Cryptography**. CRC Press (Boca Raton, Florida, USA). 816 S.

Merriam-Webster (1995): **Webster's New Encyclopedic Dictionary**. Black Dog & Leventhal Publishers Inc. (New York). 1639 S.

Merton, Robert K. & Elinor Barber (2006): **The Travels and Adventures of Serendipity: A Study in Sociological Semantics and the Sociology of Science**. Princeton University Press (Princeton, New Jersey, USA). 352 S.

Merz, Michael (2002): **E-Commerce und E-Business**. dpunkt Verlag (Heidelberg). 2. Auflage: 908 S.

Meyerovich, Leo A. & Rastislav Bodík (2010): **Fast and Parallel Webpage Layout**. *World Wide Web Conference 2010*, Raleight, North Carolina, USA. ACM Press: 711-720. http://www.eecs.berkeley.edu/~lmeyerov/projects/pbrowser/pubfiles/playout.pdf

Meyrowitz, Norman (1986): **Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework**. *OOPSLA'86*, Portland, Oregon, USA. ACM Press: 186-201.

Meyrowitz, Norman & Andries Van Dam (1982a): **Interactive Editing Systems: Part I**. ACM Computing Surveys (CSUR), 14(3): 321-352. http://dl.acm.org/citation.cfm?doid=356887.356889

Meyrowitz, Norman & Andries Van Dam (1982b): **Interactive Editing Systems: Part II**. ACM Computing Surveys, 14(3): 353-415.

Microsoft, Inc. (2011): **A History of Internet Explorer: Highlights from the first 15 years.** Microsoft, Inc. (Seattle, WA, USA). http://windows.microsoft.com/en-us/internet-explorer/products/history

Middendorf, Stefan, Reiner Singer & Jörn Heid (2002): **Programmierhandbuch und Referenz für die JavaTM-2-Plattform, Standard Edition**. DPunkt Verlag (Heidelberg). 3. Auflage: 1256 S.

Miles-Board, Timothy, Leslie A. Carr & Wendy Hall (2002): **Looking for Linking: Associative Links on the Web**. *Hypertext '02*, College Park, Maryland. ACM Press: 76-77.

Millen, David R., Jonathan Feinberg & Bernard Kerr (2006): **Dogear: Social Bookmarking in the Enterprise**. *Conf. on Human Factors in Computing Systems (CHI 2006)*, Montreal, Canada. ACM Press: 111-120. http://dl.acm.org/citation.cfm?doid=1124772.1124792

Miller, Michael & L. Jay Wantz: (1998): **COOL Links: Ride the Wave**. Computer Networks, 30(1-7): 663-665. http://www.ra.ethz.ch/CDstore/www7/1910/com1910.htm

Miller, Robert B. (1968): **Response Time in Man-Computer Conversational Transactions**. *AFIPS Fall Joint Computer Conference*. AFIPS Press: 267-277.

Miller, Robert C. & Krishna Bharat (1998):
Sphinx: A Framework for Creating Personal, Site-Specific Web Crawlers.
7th International World-Wide Web Conference, Brisbane, Australia. Elsevier: 161-172.
http://www.cs.cmu.edu/~rcm/papers/www7/

Mills, Carol Bergfeld & Linda J. Weldon (1987): **Reading Text from Computer Screens**.
ACM Computing Surveys, 19(4): 329-358.

Mobrand, Kathryn A. & Jan H. Spyridakis (2002):
A Web-Based Study of User Performance with Enhanced Local Navigational Cues.
Professional Communication Conference, IPCC 2002, Portland, Oregon, USA.
IEEE International: 500-508.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1049134

Modha, Dharmendra S. & W. Schott Spangler (2000):
Clustering Hypertext with Applications to Web Searching.
Hypertext 2000, San Antonio, Texas, USA. ACM Press: 143-152.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.996

Mohageg, Michael F. (1992):
The Influence of Hypertext Linking Structures on the Efficiency of Information Retrieval.
Human Factors, 34(3): 351-367.

Molich, Rolf & Jakob Nielsen (1990): **Improving a Human-Computer Dialogue**.
Communications of the ACM, 33(3): 338-348.
http://dl.acm.org/citation.cfm?doid=77481.77486

Morkes, John & Jakob Nielsen (1997):
Concise, SCANNABLE, and Objective: How to Write for the Web.
Sun Microsystems.
http://www.useit.com/papers/webwriting/writing.html

Morris, Dan, Meredith Ringel Morris & Gina Venolia (2008): **SearchBar: A Search-Centric Web History for Task Resumption and Information Re-finding**.
Human Factors in Computing Systems (CHI 2008), Florenz, Italien: 1207-1216.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.8556

Moulthrop, Stuart (1991): **Victory Garden.**
Eastgate Systems (Watertown, USA).
http://www.eastgate.com/VG/VGStart.html

Mukherjea, Sougata & James D. Foley (1995):
Visualizing the World-Wide Web with the Navigational View Builder.
Computer Networks and ISDN Systems, 27(6): 1075-1087.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.3747

Müller-Prove, Matthias (2001): **Vision and Reality of Hypertext and Graphical User Interfaces**.
Diplomarbeit. Fachbereich Informatik, Universität Hamburg: 122 S.
http://www.mprove.de/diplom/download.html

Müller-Prove, Matthias (2007): **Taxonomien und Folksonomien - Tagging als neues HCI-Element**.
i-com - Zeitschrift für interaktive und kooperative Medien, 6(1): 14-18.
http://www.mprove.de/script/07/icom/index.html

Müller-Wilken, Stefan (2000): **Gateway to Heaven - Schnittstellen zwischen Web und WAP**.
IX, 2000(05): 134-139.
http://www.heise.de/kiosk/archiv/ix/2000/5/134

Münz, Stefan & Clemens Gull (2010): **HTML 5 Handbuch**.
Franzis Verlag (München, Deutschland): 744 S.

Myers, Brad A. & Mary Beth Rosson (1992): **Survey on User Interface Programming**.
CHI '92 - SIGCHI Conference on Human Factors in Computing Systems, Monterey, Californien, USA. ACM Press: 195 - 202.
http://dl.acm.org/citation.cfm?doid=142750.142789

Nahrath, Michael (2000, 30. März): **The 'link'-Element in (X)HTML.**
subotnik.net - Beiträge zu HTML, CSS und Webkultur.
http://www.subotnik.net/html/link.html.en

Naimark, Michael (1997): **A 3D Moviemap and a 3D Panorama**.
Society of Photo-Optical Instrumentation Engineers (SPIE), San Jose, CA, USA: 297-305.
http://www.naimark.net/writing/spie97.html

Nake, Frieder (2002):
Data, Information, and Knowledge: A Semiotic View of Phenoma of Organization.
Organizational Semiotics: Evolving a Science of Information Systems, Montreal, Canada. Kluwer Academic Publishers: 41-50.
http://www.agis.informatik.uni-bremen.de/ARCHIV/Publikationen/DataInformKnowledge.pdf

Nanard, Jocelyne & Marc Nanard (1993):
Should Anchors Be Typed Too? An Experiment with MacWeb.
Hypertext '93, Seattle, Wash. ACM Press: 51-62.

Nation, David A. (1997):
Visualizing Websites Using a Hierarchical Table of Contents Browser: WebTOC.
3rd Conference on Human factors and the Web, Denver, USA.
ftp://ftp.cs.umd.edu/pub/hcil/Demos/WebTOC/Paper/WebTOC.html

Nation, David A. (1998): **WebTOC: A Tool To Visualize and Quantify Web Sites Using a Hierarchical Table of Contents Browser**.
Conf. on Human Factors in Computing Systems (CHI 98), Los Angeles, USA. ACM Press: 185-186.
http://www.cs.umd.edu/hcil/webtoc/

NCSA (1996, 28. März): **Mosaic(tm) for MS Windows User Guide.**
National Center for Superfcomputing Applications, University of Illinois, USA.
http://www.ncsa.uiuc.edu/SDG/Software/mosaic-w/releaseinfo/2.1/index.html, (seit Dezember 2000 nur noch im Internet Archive verfügbar: http://web.archive.org/web/200012072039/ http://www.ncsa.uiuc.edu/SDG/Software/mosaic-w/releaseinfo/2.1/index.html).

Nelson, Theodor H. (1965): **Complex information processing: a file structure for the complex, the changing and the indeterminate**.
20th National Conference of the ACM, Cleveland, Ohio, USA. ACM Press: 84-100.
http://dl.acm.org/citation.cfm?doid=800197.806036

Nelson, Theodor H. (1972): **As We Will Think**.
Online 72 Conference Proceedings, Brunel University, Uxbridge, England: 439-454.

Nelson, Theodor H. (1974): **Computer Lib/Dream Machines**. Mindful Press (Sausalito, CA, USA).

Nelson, Theodor H. (1987): **Literary Machines**. Mindful Press (Sausalito, CA, USA).

Nelson, Theodor H. (1999): **Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use**.
ACM Computing Surveys (CSUR), 31(4es).
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html

Nelson, Theodor H. (2004): **A Cosmology for a Different Computer Universe: Data Model, Mechanisms, Virtual Machine and Visualization Infrastructure**.
JoDI: Journal of Digital Information, 5(1): Article No. 298.
http://journals.tdl.org/jodi/article/viewArticle/131

Nestorov, Svetlozar, Serge Abiteboul & Rajeev Motwani (1998):
Extracting Schema from Semistructured Data.
Int. Conference On Management of Data (SIGMOD'98), Seattle, USA. ACM Press: 295 - 306.
http://dl.acm.org/citation.cfm?doid=276304.276331

Netcraft (2011, Oktober): **Netcraft Web Server Survey Archives.** Netcraft LTD (Bath, England).
http://news.netcraft.com/archives/web_server_survey.html

Netscape (1996): **JavaScript Guide for Netscape Navigator 3.0.**
Netscape Comminications Cooperation.
http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/ (seit April 2008 nur noch im Internet Archive verfügbar: http://web.archive.org/web/20080405005832/http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/).

Neumann, Peter G. & Lauren Weinstein (1999): **Inside Risks: Risks of Content Filtering**.
Communications of the ACM, 42(11): 152.
http://dl.acm.org/citation.cfm?doid=319382.319403

Neumüller, Moritz (2001): **Hypertext Semiotics in the Commercialized Internet**.
Ph.D. Thesis. Institut für Informationsverarbeitung und Informationswirtschaft, Wirtschaftsuniversität Wien (Österreich): 236 S.
http://sammelpunkt.philo.at:8080/23/

Newman, Jared (2010, 19. April): **Facebook's 'Like' Button: What We Know So Far.**
PCWorld (San Francisco, USA).
http://www.pcworld.com/article/194500/facebooks_like_button_what_we_know_so_far.html

Nichols, Peter (1987): **HyperCard.** „The Computer Chronicles" (TV Broadcast on PBS in the USA).
http://www.archive.org/details/CC501_hypercard

Nielsen, H. Frystyk, P. Leach & Scott Lawrence (2000): **RFC 2774: HTTP Extension Framework.**
The Internet Society.
http://www.w3.org/Protocols/HTTP/ietf-http-ext/

Nielsen, Henrik Frystyk, Jim Gettys, Anselm Baird-Smith, Eric Prud'hommeaux, et al. (1997):
Network Performance Effects of HTTP/1.1, CSS1, and PNG. World Wide Web Consortium.
http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html

Nielsen, Jakob (1988a): **Evaluating the Thinking Aloud Technique for use by Computer Scientists**.
IFIP Working Group 8.1 International workshop on Human Factors of Information Systems Analysis and Design, London, UK: 1-10.

Nielsen, Jakob (1988b, April): **Hypertext'87 Trip Report**. *ACM SIGCHI Bulletin*, 19(4): 27-35.
http://www.useit.com/papers/tripreports/ht87.html

Nielsen, Jakob (1989): **The Matters that Really Matter for Hypertext Usability**.
2nd Conference on Hypertext, Pittsburgh, Pennsylvania. ACM Press: 239-248.
http://dl.acm.org/citation.cfm?doid=74224.74244

Nielsen, Jakob (1990, März): **The Art of Navigating Through Hypertext**.
Communications of the ACM, 33(3): 296-310.
http://dl.acm.org/citation.cfm?doid=77481.77483

Nielsen, Jakob (1993a): **Hypertext and Hypermedia**. Academic Press Professional Inc.: 236 S.

Nielsen, Jakob (1993b): **Usability Engineering**. Morgan Kaufmann (San Francisco, USA). 3621 S.

Nielsen, Jakob (1994a): **Estimating the Number of Subjects Needed for a Thinking Aloud Test**. International Journal of Human Computer Studies, 41: 395-397.

Nielsen, Jakob (1994b):
Guerilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier, in Randolph G. Bias & Deborah J. Mayhew (ed.): "*Cost-Justifying Usability*". Morgan Kaufmann, Vol. Chapter 11: 245-272.
http://www.useit.com/papers/guerrilla_hci.html

Nielsen, Jakob (1994c): **Usability Inspection Methods**.
Human Factors in Computing Systems - CHI'94, Boston, Messachusetts. ACM Press: 413-414.
http://dl.acm.org/citation.cfm?doid=259963.260531

Nielsen, Jakob (1995a): **Features for the Next Generation of Web Browsers.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/9507.html

Nielsen, Jakob (1995b): **Multimedia and Hypertext: The Internet and Beyond**. Academic Press Professional: 480 S.

Nielsen, Jakob (1996, Mai): **Top Ten Mistakes in Web Design.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/9605.html

Nielsen, Jakob (1998a): **Using Link Titles to Help Users Predict Where They Are Going.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/980111.html

Nielsen, Jakob (1998b): **Web Pages Must Live Forever.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/981129.html

Nielsen, Jakob (1999a): **Designing Web Usability: The Practice of Simplicity**. New Riders Publishing.

Nielsen, Jakob (1999b): **URL as UI.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/990321.html

Nielsen, Jakob (1999c): **When Bad Design Elements Become the Standard.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/991114.html

Nielsen, Jakob (2000): **Is Navigation Useful?** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/20000109.html

Nielsen, Jakob (2002): **Intranet Usability: The Trillion-Dollar Question.** *Web*. useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/20021111.html

Nielsen, Jakob (2004a): **Change the Color of Visited Links.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/20040503.html

Nielsen, Jakob (2004b, Mai): **Guidelines for Visualizing Links.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/20040510.html

Nielsen, Jakob (2004c): **The Need for Web Design Standards.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/20040913.html

Nielsen, Jakob (2005): **Ten Usability Heuristics.** Nielsen Norman Group (Freemont, USA). http://www.useit.com/papers/heuristic/heuristic_list.html

Nielsen, Jakob (2008): **How Little Do Users Read?** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/percent-text-read.html

Nielsen, Jakob (2009a): **Mobile Usability.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/mobile-usability.html

Nielsen, Jakob (2009b): **Mobile Web 2009 = Desktop Web 1998.** useit Alertbox, Nielsen Norman Group (Freemont, USA). http://www.useit.com/alertbox/mobile-2009.html

Nielsen, Jakob (2010): **Website Response Times.** useit Alertbox, Nielsen Norman Group (Freemont USA). http://www.useit.com/alertbox/response-times.html

Nielsen, Jakob & Thomas K. Landauer (1993): **A Mathematical Model of the Finding of Usability Problems**. *INTERCHI '93*, Amsterdam, NL. ACM Press: 206-213.

Nielsen, Jakob & Rolf Molich (1990): **Heuristic Evaluation of User Interfaces**. *Conference on Human Factors in Computing Systems (CHI'90)*, Seattle, WA, USA. ACM Press: 249-256. http://dl.acm.org/citation.cfm?doid=97243.97281

Nielsen, Jakob & Victoria L. Phillips (1993): **Estimating the Relative Usability of Two Interfaces: Heuristic, Formal, and Empirical Methods Compared**. *CHI'93: Conference on Human Factors and Computing Systems*, Amsterdam, NL. ACM Press: 214-221. http://dl.acm.org/citation.cfm?doid=169059.169173

Nielsen, Jakob & Darrell Sano (1995): **SunWeb: User Interfaces Design for Sun Microsystem's Internal Web**. Computer Networks and ISDN Systems, 28(1&2): 179-188.

Nielsen, Janni, Torkil Clemmensen & Carsten Yssing (2002): **Getting Access to What Goes on in People's Heads? - Reflections on the Think-Aloud Technique**. *NordCHI*, Aarhus, Dänemark. ACM Press: 101-110. http://dl.acm.org/citation.cfm?doid=572020.572033

Nievergelt, Jürg (1983): **Die Gestaltung der Mensch-Maschine-Schnittstelle**. *13. GI -Jahrestagung*, Hamburg. Springer, Berlin: 41-50.

Noirhomme-Fraiture, Monique & Vincent Serpe (1998): **Visual Representation of Hypermedia Links According to Their Types**. *Advanced Visual Interfaces (AVI)*, L'Aquila, Italien. ACM Press: 146-155. http://dl.acm.org/citation.cfm?doid=948496.948516

Notess, Greg R. (2000, Februar): **Search Engine Statistics: Dead Links Report.** Search Engine Showdown. http://www.searchengineshowdown.com/statistics/dead.shtml

Notess, Greg R. (2003, Mai): **Search Engine Statistics: Freshness Showdown.** Search Engine Showdown. http://www.searchengineshowdown.com/statistics/freshness.shtml

Nürnberg, Peter J., Uffe K. Wiil & Kenneth M. Anderson (1997): **Open Hypermedia Systems Working Group Web Site.** Open Hypermedia Systems Working Group (OHSWG). http://www.csdl.tamu.edu/ohs/

Nyce, James M. & Paul Kahn (1991): **From Memex to Hypertext: Vannevar Bush and the Mind's Machine**. Academic Press, Inc. (San Diego, CA).

O'Hara, Kenton & Abigail Sellen (1997): **A Comparison of Reading Paper and On-Line Documents**. *Computer-Human Interaction (CHI '97)*, Atlanta, GA. ACM Press. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.4704

O'Reilly, Tim (2005): **What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.** O'Reilly Media, Inc. (Boston, MA, USA). http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Obendorf, Hartmut (2001): **Über den Umgang mit Links: Mögliche Auswirkungen der zweiten Generation von Links, XLink, auf das WWW**.
Diplomarbeit. Department of Informatics, University of Hamburg (Germany): 162 S.
http://agis-www.informatik.uni-hamburg.de/fileadmin/asi/Diplomarbeiten/Medieninformatik/DiplA_obendorf.pdf

Obendorf, Hartmut (2004): **The Indirect Authoring Paradigm – Bringing Hypertext into the Web**.
JoDI: Journal of Digital Information, 5(1): Article No. 249.
http://journals.tdl.org/jodi/article/viewArticle/132/130

Obendorf, Hartmut & Harald Weinreich (2003):
Comparing Link Marker Visualization Techniques - Changes in Reading Behavior.
12th Int. World Wide Web Conference - WWW 2003, Budapest, Ungarn. ACM Press: 736-745.
http://www2003.org/cdrom/papers/refereed/p391/p391-obendorf.html

Obendorf, Hartmut, Harald Weinreich & Torsten Haß (2004):
Automatic Support for Web User Studies with SCONE and TEA.
CHI '04: Human Factors in Computing Systems, Wien, Österreich. ACM Press: 1135-1138.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/199

Obendorf, Hartmut, Harald Weinreich, Eelco Herder & Matthias Mayer (2007): **Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Browser Usage**.
Human Factors in Computer Systems (CHI), San Jose, CA, USA. ACM Press: 597-606.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/280

Oinas-Kukkonen, Harri (1999): **Representing Metaknowledge through Rich Links**.
9th European-Japanese Conferences on Information Modelling and Knowledge Bases, Hachimantai, Japan. IOS Press, Amsterdam: 275-282.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.37

Ojakaar, Erik (2001): **Users Decide First; Move Second.**
User Interface Engineering, Massachusetts.
http://www.uie.com/articles/users_decide_first/

Ojakaar, Erik & Jared M. Spool (2001, Oktober): **Getting Them to What They Want**.
Rearch Report. User Interface Engineering. 19 S.

Padmanabhan, Venkata N. & Jeffrey C. Mogul (1995): **Improving HTTP latency**.
Computer Networks and ISDN Systems, 28(1/2): 25-35.
http://ntrg.cs.tcd.ie/undergrad/4ba2.96/group6/HTTPLatency.html

Padmanabhan, Venkata N. & Jeffrey C. Mogul (1996):
Using Predictive Prefetching to Improve World-Wide Web Latency.
Computer Communication Review: Proc. of SIGCOMM '96, 26(3): 22-36.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.2586

Paek, Tim, Susan Dumais & Ron Logan (2004): **WaveLens: A New View onto Internet Search Results**.
Human Factors in Computing Systems (CHI 2004), Wien, Österreich. ACM Press: 727-734.
http://dl.acm.org/citation.cfm?doid=985692.985784

Paepcke, Andreas, Hector Garcia-Molina & Gerard Rodriquez (1998):
Collaborative Value Filtering on the Web.
Seventh World-Wide Web Conference, Brisbane, Australia. Elsevier Science Publishers: 736-738.

Page, Lawrence, Sergey Brin, Rajeev Motwani & Terry Winograd (1998):
The PageRank Citation Ranking: Bringing Order to the Web.
Stanford Digital Library Working Paper. Standford University (Stanford, USA). 17 S.

Parunak, H. Van Dyke (1991): **Don't link me in: set based hypermedia for taxonomic reasoning**. *Hypertext '91*, San Antonio, Texas. ACM Press: 233-242. http://dl.acm.org/citation.cfm?doid=122974.122998

Payne, Stephen J. & T. R. G. Green (1986): **Task-Action Grammars: A Model of the Mental Representation of Task Languages**. Human-Computer Interaction, 2(2): 93-133.

Pearl, Amy (1989): **Sun's Link Service: a protocol for open linking**. *Hypertext '89*, Pittsburgh, Pennsylvania, United States. ACM Press: 137-146. http://dl.acm.org/citation.cfm?doid=74224.74236

Pemberton, Steven (2002): **XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition).** World Wide Web Consortium. http://www.w3.org/TR/xhtml1/

Pepper, Steve & Graham Moore (2000, August): **XML Topic Maps (XTM) 1.0.** TopicMaps.Org. http://www.topicmaps.org/xtm/

Petrasch, Roland (2007): **Model Based User Interface Design: Model Driven ArchitectureTM und HCI Patterns**. GI Softwaretechnik-Trends (Mitteilungen der Gesellschaft für Informatik), 27(3): 5-10.

Pettey, Christy & Holly Stevens (2011): **Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012.** Gartner Inc. (Edham, UK). http://www.gartner.com/it/page.jsp?id=1622614

Phelps, Thomas A. & Robert Wilensky (2000): **Robust Hyperlinks and Locations**. *D-Lib Magazine*, 6(7/8). http://www.dlib.org/dlib/july00/wilensky/07wilensky.html.

Pilgrim, Mark (2005): **Greasemonkey Hacks**. O'Reilly Media (Sebastopol, CA, USA). 469 S.

Pimienta, Daniel, Benoit Lamey, Daniel Prado & Marcelo Sztrum (2001): **The Fifth Study on Languages and the Internet.** Funredes (Santo Domingo, Dominikanische Republik). http://www.funredes.org/lc2005/english/L5/L5index_english.html

Pinkerton, Brian (1994): **Finding What People Want: Experiences with the WebCrawler**. *Second WWW Conference*, Chicago, USA. Elsevier. http://thinkpink.com/bp/WebCrawler/WWW94.html

Piotrowski, Arkadiusz M. Frydyada de & Michael J. Tauber (2009): **Benutzerprofile von Menschen mit Beeinträchtigungen/ Fähigkeiten**. *Mensch und Computer 2009*, Berlin. Oldenbourg Wissenschaftsverlag: 33-42. http://mc.informatik.uni-hamburg.de/konferenzbaende/mc2009/1_beitraege/mc2009_04_frydyada.pdf

Pirolli, Peter, James Pitkow & Ramana Rao (1996): **Silk from a Sow's Ear: Extracting Usable Structures from the Web**. *Conf. Human Factors in Computing Systems (CHI)*, Vancouver, Canada. ACM Press: 383-390. http://dl.acm.org/citation.cfm?doid=238386.238450

Pitkow, James E. (1998): **Summary of WWW Characterizations**. The Web Journal, 2(1): 3-13. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.3550

Pitkow, James E. & R. Kipp Jones (1996):
Supporting the Web: A Distributed Hyperlink Database System.
Computer Networks and ISDN Systems (Special Issue for the 5th WWW Conf.), 28(7-11): 981- 991.
http://smartech.gatech.edu/handle/1853/3495

POET (2000): **POET Object Server Suite: POET Java Programmer's Guide.**
POET Software GmbH (Hamburg, Deutschland).

Polson, Peter G., Clayton Lewis, John Rieman & Cathleen Wharton (1992):
Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces.
International Journal of Man-Machine Studies, 36(5): 741-773.

Postel, Jon (1981, September 1981):
Internet Protocol: DARPA Internet Program Protocol Specification, RFC 791.
http://www.faqs.org/rfcs/rfc791.html

Pree, Wolfgang (1994):
Meta Patterns - A means For Capturing the Essentials of Reusable Object-Oriented Design.
8th European Conf. on Object-Oriented Programming (ECOOP), Bologna, Italien. Springer: 150-162.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.7935

Price, Morgan N., Gene Glovchinsky & Bill N. Schilit (1998): **Linking By Inking: Trailblazing in a Paper-linke Hypertext**. *Hypertext '98*, Pittsburgh, PA. ACM Press: 30-39.

Puppe, Frank (1991): **Einführung in Expertextsysteme**. Springer Verlag (Berlin, Heidelberg). 215 S.

Rada, Roy (1990, August): **Hypertext Writing and Document Reuse: The Role of a Semantic Net**.
Electronic Publishing, 3(3): 125-140.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.3537

Rada, Roy (1991): **Hypertext: From Text to Expertext**. McGraw-Hill (Berkshire, UK). 230 S.

Rada, Roy, Phillip Ramsey & Antonios Michailidis (1993):
Educational Perspectives in Collaborative Hypermedia.
ACM Conference on Computer Science, Indianapolis, USA. ACM Press: 304-309.
http://dl.acm.org/citation.cfm?doid=170791.170848

Rademacher, Cay (2001, März): **Internet: Das Netz der Netze**. *GEO*, 3(2001): 67-82.

Raggett, Dave (1997, 14. Januar): **HTML 3.2 Reference Specification.**
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/REC-html32

Raggett, Dave, Arnaud Le Hors & Ian Jacobs (1999): **HTML 4.01 Specification.**
W3C Recommendation. World Wide Web Consortium.
http://www.w3.org/TR/html4/

Rajamony, Ramakrishnan & Mootaz Elnozahy (2001): **Measuring Client-Perceived Response Times on the WWW**. *3rd USENIX Symposium on Internet Technologies and System*, San Francisco, USA.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.7329

Ramakrishan, Naren (2000): **PIPE: Web Personalization by Partial Evaluation**.
IEEE Internet Computing(6/2000): 21-31.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.7249

Ramanathan, Praveen, Michael Bieber & Ajaz Rana (1997):
Providing Metainformation for World Wide Web Links and Documents.
Third Americas Conference on Information System - AIS'97 Conference, Indianapolis, Indiana.
Association for Information Systems: 143-145.
http://hsb.baylor.edu/ramsower/ais.ac.97/papers/ramanat.htm

Rao, Usha & Murray Turoff (1990): **Hypertext Functionality: A Theoretical Framework**. International Journal of Human-Computer Interaction, 4(2): 333-358.

Reese, George (2000): **Database Programming with JDBC & Java**. O'Reilly & Associates (Sebastopol, Kalifornien, USA). 348 S.

Reese, George (2003): **Java Database Best Practices**. O'Reilley & Associates (Sebastopol, Kalifornien, USA). 286 S.

Reich, Seigfied, Uff K. Wiil, Peter J. Nürnberg, Hugh C Davis, et al. (1999): **Addressing Interoperability in Open Hypermedia : the Design of the Open Hypermedia Protocol**. The New Review of Hypermedia and Multimedia 5: 207-248. http://www.daimi.au.dk/~kgronbak/homepage/pubs/Reich.pdf

Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom & John Riedl (1994): **GroupLens: An Open Architecture for Collaborative Filtering of Netnews**. *Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, USA. ACM Press: 185-186. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9351

Resnick, Paul & Jim Miller (1996): **PICS: Internet Access Controls Without Censorship**. Communications of the ACM, 39(10): 87-93. http://www.w3.org/PICS/iacwcv2.htm

Ridder, Isabelle De (2002): **Visible or Invisible Links: Does the Highlightning of Hyperlinks Affect Incitental Vocabulary Learning, Text Comprehension, and the Reading Process?** Language Learning & Technology, 6(1): 123-146. http://llt.msu.edu/vol6num1/pdf/deridder.pdf

Roberts, Scott (1999): **Programming Microsoft Internet Explorer 5**. Microsoft Press (Redmond, Washington). 511 S.

Robertson, George, Donald L. McCracken & Allen Newell (1979, Oktober): **The ZOG Approach to Man-Machine Communication**. *Technical Report*. Carnagie Mellon University, Department of Computer Science (Pittsburgh, PA).

Robertson, Stephen E. (1977): **Progress in Documentation: Theories and Models in Information Retrieval**. Journal of Documentation, 33: 126-148.

Roby, Warren B. (1999): **What's in a Gloss?** Language Learning & Technology, 2(2): 94-101. http://llt.msu.edu/vol2num2/roby/

Rodríguez-Mulà, Gerard, Hector García-Molina & Andreas Paepcke (1998): **Collaborative Value Filtering on the Web**. Computer Networks, 30(1-7): P9. http://ilpubs.stanford.edu:8090/468/

Roessler, Thomas (2002): **WHOIS: Datenschutz im DNS**. *Datenschutz und Datensicherheit (DuD)*, 26(11): 666-671. http://does-not-exist.org/roessler/dud-1102-whois-policy.pdf.

Roessler, Thomas & Anil Saldhana (2010): **Web Security Context: User Interface Guidelines.** *W3C Recommendation, 12 August 2010*. World Wide Web Consortium. http://www.w3.org/TR/wsc-ui/

Root, Robert W. & Steve Draper (1983): **Questionnaires as a Software Evaluation Tool**. *Computer Human Interaction (CHI)*. ACM Press: 83-87.

Roth, Wolf-Dieter (2006, 3. September): **"Web 2.0 ist nutzloses Blabla, das niemand erklären kann" - Tim Berners-Lee zum Hype des "neuen Web".** Heise Zeitschriften Verlag. http://www.heise.de/tp/r4/artikel/23/23472/1.html

Rötzer, Florian (1999, 19. Oktober): **Deep Linking**. *Telepolis - magazin der netzkultur*. http://www.heise.de/tp/deutsch/inhalt/te/5394/1.html.

Russo, Patricia & Stephen Boor (1993): **How Fluent is Your Interface? Designing for International Users**. *Conf. on Human Factors in Computing Systems (CHI 1993)*, Amsterdam, Niederlande. ACM Press: 342-347. http://dl.acm.org/citation.cfm?doid=169059.169274

Salamon, András (1998, 5. Februar): **Internet Statistics.** http://www.dns.net/andras/stats.html (seit Mai 2010 nur noch im Internet Archive verfügbar: http://web.archive.org/web/20100527215308/http://www.dns.net/andras/stats.html).

Salton, Gerard, Amit Singhal, Mandar Mitra & Chris Buckley (1997): **Automatic Text Structuring and Summarization**. Information Processing and Management, 33(2): 193-207.

Sandvad, Elmer, Kaj Grønbæk, Lennert Sloth & Jørgen Lindskov Knudsen (2001): **A Metro Map Metaphor for Guided Tours on the Web: the Webvise Guided Tour System**. *10th International World Wide Web Conference*, Hong Kong. ACM Press: 326-333. http://www10.org/cdrom/papers/pdf/p511.pdf

Sato, Daisuke, Shaojian Zhu, Masatomo Kobayashi, Hironobu Takagi & Chieko Asakawa (2011): **Sasayaki: Voice Augmented Web Browsing Experience**. *CHI 2011*, Vancouver, BC, Canada. ACM Press: 2769-2778. http://dl.acm.org/citation.cfm?doid=1978942.1979353

Sauter, Martin (2011): **Grundkurs Mobile Kommunikationssysteme**. Vieweg+Teubner Verlag: 420 S.

Scholtz, Jean & Sharon Laskowski (1998): **Developing Usability Tools and Techniques for Designing and Testing Web Site**. *4th Conference on Human Factors and the Web*, Basking Ridge, NJ, USA. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.2559

Schulze, Hendrik & Klaus Mochalski (2009): **Internet Study 2008/2009.** ipoque GmbH (Leipzig). http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf

Schwarzer, Bettina & Helmut Krcmar (2004): **Wirtschaftsinformatik. Grundzüge der betrieblichen Datenverarbeitung**. Schäffer-Poeschel (Stuttgart). 290 S.

Seybert, Heidi & Anna Lööf (2010): **Internet Usage in 2010 - Household and Individuals.** Eurostat (Luxemburg). http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-QA-10-050/EN/KS-QA-10-050-EN.PDF

Shafi, S. M. & Rafiq A. Rather (2005): **Precision and Recall of Five Search Engines for Retrieval of Scholarly Information in the Field of Biotechnology**. Webology, 2(2): Article 12. http://www.webology.ir/2005/v2n2/a12.html

Shafranovich, Yakov (2005): **RFC 4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files.** Network Working Group. http://www.ietf.org/rfc/rfc4180.txt

Shannon, Claude E. (1948): **A Mathematical Theory of Communication**. Bell System Technical Journal, 27: 379-423 und 623-656. http://dl.acm.org/citation.cfm?doid=584091.584093

Shenk, David (1997): **Data Smog, Surviving the Information Glut**. Harper (San Francisco). 288 S.

Shipman III, Frank M. & Catherine C. Marshall (1999): **Spatial Hypertext: An Alternative to Navigational and Semantic Links**. ACM Computing Surveys, 31(4es (Electronic Symposium)). http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/37.html

Shneiderman, Ben (1984): **Response Time and Display Rate in Human Performance with Computers**. ACM Computing Surveys (CSUR) 16(3): 265-285. http://dl.acm.org/citation.cfm?doid=2514.2517

Shneiderman, Ben (1992): **Designing the User Interface: Strategies for Effective Human-Computer Interaction**. Addison Wesley (Reading, Massachusetts). 2. Auflage: 573 S.

Shneiderman, Ben (1997): **Designing the User Interface: Strategies for Effective Human-Computer Interaction**. Addison Wesley (Reading, Massachusetts). 3. Auflage: 640 S.

Shneiderman, Ben & Greg Kearsley (1989): **Hypertext Hands-On: An Introduction to a New Way of Organizing and Accessing Information**. Addison-Wesley Publ. (Reading, MA.).

Shum, Simon Buckingham (1996): **The Missing Link: Hypermedia Usability Research & The Web**. SIGCHI Bulletin, 28(4): 68-75.

Sidler, Gabriel, Andrew Scott & Heiner Wolf (1997): **Collaborative Browsing in the World Wide Web**. *8th Joint European Networking Conference*, Edinburgh, Großbritannien: 122/1-122/8. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.904

Siegel, David (1999): **Web Site Design: Creating Killer Web Sites**. Zweitausendeins (Frankfurg a. M.). 305 S.

Siegmund, Katja (2008): **Accessibility-orientierte Anforderungsanalyse (AORA)**. i-com: Zeitschrift für interaktive und kooperative Medien, 7(2/2008): 34-40.

Sivasubramanian, Swaminathan, Michal Szymaniak, Guillaume Pierre & Maarten van Steen (2004): **Replication for Web Hosting Systems**. ACM Computing Surveys, 36(3): 291-334. http://dl.acm.org/citation.cfm?doid=1035570.1035573

Smith, Pauline A., Ian A. Newman & Lesley M. Parks (1997): **Virtual hierarchies and virtual networks: some lessons from hypermedia usability research applied to the World Wide Web**. International Journal of Human-Computer Studies, 47(1): 67-95. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.7508

Smith, Robert E. & William R. Swinyard (1982): **Information Response Models: An Integrated Approach**. Journal of Marketing, 46(1): 81-93.

Sneed, Harry M. & Stefan Jungmayr (2011): **Mehr Testwirtschaftlichkeit durch Value-Driven-Testing**. informatik-Spektrum, 34(2/2011): 192-209.

Sommerville, Ian (2004): **Software Engineering**. Pearson (Harlow, Essex, England). 7th Edition: 963 S.

Specht, Günther (1997): **Complexity Analysis of Link Navigation in Dexer Based Hypermedia Database Systems**. International Journal Informatica, 8(1): 23-42.

Sperberg-McQueen, C. Michael & Lou Burnard (2004): **TEI P4: Guidelines for Electronic Text Encoding and Interchange.** Text Encoding Initiative Consortium. http://www.tei-c.org/P4X/

Spiliopoulou, Myra, Bamshad Mobasher, Bettina Berendt & Miki Nakagawa (2003): **A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis**. INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications, 15(2): 171-190. http://maya.cs.depaul.edu/mobasher/papers/SMBN03.pdf

Spool, Jared, Taran Scanlon, Will Schroeder, Carolyn Snyder & Terri DeAngelo (1998): **Web Site Usability: A Designer's Guide**. Morgan Kaufmann Publishers (San Francisco, CA). 156 S.

Spool, Jared & Will Schroeder (2001): **Testing Web Sites: Five Users Is Nowhere Near Enough**. *CHI Conf. on Human Factors in Computing Systems (Extended Abstracts)*, Seattle, USA. ACM Press: 285 - 286. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.4127

Spyridakis, Jan H. (1989): **Signaling Effects: Increased Content Retention and New Answers - Part II**. Technical Writing and Communication 19(4): 395-415.

Stanyer, Dominic & Rob Procter (1999): **Improving Web Usability with the Link Lens**. J ournal of Computer Networks and ISDN Systems (Proceedings of the 8th International WWW Conference, Toronto), 31(11-16): 455-466.

Stein, Lincoln & Doug MacEachern (1999): **Writing Apache Modules with Perl and C**. O'Reilly & Associates (Sebastopol, Kalifornien, USA). 724 S.

Stephan, Björn (2002): **Vergleich von Objektpersistenzmechanismen für Java: Integration der Object Server Suite von Poet in Scone**. *Studienarbeit*. Fachbereich Informatik, Universität Hamburg (Hamburg): 48 S.

Stevens, Albert & Patty Coupe (1978): **Distortions in Judged Spatial Relations**. Cognitive Psychology, 10(4): 422-437. http://dx.doi.org/10.1016/0010-0285(78)90006-3

Strauch, Thomas & Niklas Luhmann (1989): **Philosophie Heute: Niklas Luhmann - Beobachter im Krähennest.** Junius Verlag GmbH (VHS Video, 45 min).

Streitz, Norbert, Jörg M. Haake, Jörg Hannemann, Andreas Lemke, et al. (1992): **SEPIA: A Cooperative Hypermedia Authoring Environment**. *ACM ECHT*, Mailand, Italien. ACM Press: 11-22.

Strube, Gerhard (1984): **Assoziation: Der Prozeß des Erinnerns und die Struktur des Gedächtnisses**. Springer Verlag (Berlin).

Suignard, Michel & Martin Dürst (2005): **Internationalized Resource Identifiers (IRIs).** *Request for Comments: 3987*. Network Working Group. http://tools.ietf.org/html/rfc3987

Sullivan, Danny (2006): **Hitwise Search Engine Ratings.** Search Engine Watch, Incisive Interactive Marketing LLC (New York, USA). http://searchenginewatch.com/article/2067666/Hitwise-Search-Engine-Ratings

Sullivan, Danny (2007): **How To Use HTML Meta Tags.** Search Engine Watch, Incisive Interactive Marketing LLC (New York, USA). http://searchenginewatch.com/article/2067564/How-To-Use-HTML-Meta-Tags

Sullivan, Terry (1999): **The Need for Speed.** All Things Web. http://www.pantos.org/atw/speed.html

Swank, Mark & Drew Kittel (1996): **Designing & Implementing Microsoft Index Server**. SAMS.NET Publishing (Indianapolis). 350 S.

Tabard, Aurélien, Wendy Wackay, Nicals Roussel & Catherine Letondal (2007): **PageLinker: Integrating Contextual Bookmarks within a Browser**. *Human Factors in Computing Systems (CHI 2007)*, San Jose, USA. ACM Press: 337 - 346. http://dl.acm.org/citation.cfm?doid=1240624.1240680

Takano, Hajime & Terry Winograd (1998): **Dynamic Bookmarks for the WWW**. *Hypertext '98*, Pittsburgh, PA. ACM Press: 297-298. http://dl.acm.org/citation.cfm?doid=276627.276667

Tanenbaum, Andrew S. (2003): **Computer Networks**. Prentice Hall: 912 S.

Tanenbaum, Andrew S. & Maarten van Steen (2002): **Distributed Systems: Principles and Paradigms**. Prentice Hall (New Jersey, USA). 1. Ausgabe: 803 S.

Tauscher, Linda & Saul Greenberg (1997): **How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems**. Int. J. Human-Computer Studies, 47(1): 97-137.

Tauscher, Linda Marie (1996): **Evaluating History Mechanisms: An Empirical Study of Reuse Patterns in World Wide Web Navigation**. *Master's Thesis*. Department of Computer Science, University of Calgary (Calgary, Alberta): 173 S.

Technologie, Bundesministerium für Wirtschaft und & TÜV Rheinland (2011): **Breitbandatlas 2010.** Bundesministerium für Wirtschaft und Technologie (BMWi), Abt. Öffentlichkeitsarbeit (Berlin). http://www.zukunft-breitband.de/BBA/Navigation/service,did=424764.html

Teevan, Jaime (2004, 18. Juni): **How People Re-find Information When the Web Changes**. *Technical Report*. Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory (Cambridge, MA, USA). 10 S.

Teevan, Jaime (2007): **The Re:Search Engine: Simultaneous Support for Finding and Re-Finding**. *UIST'07*, Newport, Rhode Island, USA. ACM Press: 23-32.

Teevan, Jaime, Eytan Adar, Rosie Jones & Michael A.S. Potts (2007): **Information Re-Retrieval: Repeat Queries in Yahoo's Logs**. *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, Amsterdam, Niederlande. ACM Press. http://people.csail.mit.edu/teevan/work/publications/papers/sigir07.pdf

Teevan, Jaime, Edward Cutrell, Danyel Fisher, Steven M. Drucker, et al. (2009): **Visual Snippets: Summarizing Web Pages for Search and Revisitation**. *27th International Conference on Human Factors in Computing Systems - CHI 2009*, Boston, MA, USA. ACM Press: 2023-2032. http://dl.acm.org/citation.cfm?doid=1518701.1519008

Teevan, Jaime, Susan Dumais & Dan Liebling (2010): **A Longitudinal Study of How Highlighting Web Content Change Affects People's Web Interactions**. *2010 ACM Conf. on Human Factors in Computing Systems (CHI 2010)*, Atlanta, Georgia, USA. ACM Press: 1353-1356. http://research.microsoft.com/apps/pubs/default.aspx?id=130996

Thistlewaite, Paul (1997): **Automatic construction and management of large open webs**. Information Processing and Management, 33(2): 161-173.

Thompson, Henry S., David Beech, Murray Maloney & Noah Mendelsohn (2004): **XML Schema Part 1: Structures Second Edition.** World Wide Web Consortium. http://www.w3.org/TR/xmlschema-1/

Thüring, Manfred, Jörg M. Haake & Jörg Hannemann (1991): **What's Eliza Doing in the Chinese Room? Incoherent hyperdocuments - and how to avoid them**. *Hypertext und Hypermedia'91*, Graz. Springer Verlag: 119-134. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.3045

Thüring, Manfred, Jörg Hannemann & Jörg M. Haake (1995, August): **Hypermedia and Cognition: Designing for Comprehension**. *Communications of the ACM*, 38(8): 57-66.

TMG (2007): **Telemediengesetz (TMG)**.
Bundesgesetzblatt Jahrgang 2007, Teil I, Nr. 6 (Bonn): S. 179-185.
http://www.bgbl.de/Xaver/start.xav?startbk=Bundesanzeiger_BGBl&bk=Bundesanzeiger_BGBl&start=//*[@attr_id=%27bgbl107s0179.pdf%27]

Tognazzini, Bruce (1998): **Ask Tog, December 1998: Representing link destinations.**
Nielsen Norman Group.
http://www.asktog.com/readerMail/1998-12ReaderMail.html#RepresentingLinkDesinations

Tomek, Ivan & Hermann Maurer (1992): **Helping the User to Select a Link**.
The New Review of Hypermedia and Multimedia, 4(2): 111-122.

Touch, Joe, John Heidemann & Katia Obraczka (1998): **Analysis of HTTP Performance**.
Research Report. USC / Information Sciences Institute. 9 S.

Trigg, Randall H. (1983):
A Networked Approach to Text Handling for the Online Scientific Community.
Ph.D. Thesis. Computer Science, University of Maryland (College Park, MD).
http://www.workpractice.com/trigg/thesis-default.html

Trigg, Randall H. (1988): **Guided tours and tabletops: tools for communicating in a hypertext environment**. Transactions on Information Systems (TOIS), 6(4): 398-414.
http://dl.acm.org/citation.cfm?doid=58566.59299

Trigg, Randall H. & Peggy M. Irish (1987): **Hypertext Habitats: Experiences of Writers in NoteCards**.
Hypertext '87, Chapel Hill, North Carolina, USA. ACM Press: 89 - 108.
http://dl.acm.org/citation.cfm?doid=317426.317435

Trigg, Randall H., Lucy A. Suchman & Frank G. Halasz (1986):
Supporting Collaboration in Notecards.
Conference on Computer-Supported Cooperative Work, Austin, Texas, USA. ACM Press: 153-162.
http://dl.acm.org/citation.cfm?doid=637069.637089

Trigg, Randall H. & Mark Weiser (1986): **TEXTNET: a network-based approach to text handling**.
ACM Transactions on Information Systems (TOIS), 4(1): 1-23.
http://dl.acm.org/citation.cfm?doid=5401.5402

Tulving, Endel & D.M. Thompson (1973):
Encoding specificity and retrieval processes in episodic memory.
Psychological Review, 80: 352-373.

UMTS-Report (2007, 1. März): **Steigende Nutzerzahlen im mobilen Internet.**
BörseGo AG (München).
http://www.umts-report.de/news.php?ida=572228&idc=281

Utting, Kenneth & Nicole Yankelovich (1989): **Context and Orientation in Hypermedia Networks**.
ACM Transactions on Information Systems (TOIS), 7(1): 58-84.

Vainio-Larsson, Arja (1989): **Hypermedia, a Vision for Human-Computer Interaction**. *LIBLAB Research Report*. Linköping University (Linköping, Denmark).

Van Dam, Andries (1971): **On-line Text Editing: A Survey**. Computing Surveys, 3(3): 93-114.

Van Dam, Andries (1988, July): **Hypertext '87 Keynote Address**.
Communications of the ACM, 31(7): 887-895.
http://dl.acm.org/citation.cfm?doid=48511.48519

Verbyla, Janet (1999): **Unlinking the Link**. ACM Computing Surveys, 31(4es).
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/61.html

Vitali, Fabio & Michael Bieber (1999): **Hypermdia on the Web: What Will It Take?**
ACM Computing Surveys, 31(4es).
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/57.html

Vitali, Fabio, Chao-Min Chiu & Michael Bieber (1997):
Extending HTML in a Principled Way with Displets.
Computer Networks and ISDN Systems, 29(8-13): 1115-1128.
http://www.cs.unibo.it/~fabio/bio/papers/1997/WWW97/Displets/PAPER155.html

Vora, Pawan R., Martin G. Helander & Valerie L. Shalin (1994):
Evaluating the Influence of Interface Styles and Multiple Access Paths in Hypertext.
CHI: Human Factors in Computing Systems, Boston, USA. ACM Press: 323-329.
http://dl.acm.org/citation.cfm?doid=191666.191777

Wal, Thomas Vander (2007): **Folksonomy Coinage and Defintion.** vanderwal.net.
http://www.vanderwal.net/folksonomy.html

Walker, Janet H. (1987): **Document Examiner: Delivery Interfaces for Hypertext Documents**.
ACM Hypertext '87, Chapel Hill, USA. ACM Press: 307-323.
http://dl.acm.org/citation.cfm?doid=317426.317448

Wan, Ernest, Philip Robertsin, John Brook, Stephen Bruce & Kristine Armitage (1999):
Retaining Hyperlinks in Printed Hypermedia Document.
Computer Networks, 31(11-16): 1509 - 1524.

Wang, Weigang & Jörg Haake (1997): **Supporting User-Defined Activity Spaces**.
Hypertext'97, Southampton, UK. ACM Press: 112-123.

Wang, Weigang & Roy Rada (1995): **Experiences with Semantic Net Based Hypermedia**.
International Journal of Human Computer Studies, 43(3): 419-439.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.495

Warth, Dora (1999):
Praktische Umsetzung von Hypertext-Forschungsergebnissen in HTML-Publishing.
Diplomarbeit. Angewandte Sprachwissenschaft, Johannes Gutenberg-Universität (Mainz).
http://www.fask.uni-mainz.de/user/warth/hypertext/diplom/Hypertext.html

Weibel, Stuart, John Kunze, Carl Lagoze & Misha Wolf (1998):
Dublin Core Metadata for Resource Discovery.
Technical Report RFC 2413. Internet Engineering Task Force (IETF).
http://dublincore.org/documents/1998/09/dces/

Weinreich, Harald (1997): **Ergonomie von Hypertext-Systemen und das World Wide Web**.
Diplomarbeit. Fachbereich Informatik, Universität Hamburg (Hamburg): 240 S.
http://vsis-www.informatik.uni-hamburg.de/publications/viewpub.phtml/41

Weinreich, Harald, Volkert Buchmann & Winfried Lamersdorf (2003):
Scone: Ein Framework zur evaluativen Realisierung von Erweiterungen des Webs.
Kommunikation in Verteilten Systemen - KiVS 2003, Leipzig, Deutschland. Springer: 31-42.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/111

Weinreich, Harald & Winfried Lamersdorf (2000):
Concepts for Improved Visualization of Web Link Attributes.
Computer Networks, 33(1-6): 403-416.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/82

Weinreich, Harald & Winfried Lamersdorf (2003):
Eine Umfrage zu Link- und Objekt-Attributen im Web.
Mensch und Computer 2003, Stuttgart. B.G. Teubner Verlag: 387-389.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/121

Weinreich, Harald, Hartmut Obendorf & Eelco Herder (2006a):
Data Cleaning Methods for Client and Proxy Logs.
World Wide Web Conference 2006, Workshop Proceedings: Logging Traces of Web Activity, Edinburgh, Scotland, UK: 4 S.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/267

Weinreich, Harald, Hartmut Obendorf, Eelco Herder & Matthias Mayer (2006b):
Off the Beaten Tracks: Exploring Three Aspects of Web Navigation.
WWW Conference 2006, Edinburgh, Großbritannien. ACM Press: 133-142.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/263

Weinreich, Harald, Hartmut Obendorf, Eelco Herder & Matthias Mayer (2008):
Not Quite the Average: An Empirical Study of Web Use.
ACM TWEB - Transactions on the Web, 2(1): 26 S.
http://dl.acm.org/citation.cfm?doid=1326561.1326566

Weinreich, Harald, Hartmut Obendorf & Winfried Lamersdorf (2001):
The Look of the Link - Concepts for the User Interface of Extended Hyperlinks.
12th ACM Conference on Hypertext and Hypermedia, Aarhus, Dänemark. ACM Press: 19-28.
http://vsis-www.informatik.uni-hamburg.de/publications/viewpub.phtml/87

Weinreich, Harald, Hartmut Obendorf & Winfried Lamersdorf (2003): **HyperScout: Darstellung erweiterter Typinformationen im World Wide Web – Konzepte und Auswirkungen**.
Mensch und Computer 2003, Stuttgart. Teubner Verlag.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/120

Weinreich, Harald, Hartmut Obendorf & Winfried Lamersdorf (2004):
HyperScout: Linkvorschau im World Wide Web.
i-com: Zeitschrift für interaktive und kooperative Medien, 3(1): 4-12.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/202

Weinreich, Harald, Hartmut Obendorf, Matthias Mayer & Eelco Herder (2006c):
Der Wandel in der Benutzung des World Wide Webs.
Mensch und Computer 2006, Gelsenkirchen. Oldenbourg Wissenschaftsverlag: 155-164.
http://vsis-www.informatik.uni-hamburg.de/publications/view.php/271

Weis, Erich, Erwin Weis & Heinrich Mattutat (1967):
Schöffler/Weis: Wörterbuch der englischen und deutschen Sprache.
Ernst Klett Verlag (Stuttgart). 1. Auflage.

Weissinger, A. Keyton (2000): **ASP in a Nutshell**.
O'Reilly & Associates (Sebastopol, Kalifornien, USA). 492 S.

Weßendorf, Matthias (2011, 15. Juni):
Besser jetzt als gleich - WebSocket: Annäherung an Echtzeit im Web.
Heise Zeitschriften-Verlag (Hannover). http://heise.de/-1260189

Wexelblat, Alan (1999): **History-Based Tools for Navigation**.
Hawai'i International Conference on System Sciences (HICSS '99). IEEE Press.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.1298

Wharton, Cathleen, John Rieman, Clayton Lewis & Peter G. Polson (1994):
The Cognitive Walkthrough Method: A Practitioner's Guide.
John Wiley & Sons Inc. (New York, USA). 105-140 S.

Wheeler, David A. (2002): **More than a Gigabuck: Estimating GNU/Linux' Size.**
"dwheeler.com" (TM).
http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html

Wiener, Norbert (1963): **Kybernetik. Regelung und Nachrichtenübertragung im Lebewesen und in der Maschine**. Econ-Verlag GmbH (Düsseldorf, Wien). 2. revidierte und ergänzte Auflage: 287 S.

Wigdor, Daniel & Dennis Wixon (2011): **Brave NUI World - Designing Natural User Interfaces for Touch and Gesture**. Elsevier Science & Technology (Burlington, Massachusetts, USA). 242 S.

Wiil, Uffe K. (2000): **Towards a Proposal for a Standard Component-Based Open Hypermedia System Storage Interface**. *OHS-6 and SC-2*. Springer Verlag: 23-30.

Wiil, Uffe Kock (1997): **Open Hypermedia: Systems, Interoperability and Standards**.
Journal of Digital Information, 1(2).
http://journals.tdl.org/jodi/article/viewArticle/9/9

Wilde, Erik & David Lowe (2000): **From Content-centered Publishing to a Link-based View of Information Resources**. *Hawai'i International Conference On System Sciences (HICSS)*, Maui, Hawaii, USA. IEEE Publishing: 1-10.

Wilson, Brian (2008): **MAMA: What is the Web made of?** *Report*. Opera Software.
http://dev.opera.com/articles/view/mama/

Witt, Rebecca J. & Susan P. Tyerman (2002): **Reducing Cognitive Overhead on the World Wide Web**.
25th Australasian Conference on Computer Science, Melbourne. ACM Press: 311-320.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.502

Wolf, Gary (1995, Juni): **The Curse of Xanadu**. *Wired*, 3(6).
http://www.wired.com/wired/archive/3.06/xanadu_pr.html

Wolff, Christian & Stefanie Götzfried (2009):
"I can"-Design: Methodik für das benutzerzentrierte Design nicht-standardisierte Icons.
7. Workshop des German Chapters der Usability Professionals Association e.V., Berlin, Deutschland.
Fraunhofer Verlag, Stuttgart: 119-123.

Wollenweber, Frank (2002): **Entwicklung eines generischen Robots für das Scone-Framework**.
Studienarbeit. AG VSIS - Fachbereich Informatik, Universität Hamburg: 78 S.
http://vsis-www.informatik.uni-hamburg.de/publications/viewThesis.php/131

Wollenweber, Frank (2004): **Kollaborative Nutzung des World Wide Web**.
Diplomarbeit. AG VSIS - Fachbereich Informatik, Universität Hamburg: 213 S.
http://vsis-www.informatik.uni-hamburg.de/getDoc.php/thesis/132/FrankWdiplomarbeit-final-070104.pdf

Won, Sungjoon Steve, Jing Jin & Jason I. Hong (2009):
Contextual Web History: Using Visual and Contextual Cues to Improve Web Browser History.
Human Factors in Computing Systems (CHI 2009), Boston, MA, USA. ACM Press: 1457-1466.
http://dl.acm.org/citation.cfm?doid=1518701.1518922

Woodhead, Nigel (1991): **Hypertext and Hypermedia: Theory and Applications**.
Sigma Press (Wilmslow, England).

Woodruff, Allison, Andrew Faulring, Ruth Rosenholtz, Julie Morrison & Peter Pirolli (2001):
Using Thumbnails to Search the Web.
Int. Conference on Computer Human Interaction (CHI'01), Seattle, WA, USA. ACM Press: 198-205.

Wooldridge, Michael (2002): **Introduction to MultiAgent Systems**.
John Wiley & Sons (New York). 256 S.

Wörndl, Wolfgang, Roland Bader & Johann Schlichter (2011):
Kontextsensitive Recommender Systeme in mobilen Anwendungsbereichen.
i-com - Zeitschrift für interaktive und kooperative Medien, 10(1/2011): 26-33.

Wu, Xian, Lei Zhang & Yong Yu (2006): **Exploring Social Annotation for the Semantic Web**.
15th International World Wide Web Conference, Edinburgh. ACM Press: 417-426.
http://dl.acm.org/citation.cfm?doid=1135777.1135839

Xiao, Lun (1995): **An Object-Oriented Extensible Transaction Management System**.
Ph.D. Thesis. Computer Science, University of Illinois at Urbana-Champaign (Illinois): 131 S.
http://srg.cs.uiuc.edu/Bib/LXiao-PhD.thesis.pdf

Yankelovich, Nicole, Bernard J. Haan, Norman Meyrowitz & Steven M. Drucker (1988):
Intermedia: The Concept and the Construction of a Seamless Information Environment.
IEEE Computer, 21(1): 81-96.

Yankelovich, Nicole, Norman Meyrowitz & Andries Van Dam (1985): **Reading and Writing the Electronic Book**. IEEE Computer, 18(10): 15-30.

Yee, Ka-Ping (2002): **CritLink: Advanced Hyperlinks Enable Public Annotation on the Web**.
Computer-Supported Co-operative Work. ACM Press.
http://zesty.ca/pubs/cscw-2002-crit.pdf

Yergeau, Francois (1998): **RFC2279: UTF-8, a transformation format of ISO 10646.**
Network Working Group.
http://www.ietf.org/rfc/rfc2279.txt

Yesilada, Yeliz, Darren Lunn & Simon Harper (2007): **Experiments Toward Reverse Linking on the Web**. *Conference on Hypertext and Hypermedia*, Manchester, England. ACM Press: 3-10.
http://dl.acm.org/citation.cfm?doid=1286240.1286244

ZDF (2011): **ARD/ZDF-Onlinestudien.** ZDF Pressestelle (Mainz, Frankfurt a. M.).
http://www.ard-zdf-onlinestudie.de/index.php?id=264

Zeinalipour-Yazti, Demitris & Marios Dikaiakos (2001, Juni): **High-Performance Crawling and Filtering in Java**. *Technical Report*. Department of Computer Science, University of Cyprus. 23 S.

Zellweger, Polle, Bay-Wie Chang & Jock Mackinlay (1998): **Fluid Links for Informed and Incremental Link Transitions**. *Hypertext '98*, Pittsburgh, PA. ACM Press: 50-57.
http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.168.4891

Zellweger, Polle T. (1989): **Scripted Documents: A Hypermedia Path Mechanism**.
2nd Conference on Hypertext and Hypermedia, Pittsburgh. ACM Press: 1-14.
http://dl.acm.org/citation.cfm?doid=74224.74225

Zellweger, Polle T., Niels Olof Bouvin, Henning Jehøj & Jock D. Mackinlay (2001):
Fluid Annotations in an Open World.
Hypertext'01 - Conference on Hypertext and Hypermedia, Aarhus, Dänemark. ACM Press: 9-18.
http://dl.acm.org/citation.cfm?doid=504216.504224

Zellweger, Polle T., Susan Harkness Regli, Jock D. Mackinlay & Bay-Wei Chang (2000):
The Impact of Fluid Documents on Reading and Browsing: An Observational Study.
CHI 2000 Conf. on Human factors in computing systems, Den Hague, The Netherlands.
ACM Press: 249-256.
http://dl.acm.org/citation.cfm?doid=332040.332440

Zhang, Haimo & Shengdong Zhao (2011): **Measuring Web Page Revisitation in Tabbed Browsing**. *CHI 2011 - Annual Conference on Human Factors in Computing Systems*, Vancouver, Canada. ACM Press: 1831-1834. http://dl.acm.org/citation.cfm?doid=1925820.1925836

Zhang, Weifeng, Baowen Xu, William Song, Hongji Yang & Kecheng Liu (2000): **Data Mining Algorithms for Web Pre-Fetching**. *WISE: Conference on Web Information Systems Engineering*, Hong Kong. IEEE Computer Society: 34-38.

Zhi, Jing (2001): **Web Page Design and Download Time**. *27. International CMG Conference*, Anaheim, CA, USA: 689-704. http://www.cmg.org/downloads/JingZhi.pdf

Zirpins, Christian, Harald Weinreich, Andreas Bartelt & Winfried Lamersdorf (2001): **Advanced Concepts for Next Generation Portals**. *12th International Workshop on Database and Expert Systems Applications*, München. IEEE Computer Society: 501-506. http://vsis-www.informatik.uni-hamburg.de/publications/view.php/88

Zschau, Oliver, Dennis Traub & Rik Zahradka (2002): **Web Content Management. Websites professionell planen und betreiben**. Gallileo Business Press (Bonn). 2. Ausgabe: 432 S.

Zühlsdorff, Arne (2002): **Konzepte zur Repräsentation von Webseiten**. *Diplomarbeit*. VSIS, Fachbereich Informatik, Universität Hamburg: 108 S. http://vsis-www.informatik.uni-hamburg.de/members/info.php/188

Abbildungsverzeichnis

Viele der Abbildungen wurden vom Autor dieser Arbeit überarbeitet oder gemäß der Originalabbildung neu erstellt, da die Vorlage nicht in guter Qualität erhältlich war oder weil bestimmte Aspekte hervorgehoben werden sollten. Dies wird in der Referenz durch „nach“ gekennzeichnet.

Abb. 1: Assoziationen verknüpfen im Gedächtnis vorhandene Eindrücke. 11

Abb. 2: Drei Knoten (bzw. Dokumente), die durch zwei Hyperlinks miteinander verknüpft sind. Der Leser kann vom Knoten A direkt zu B und C springen. 15

Abb. 3: Links ein Ausschnitt aus dem Brief an die Korinther, rechts die „Certaine Annotations“ (nach DeRose & Durand 1994)..... 16

Abb. 4: Bei Wan et al. sorgen Kerben in den Seiten für die schnelle Navigation (nach Wan, Robertsin et al. 1999)..... 17

Abb. 5: Bildschirmfoto des VIKI-Systems (nach Marshall & Shipman III 1995). Die Kästen repräsentieren die Knoten des Hypertextes; Attribute wie Anordnung, Schachtelung und Färbung drücken ihre Relation zueinander aus. 20

Abb. 6: Schematische Darstellung unterschiedlicher Link-Eigenschaften: Knoten A bindet Objekte X und Y mittels Transklusionen ein. Ein assoziativer Link setzt Knoten A und Knoten B in inhaltliche Beziehung. Ein programmatischer Link startet oder steuert das Programm P. Von Knoten B aus führen strukturelle Links zu den Unterknoten B.1 bis B.3. 23

Abb. 7 Zwei Links unterschiedlicher Direktionalität. 23

Abb. 8: Zwei Links unterschiedlichen Grades. 24

Abb. 9: Typisierte Links spezifizieren die Relation zwischen Link-Start (Knoten 1) und Link-Zielen (Knoten 2 und 3). 24

Abb. 10: Knoten A bietet drei Link-Anker. Die ersten beiden verweisen jeweils auf die ganzen Knoten B und C, der dritte Anker bezieht sich auf Anker 4 innerhalb von Knoten C. Link-Anker 2 hat zusätzlich einen Link-Kontext, der die Bedeutung des Links genauer spezifiziert. Als Link-Marker dienen in dieser Illustration hellblaue Kästchen mit dunkelblauem Rahmen und blauem, unterstrichenem Text. 24

Abb. 11: Schematische Darstellung eines verteilten Hypertext-Informationssystems mit zwei Servern. Der Benutzer navigiert mithilfe der Links von Dokument A eines Anbieters zu den Dokumenten B und C eines zweiten Anbieters. 27

Abb. 12: Übersicht über die Aktionen der Studienteilnehmer der Web-Browsing-Studie von 2005. Links die Anteile aller Navigationsaktionen, rechts die unterschiedlichen Link-Aktionen (Weinreich, Obendorf et al. 2008). 33

Abb. 13: Anteil geänderter Seiten für unterschiedliche Wiederbesuchszeiten (nach Weinreich, Obendorf et al. 2008). 36

Abb. 14: Das Information-Retrieval-Modell von Robertson (nach Robertson 1977). 41

Abb. 15: Oben die typische „minimalistische“ Benutzungsschnittstelle für die Suche im Web, unten die Recherche-Schnittstelle einer Bibliotheksdatenbank, die eine wesentlich präzisere Anfrage erlaubt..... 43

Abb. 16: Links die hierarchische Ansicht des Hyper-G Systems (s. Anhang B.14; Andrews 1996), rechts die Übersichtsseite des hierarchisch strukturierten Open Directory Projects DMOZ. 44

Abb. 17: Zwei Werkzeuge zur Visualisierung von Web-Ressourcen: Links eine 2D-Übersicht bestimmter Dokumenttypen im „Navigational View Builder“ (Mukherjea & Foley 1995), rechts die 3D-Ansicht einer Website mit dem System „Narcissus“ (Hendley, Drew et al. 1995). 46

Abb. 18: Oben das Bookmark-Menü und der Bookmark-Toolbar, links unten die History im Sidebar von Firefox 3. 49

Abb. 19: Verzögerung zwischen der Navigationsaktion des Benutzers und der Verarbeitung der ersten Daten im Webbrowser (Weinreich, Obendorf et al. 2006b).....59

Abb. 20: Verteilung der Übertragungszeiten von Webseiten bei unterschiedlichen Bandbreiten. 60

Abb. 21: Ein „Extended Validation Certificate“ belegt die Identität des Betreibers einer Website.....66

Abb. 22: Der Firefox-Browser mit den Extensions NetCraft Toolbar (oben), McAfee SiteAdvisor und HTML-Validator (beide rechts unten im Statusbereich). 67

Abb. 23: Illustration der Hub-and-Spoke-Navigationsmetapher.....75

Abb. 24: Intermedia *Web View*: Oberhalb des aktuellen Dokuments („Cilia OV“) befindet sich die Personal History des Benutzers, unterhalb die Link-Vorschau. Der schwarze Balken vor „Cilia Function OV“ hebt den angewählten Link hervor (nach Utting & Yankelovich 1989).....77

Abb. 25: Ein Beispiel für das „Shepardizing“ und eine Übersicht der Verweistypen (LexisNexis 2003). 79

Abb. 26: Frei typisierte Links im *NoteCards*-System (aus Trigg, Suchman et al. 1986).....81

Abb. 27: „*Embedded Menues*“ sind (im Beispiel durch Unterstreichung) hervorgehobene Phrasen eines Dokuments, die direkt ausgewählt werden können und als Link-Anker dienen (aus Koved & Shneiderman 1986).....82

Abb. 28: Typ-Hierarchie des TextNet-Systems (aus Trigg & Weiser 1986). 83

Abb. 29: Auswahl der Link-Typen in MUCH (aus Wang & Rada 1995).....84

Abb. 30: Ein Start- und ein Zielanker im HTML-Code einer Webseite.....89

Abb. 31: Illustration zur Gestaltung von Link-Markern mit CSS. Oben der CSS-Code, unten die Darstellung eines Beispiel-Dokuments. 93

Abb. 32: CSS-Code, der eine Grafik hinter dem Link-Anker einfügt, die von dem Ziel-URI des Links abhängt. 94

Abb. 33: Opera 11.5 zeigt strukturelle Links in einem Navigationsbalken an. Der Benutzer wählt gerade den Link zur Suche in der Website aus. 96

Abb. 34: Das „Link“-Element in HTML. Das Beispiel führt zur Homepage der Website. 96

Abb. 35: Der Navigationsbalken in iCab zeigt strukturelle Links als Icons an. 97

Abb. 36: Drei Links auf Text-Dokumente in unterschiedlichen Dateiformaten.....112

Abb. 37: Drei unterschiedliche Medientypen mit derselben Kurzgeschichte als Inhalt 114

Abb. 38: Ein Text mit angrenzenden und mehrzeiligen Ankern und einer hohen Anker-Dichte.....128

Abb. 39: Link-Marker für zwei überlappende Link-Anker in Hyper-G (nach Andrews 1996). 128

Abb. 40: Hyperlinks in Browser Nexus von Tim Berners-Lee.....131

Abb. 41: Text- und Hintergrundfarben zur Hervorhebung von Textbereichen. 133

Abb. 42: Die *CritSuite* nutzte kleine bunte Pfeile, um Anfang und Ende von zusätzlich eingefügten Link-Ankern anzuzeigen. Zu sehen ist die ehemalige Homepage des Projektes unter http://www.crit.org/.....134

Abb. 43: Link-Marker im *Neptune Document Viewer* (aus Delisle & Schwartz 1987). 135

Abb. 44: Link-Marker im Webbrowser UdiWWW aus dem Jahr 1996. 135

Abb. 45: Laterale Link-Marker, Schattierungen für den Link-Kontext und Markierungen im Scrollbalken für im Link verteilte Link-Anker (aus Weinreich, Obendorf et al. 2001). 136

Abb. 46: Bei Eclipse werden rechts neben dem Scrollbalken Fehler im Code als rote Kästchen hervorgehoben. 137

Abb. 47: Der seitliche schwarze Balken und der gepunktete Rahmen symbolisieren Zielanker von Links (aus Grønbæk & Trigg 1999)..... 138

Abb. 48: Hyperlinks in *NoteCards* (nach Trigg, Suchman et al. 1986)..... 144

Abb. 49: Zwei Beispiele für ergänzende Link-Texte in Webseiten. 145

Abb. 50: Link-Icons und Link-Symbole nach Evenson (aus Evenson, Rheinfrank et al. 1989, S. 90). ... 146

Abb. 51: Ein Beispiel der Darstellung von Link-Ankern mit der Browser-Erweiterung „*Traffic Lights*“. 147

Abb. 52: Die Browser Card von *NoteCards* unterschied zwischen strukturellen Links und Querverweisen (aus Halasz 1988). 148

Abb. 53: Illustration eines *Issue Networks* in gIBIS (nach Nielsen 1993a, S. 49). 149

Abb. 54: Eine Karte in SEPIA, die als „Planungsraum“ dient (nach Streitz, Haake et al. 1992). 149

Abb. 55: Der Browser Mosaic für Windows zeigte den URI der des Links im Statusbereich an. Die Abbildung zeigt die erste öffentliche Beta-Version 0.6 vom September 1993. 151

Abb. 56: Die wichtigsten Mauszeiger des Guide-Systems (vergl. Nielsen 1993a, S. 57). 151

Abb. 57: Ein Tooltip beschreibt die Funktion eines Icons in Microsoft Word XP. 153

Abb. 58: Ein Tooltip mit dem Inhalt des „title“-Attributs im Internet Explorer. 153

Abb. 59: Eine Illustration zur Kennzeichnung externer Links (aus Nielsen 1999a, S. 69). 153

Abb. 60: Ein *Smart Tag* in der Beta-Version des Microsoft Internet Explorer 6. 154

Abb. 61: Vorschläge für die visuelle Darstellung unterschiedlicher Link-Typen (nach Noirhomme-Fraiture & Serpe 1998). 158

Abb. 62: Die Java-Toolbox zeigt mithilfe von Java Applets zusätzliche Informationen zum Link-Ziel an (nach Kreutz, Conradi et al. 1998). 159

Abb. 63: Die drei Vorschau-Grafiken des *Enriched Link Frameworks* (nach Marchionini, Geisler et al. 2000). 161

Abb. 64: Erweiterung des Link-Kontextes und Einbindung von Preview-Informationen für die Links in einer Webseite (nach Harper, Goble et al. 2004). 161

Abb. 65: Mehrzeilige Tooltips mit Dateiinformationen im Explorer von Windows XP (links) und von Windows 7 (rechts). 165

Abb. 66: Screenshots der „Proof-of-Concept“-Implementation von *HyperScout* (aus Weinreich & Lamersdorf 2000). 166

Abb. 67: Schematische Darstellung der Funktionsweise des Software-Prototyps HyperScout I. 168

Abb. 68: Tooltip mit *Titel* und *Inhalt* des Link-Ziels. Der Benutzer hat es bereits viermal aufgerufen, und es befindet sich in einem *untergeordneten* Verzeichnis. 171

Abb. 69: Tooltip mit Angabe des letzten Updates. Das Zieldokument wird von Bildern und Grafiken dominiert („Grafisch“) und ist die Homepage der Website. 171

Abb. 70: Ein Link zur aktuell angezeigten Seite. 171

Abb. 71: Ein Link zur Homepage mit Hinweisen zum letzten Besuch auf der Seite und auf der Website. 172

Abb. 72: Link zum Fuß der Seite. 172

Abb. 73: Ein Sprung zum Seitenkopf. 172

Abb. 74: Das Link-Ziel wird in einem neuen Browser-Fenster dargestellt. 173

Abb. 75: Ein *Stretchtext-Link* im Webbrowser: Nach dem Anklicken des Ankers „*Inline Small*" (obere Grafik) wird mittels JavaScript ein Bereich eingefügt, in dem die Zielseite erscheint (untere Grafik). 173

Abb. 76: Eine E-Mail-Adresse als Link-Ziel. 173

Abb. 77: Ein Link zu einem aktuellen Dokument, das fast nur aus Text besteht. 175

Abb. 78: Ein Querverweis auf ein englischsprachiges Dokument. 175

Abb. 79: Ein Beispiel für einen Link auf eine PDF-Datei. 176

Abb. 80: Das Link-Ziel ist eine große Datei mit vielen Site-internen Links. Im Beispiel zeigt die Heuristik zur Bestimmung des Inhaltes Schwächen. 176

Abb. 81: Ein Verweis auf eine zentrale Landmark-Seite mit Formularfeldern. 176

Abb. 82: Der Verweis im linken Bild führt zu der „Sackgassen-Seite" ohne Titel und Links im rechten Bild. 176

Abb. 83: Das Zieldokument dieses Links wurde entfernt. Das Beispiel zeigt zusätzlich eine Beschreibung der Site, die aus dem Titel der Homepage gewonnen wurde. 178

Abb. 84: Die Site http://vsys1.informatik.uni-hamburg.de/ war noch bei Google indexiert, aber nicht mehr online verfügbar (vom August 2006, der Server wurde kurz zuvor ausrangiert). 178

Abb. 85: Ein externer Link zur Homepage eines schnell antwortenden Servers. 179

Abb. 86: Ein externer Link zu einem Server mit gegenwärtig langsamer Antwortzeit. Das Link-Ziel wird in einem neuen Fenster dargestellt und benötigt JavaScript. 179

Abb. 87: Der *CoInternet-Client* nutzt HyperScout I, um Bewertungen anderer Benutzer einzublenden. 180

Abb. 88: Illustration des Studienaufbaus mit zwei Bildschirmen für die Evaluation der Konzepte von HyperScout I. 189

Abb. 89: Zugriff ist nur für Mitglieder des W3C gestattet und erfordert ein Passwort. 194

Abb. 90: Bei externen Links wird eine Beschreibung der Site aus dem Titel der Homepage extrahiert (hier: „Regionales Rechenzentrum der Universität Hamburg"). 196

Abb. 91: Ein Link, der einen Bereich weiter unten auf der Seite referenziert. 197

Abb. 92: Ausschnitte aus dem Online-Fragebogen. 207

Abb. 93: Erwartungskonformität von Links im Web. 208

Abb. 94: Die Berücksichtigung des URI im Statusbereich des Browsers. 208

Abb. 95: Berücksichtigung der Tooltips mit Link-Titeln im Web. 209

Abb. 96: Die Grafik aus dem Online-Fragebogen zur Illustration von Link-Tooltips. 209

Abb. 97: Bewertung der Wichtigkeit von 16 Link- und Objekteigenschaften für die Navigation im Web. 211

Abb. 98: Rechts sind die Link-Overlays des zweiten HyperScout-Prototyps zu sehen. 214
Abb. 99: Beispiel der multiplen Maus-Icons mit drei Icons beim Mauszeiger. 217
Abb. 100: 15 Vorschläge für ein Icon für fehlerhafte Links im Web 218
Abb. 101: Drei Beispiele für die Maus-Icons von HyperScout II bei aktivierten Overlays. 220
Abb. 102: Details zu fehlerhaften Links erscheinen bei Bedarf im Tooltip. 221
Abb. 103: Link zu einem Download einer größeren ZIP-Datei. 221
Abb. 104: Tooltip mit Titel und Beschreibung. Die Zielseite hat sehr viele Links. 224
Abb. 105: Der Link führt zu einer bekannten Seite, die kurz zuvor überarbeitet wurde. 223
Abb. 106: Ein externer Link zu einem älteren Dokument, das in einem neuen Fenster erscheint. 223
Abb. 107: Ein Reference-Link innerhalb der Webseite. 224
Abb. 108: Die zwei Link-Overlay-Varianten im Vergleich. 228
Abb. 109: Mockup einer neuen Overlay-Technik für Grafiken und Textanker. 229
Abb. 110: Ein Agent, der die Metadaten für alle Verweise eines Dokuments ermittelt und bereitstellt. 248
Abb. 111: Ein autonomer Dienst zur Bereitstellung von Metadaten zu Webseiten. 251
Abb. 112: Jeder Webserver verfügt über eine eigene Komponente zur Bereitstellung erweiterter Link-Informationen. 254
Abb. 113: Funktionsweise des dezentralen HyperScout-Konzeptes zur Bereitstellung zusätzlicher Link-Metadaten. 258
Abb. 114: Schematische Funktionsweise des clientseitigen HyperScout-Agenten. 261
Abb. 115: Schematische Funktionsweise des serverseitigen Metadaten-Dienstes. 264
Abb. 116: Zwei Beispiele für die Formulierung der HyperScout-Metadaten in RDF. 266
Abb. 117: Ein Ausschnitt aus einem XHTML-Dokument mit erweiterter Syntax. 268
Abb. 118: Das Hypertext-Teilmodul von XHTML 1.1 (ohne Kommentare, siehe: McCarron & Ishikawa 2010). 275
Abb. 119: Die URN des Namensraums (Namespace) des HyperScout-Moduls 276
Abb. 120: Definition des Qualified-Name-Teilmoduls von HyperScout 277
Abb. 121: Definition des Content-Model-Teilmodul von HyperScout 279
Abb. 122: Die Definition des DTD-Driver-Moduls von HyperScout 280
Abb. 123: Ein HyperScout-XHTML-Dokument mit einem zusätzlichen Link-Attribut. 281
Abb. 124: Ein HyperScout-XHTML-Dokument, das Namespace-Prefixing einsetzt. 281
Abb. 125: Der URN der HyperScout-HTTP-Extension 283
Abb. 126: Definition des *Accept-Anchor-Attributes* für die HyperScout-HTTP-Erweiterung. 284
Abb. 127: Vier Beispiele für das neue HTTP-Request-Attribut *Accept-Anchor-Attributes.* 284
Abb. 128: Die Definition der zwei neuen HTTP-Response-Attribute 285
Abb. 129: Der HTTP-Request des erweiterten Web-Clients 286
Abb. 130: Die HTTP-Response des erweiterten Webservers 286
Abb. 131: Die vier Basiskomponenten von Scone 316

Abb. 132: WBIs grafische Schnittstelle für die Verarbeitung einzelner Anfragen. 318
Abb. 133: Schematischer Einsatz von MEGs innerhalb der Proxy-Komponente von Scone. 319
Abb. 134: Registrieren eines WBI-MEGs innerhalb eines Scone Plug-ins. 319
Abb. 135: Schematische Darstellung der WBI-ObjectStreams. Im Beispiel sind die Objekte DOM-Bäume, es werden daher DOM-InputStreams und DOM-OutputStreams verwendet. 320
Abb. 136: Ein schematischer Überblick über die Architektur des „*Scone Robot*“ 324
Abb. 137: Das grafische Interface zur Basis-Konfiguration des „*Scone Robot*“ 325
Abb. 138: Ein einfaches Beispiel für die Programmierung des „*Scone Robot*“ (Code-Auszug). 326
Abb. 139: Der *RobotMonitor* visualisiert die Aktivität des Crawlers und erlaubt seine Steuerung. 327
Abb. 140: Schematische Darstellung der Klassen zur Datenrepräsentation in den *NetObjects.* 329
Abb. 141: Struktur des Scone Datenbankadapters, gezeigt am Beispiel der *NetNodes* 336
Abb. 142: EER-Modell der Scone-Datenbank (Version 1.2). 337
Abb. 143: Schematische Darstellung des Scone *AccessTrackings* mit dem AccessTracking-Applet. 343
Abb. 144: Das Scone *UserHandling* wird über den Webbrowser bedient. 346
Abb. 145: Der Scone Rückspiegel zeigt Thumbnails von Seiten, die auf die aktuelle Seite verweisen. 352
Abb. 146: Beispiel für den Einsatz des Intermediary von Scone auf Serverseite. 354
Abb. 147: Ausschnitt einer einer XML-Datei zur Definition eines Testablaufes mit TEA. 356
Abb. 148: Links der Start-Bildschirm des *UserTestTools*, rechts ist der Browser zu sehen. 357
Abb. 149: Zwei Bildschirme des *Scone UserTestTools* in Aktion 357
Abb. 150: Beispiel-Ausgabe eines Tests als XML-Datei. 358
Abb. 151: Beispiel-Ausgabe eines Tests als CSV-Datei. 358
Abb. 152: Beispiel-Code für ein Scone Plug-in 360
Abb. 153: Eine Liste aller registrierten Plug-ins im Scone-Framework. 362
Abb. 154: GUI zum Aktivieren und Konfigurieren des Plug-ins „hyperscout3.Plugin“. 362
Abb. 155: Die grafische Schnittstelle zur Konfiguration der Parameter eines Plug-in. 362
Abb. 156: Ein Ausschnitt aus der Konfigurationsdatei „hyperscout3.Plugin.xml“. 362
Abb. 157: Das *SessionGraphs-System.* 365
Abb. 158: Zwei unterschiedliche Visualisierungsformen für Links im Web. 363
Abb. 159: Der *Navigation Visualizer* ist eine interaktive Analyseplattform für Benutzerpfade im Web. 364
Abb. 160: Die wichtigsten Schnittstellenkomponenten des *CoInternet*-Systems: Links ist der Client zu sehen, rechts Beispiele für bewertete Suchausgaben und Link-Previews. 366
Abb. 161: „App Tabs“ und die Übersicht der meistbesuchten Seiten in Google Chrome 14. 378
Abb. 162: Die Erweiterung *WOT* zeigt mithilfe von Icons und Tooltips die Reputation von Websites an. Der erste Link der Liste wird als sehr vertrauenswürdig bewertet, der dritte Treffer hingegen nicht. 383
Abb. 163 links: Die Erweiterung *LinkChecker* kennzeichnet den Status aller Link-Ziele durch farbige Overlays. Rechts: *Link Alert* hebt besondere Links durch Icons in einem Tooltip hervor. Das Beispiel zeigt einen Link zu einer anderen Website, der ein neues Fenster öffnet. 384

Abb. 164: Vannevar Bush, 1942 (Quelle: Library of Congress Prints & Photographs Division). A-3

Abb. 165: NLS-Bildschirmfoto mit Doug Engelbart (bei der „*Joint Computer Conference Demo*“, siehe: Engelbart & al. 1968).....A-5

Abb. 166: Ted Nelson auf der Hypertext-Konferenz 2003. .....A-6

Abb. 167: Tim Berners-Lee bei der World-Wide-Web-Konferenz 2006. .....A-7

Abb. 168: Die zwei berührungsempfindlichen Bildschirme eines Memex (aus Bush 1945b, S. 124; Nachdruck in Nyce & Kahn 1991, S. 110). B-2

Abb. 169: Links ein Arbeitsplatz des NLS-Systems, rechts Maus und Chord Keyset..... B-3

Abb. 170: Fünf Fenster auf dem Bildschirm des NLS/Augment-Systems. B-4

Abb. 171: Tags und Jumps in FRESS (nach Meyrowitz & Van Dam 1982a). B-6

Abb. 172: *KMS* Screenshot, der die unterschiedlichen Link-Typen zeigt. Die letzte Version von *ZOG* hatte eine vergleichbare Schnittstelle (aus Akscyn, McCracken et al. 1988). B-8

Abb. 173: Fünf Beispiele für Tooltips beim Mauszeiger von *KMS*. Die Texte repräsentieren die Funktionen der drei Mausknöpfe (aus Akscyn, McCracken et al. 1988)..... B-9

Abb. 174: Mockup der Transpointing Windows in Xanadu (Nelson 1972). B-10

Abb. 175: Ein Xanadu-Prototyp mit “Transpointing Windows” (aus Nelson 1999). B-11

Abb. 176: Ein Bildschirmfoto von *TextNet* (aus Trigg & Weiser 1986). B-13

Abb. 177: Ein *Inline Link* im Guide-System; rechts ist die expandierte Version des Links (aus Rada 1991, S. 48ff). B-14

Abb. 178: Das Guide-System für Windows aus dem Jahre 1996..... B-15

Abb. 179: Eine Text-Seite in der DOS-Version von HyperTies (aus Shneiderman & Kearsley 1989). B-16

Abb. 180: Die „Extra-Funktionen“ von HyperTies. B-17

Abb. 181: Ein NoteCards-Bildschirmfoto mit drei Textkarten und einer Karte vom Typ „Map“ (aus Halasz 1988)..... B-18

Abb. 182: Eine „Browser Card“ und mehrere „Fileboxes“ in NoteCards (aus Halasz 1988). B-19

Abb. 183: Ein Screenshot von Intermedia (aus Landow 2004). B-21

Abb. 184: Ein Konzept für überlappende Link-Extents in Intermedia (nach Garrett, Smith et al. 1986). B-22

Abb. 185: Knöpfe und Icons als aktive Objekte in HyperCard. B-24

Abb. 186: Sun’s Linking Command Panel (aus Pearl 1989). B-25

Abb. 187: Einbettung des Link Service in *textedit*. Links oben ist der Link-Marker zu sehen (Pearl 1989). B-25

Abb. 188: Der Windows Calendar mit Microcosms Universal Viewer. Der Benutzer folgt gerade dem Link „Y212” (nach Davis, Knight et al. 1994). B-27

Abb. 189: Die Auswahl multipler Links in Microcosm DLS..... B-28

Abb. 190: Beispielansicht der Client-Anwendungen von Harmony (nach Andrews 1996). B-30

Abb. 191: Der Zielanker eines Links in einem Postscript-Dokument (aus Andrews 1996, S. 62)..... B-31

Abb. 192: Der Browser Nexus auf dem NeXTStep-System. Zu sehen ist Version 2 aus dem Jahr 1994 (nach Cern 2008). B-33

Abb. 193: Der *Line-mode Browser* von Nicola Pellow war anfänglich der einzige verfügbare Browser für viele Benutzer des Web (Quelle: Cern 2008). B-34

Abb. 194: Ein Screenshot von ViolaWWW. Die dargestellte Version ist aus dem Frühjahr 1993. ..... B-34

Abb. 195: Eine Webseite im Browser Cello (Version 1.01 aus dem Jahr 1994). B-35

Abb. 196: NCSA Mosaic für Windows. Zu sehen ist Version 0.4 vom September 1994. ..... B-36

Abb. 197: Der *Netscape Navigator 1.0* aus dem Jahr 1994. Zu sehen ist die „Welcome Page", die beim Start des Browsers erscheint. B-37

Abb. 198: Opera bot als erster Browser Tabs an (oben) und zeigt im Tooltip neben dem Titel immer den URI des Link-Zieles (rechts). B-38

Abb. 199: Die drei Schichten und zwei Schnittstellen des Dexter-Modells. C-2

Abb. 200: Der Aufbau von Komponenten im Dexter-Modell. C-4

Abb. 201: Link-Anker im DHM. Auf der linken Seite ist eine Auswahl von Link-Zielen zu sehen (aus Grønbæk & Trigg 1999). C-7

Abb. 202: Die Illustration von Links in einem *AHM*-konformen System sowie das Folgen des Links „Walking Routes" (Hardman 1998). C-9

Abb. 203: Die Benutzungsschnittstelle des Xspect-Systems im Überblick (nach Christensen, Hansen et al. 2003). C-19

Glossar

Dieses Glossar erklärt eine Reihe von Fachbegriffen und Abkürzungen, die in dieser Arbeit besondere Relevanz haben. Für das Glossar wird kein Anspruch auf Vollständigkeit erhoben; es dient vor allem dazu, die Verständlichkeit der Arbeit zu verbessern.

API, Application Programming Interface: Beschreibt die technische Programmschnittstelle eines Software-Systems, die es zur Verfügung stellt, um eine Anbindung an andere Software-Systeme oder eine Kommunikation mit anderen Software-Systemen zu ermöglichen.

Benutzbarkeit von Software: Die Benutzbarkeit von Software wird häufig anhand von Kriterien beschrieben. Beispielsweise definiert Teil 110 der DIN ISO 9241 Normenreihe sieben Qualitätskriterien für die Dialoggestaltung benutzbarer Software: Dies sind die *Aufgabenangemessenheit*, *Selbstbeschreibungsfähigkeit*, *Steuerbarkeit*, *Erwartungskonformität*, *Fehlertoleranz*, *Individualisierbarkeit* und *Lernförderlichkeit* eines Systems (ISO9241-110 2006). Allerdings geben diese Kriterien immer nur Anhaltspunkte für die Benutzbarkeit eines Systems, da ein so allgemein gehaltener Katalog nicht die spezifischen Anforderungen von Benutzern im Kontext ihrer Aufgaben, der Organisation und der Umgebung berücksichtigen kann (s. auch Eberleh, Oberquelle et al. 1994: 385). Eine frühe Einbeziehung der Anwender bei der Entwicklung als auch die *Benutzbarkeitsevaluation* von Software-Systemen ist daher notwendig, um konkrete Benutzbarkeitsdefizite feststellen und beheben zu können (s. auch *Gebrauchstauglichkeit*).

Benutzbarkeitstest, Benutzbarkeitsevaluation, Evaluation: Verfahren, mit denen sich die Benutzbarkeit eines Systems systematisch überprüfen lässt. Die **Software-ergonomische Evaluation** versucht unter Berücksichtigung der Benutzer, ihrer Aufgaben und weiterer bedeutender Faktoren, Probleme bei der Benutzbarkeit eines Systems zu ermitteln. Auf die wichtigsten Evaluationsmethoden für Software-Systeme wird in Kapitel 6.1.2 eingegangen.

Crawler, Web-Crawler, Web-Robot: Software-System, das selbsttätig anhand von bestimmten Regeln ein verteiltes Informationssystem wie das Web durchsucht, um die vorhandenen *Ressourcen* zu erfassen. Die gewonnenen Daten können beispielsweise für den Suchindex von Suchmaschinen oder für Navigationswerkzeuge eingesetzt werden (s. Abschnitte 3.1.1 und 8.6.2.1).

DOM, Document Object Model: Die Spezifikation einer Schnittstelle, mit der sich HTML-Dokumente (Webseiten) softwaretechnisch manipulieren lassen. Das Dokument wird dabei in Form eines Baumes repräsentiert.

Ergonomie: Ist die *Wissenschaft von der Anpassung von Technik an die Benutzer*. Software-Ergonomie beschäftigt sich „disziplinübergreifend … mit der benutzergerechten Gestaltung der Mensch-Computer-Interaktion“ (nach Eberleh, Oberquelle et al. 1994: 1). Ziel der **Software-**

Ergonomie ist die Gestaltung von *menschengerechten Software-Systemen*, also von leicht *benutzbarer* und *gebrauchstauglicher* Software.

Framework, Software-Framework: Ein einfach anpassbares „Software-Skelett" für die Entwicklung von Systemen einer vordefinierten Kategorie. Ein Framework sollte ein wiederverwendbares Design aufweisen, bei dem alle Bestandteile aufeinander abgestimmt sind und bekannte Entwurfsmuster verwendet werden. Es gibt die Architektur und den Kontrollfluss der späteren Anwendungen vor (s. Abschnitt 8.5.1).

Gebrauchstauglichkeit: Eine Definition dieses Begriffs in Bezug auf Software-Systeme findet man in Teil 11 der DIN ISO 9241 Normenreihe (ISO9241-11 1999). Sie spezifiziert drei Faktoren für die Gebrauchstauglichkeit von Software: Die *Effektivität* oder der *Wirkungsgrad* beschreibt funktionale und nichtfunktionale Aspekte eines Systems wie die Verlässlichkeit und Wirtschaftlichkeit. Die *Effizienz* beschreibt den Aufwand und die nötige Anstrengung des Anwenders, um ein Ziel zu erreichen, und die *Zufriedenheit* ist die subjektive Bewertung des Systems (ISO9241-11 1999). In Ergänzung zur *Benutzbarkeit* eines Systems (s. o.) wird bei der Gebrauchstauglichkeit somit noch die Zufriedenheit des Anwenders gefordert und die Effektivität des Systems für den vorgesehen Einsatzbereich verlangt; meist werden aber dennoch beide Begriffe synonym verwendet, so auch im Rahmen dieser Arbeit. Da die Anforderungen der DIN ISO 9241 Teil 11 zu allgemein sind, um sie direkt zur Bestimmung von Benutzbarkeitsproblemen heranzuziehen, werden in den Kapitel 3.2ff und 6.1.1 dieser Arbeit die Faktoren für die Gebrauchstauglichkeit verteilter *Hypertext-Informationssysteme* konkretisiert.

HTML, XHTML: Steht für **H**yper**t**ext **M**arkup **L**anguage und ist die grundlegende Sprache, in der alle Webseiten erstellt werden (s. Abschnitt 7.4.3). Das „X" bei XHTML steht für „E**x**tensible" und spezifiziert die „XML-konforme" Variante des Standards (vergl. Anhang C.3), die sich durch Erweiterbarkeit und eine einfachere Verarbeitung durch Software-Systeme auszeichnet. Inzwischen gibt es zahlreiche ergänzende Sprachen zu HTML wie **CSS** (vergl. Abschnitt 4.4.4) für die Gestaltung (Darstellung, Layout, Farben, Schrifteigenschaften) der Webseiten und **JavaScript**, das interaktive Seiten im Webbrowser und komplexe Web-Applikationen ermöglicht (s. Abschnitt 2.2.3).

Hyperlink, Link: Bezeichnet eine *Referenz* (auch: *Verweis* oder *Verknüpfung* genannt) in einem Online-Informationssystem, die von einer *Ressource* oder dem Teil einer solchen (z. B. einer *Webseite*) zu einer anderen Ressource oder einem Teil einer anderen Ressource führt und es Benutzern erlaubt, direkt zwischen ihnen zu *navigieren*. In Kapitel 2.2.1 wird genauer auf Konzepte und Möglichkeiten von Hyperlinks eingegangen.

HyperScout: Ein Konzept zur Erweiterung der Schnittstelle von *Hyperlinks* im World Wide Web, bei dem auf Anforderung des Benutzers automatisch zusätzliche *Link-Previews* für alle Links einer Webseite zur Verfügung gestellt werden. HyperScout wurde im Rahmen dieser Arbeit konzipiert, in zwei Versionen prototypisch realisiert und evaluiert (s. Abschnitt 5.5ff).

Hypertext: Ein Konzept, das den Umgang mit Informationen vereinfacht, indem inhaltlich verwandte *Ressourcen* mithilfe von Verknüpfungen in Beziehung zueinander gesetzt werden. Hypertext ermöglicht es Anwendern, anhand der Verknüpfungen schnell von einer Ressource zu inhaltlich verwandten Informationen zu gelangen (s. Abschnitt 2.1.4). In dieser Arbeit geht es um **assoziativen Hypertext**, der explizite *Hyperlinks* verwendet, um Ressourcen zu vernetzen (s. Abschnitt 2.2).

Hypertext-Informationssystem (auch kurz **Hypertext-System** oder **Informationssystem**): Soziotechnisches System, das zum Ziel der optimalen Bereitstellung von Information und Kommunikation eingesetzt wird und mithilfe von Verknüpfungen den Zugriff auf weiterführende Informationen erleichtert (s. Kapitel 2.1.4ff). In dieser Arbeit geht es um **offene, verteilte Hypertext-Informationssysteme**, die es erlauben, beliebige Dateitypen einzubinden und die Objekte auf viele Computersysteme zu verteilen (s. Abschnitte 2.2.2.1 und 2.2.2.2).

Indexierung: Technischer Vorgang der Erstellung eines Verzeichnisses von *Ressourcen*, sodass auf die erfassten Daten schnell zentral zugegriffen werden kann. Die *Crawler* von Web-Suchsystemen nutzen diese Methode zur Erstellung ihrer Suchindexe (s. Abschnitt 3.1.1).

Intermediary: Ein Software-System, das sich zwischen Client und Server eines verteilten Informationssystems befindet und so auf den gesamten Datenstrom zwischen beiden Komponenten zugreifen kann (s. Kapitel 8.4). Intermediaries können vielfältige Funktionen übernehmen, indem sie Daten bei der Übertragung aufzeichnen, modifizieren und auch neu generieren (s. Abschnitt 8.4.2).

Knoten, Ressource: Bezeichnet die *Datenobjekte* eines *Hypertext-Systems*, die mittels *Hyperlinks* verknüpft werden können. Dies können beliebige digitale Medien, wie Texte, Grafiken und Videos sein (s. Kapitel 2.2.1). Im Zusammenhang mit dem Web wird meist von Ressourcen gesprochen, da sich alle Objekte des Webs mittels *URIs* adressieren lassen.

Link-Anker: Teil einer *Ressource*, der *Ausgangspunkt* für einen *Hyperlink* ist und der sie auf diese Weise mit einer anderen Ressource verknüpft (s. 2.2.1). Ein Link-Anker wird mittels eines *Link-Markers* gekennzeichnet.

Link-Benutzungsschnittstelle, Link-Schnittstelle, Link-Interface: Die Benutzungsschnittstelle von Links umfasst die Methode zur Hervorhebung der Link-Anker im *Hypertext*-Dokument, die Darstellung von zusätzlichen Informationen sowohl zum Link als auch zum Link-Ziel und die Art und Weise der Interaktion mit dem Link (z. B. die Benutzeraktion zur Auswahl des Links).

Link-Marker: Hervorgehobene Teile eines Dokuments, die die *Link-Anker* des Dokuments kennzeichnen und so die Auswahl der *Hyperlinks* erlauben (s. Abschnitt 2.2.1).

Link-on-Demand: Ein Konzept, bei dem *Link-Anker* erst gekennzeichnet werden, wenn der Benutzer eine bestimmte Aktion ausführt (z. B. eine Taste drückt). Links-on-Demand bieten

den Vorteil, dass sich ein Dokument auch ohne die (teilweise störend) hervorgehobenen Link-Anker lesen lässt (s. Kapitel 5.2.8).

Link-Overlay: Methode zur Kennzeichnung von *Link-Ankern* mithilfe von durchscheinenden („transluzenten“) farbigen Flächen, die über die aktiven Bereiche der Link-Anker gelegt werden (s. Abschnitt 5.2.5). Auf diese Weise lassen sich sowohl textliche als auch grafische Link-Anker gleichermaßen hervorheben.

Link-Preview (auch **Link-Hinweis**): Textliche oder grafische Darstellung von Informationen bei einem *Link-Anker*, die ergänzende Hinweise über den *Typ*, die *Aktion* und/oder das *Ziel eines Links* enthält und so dem Benutzer wichtige Hinweise für seine Navigationsentscheidung geben kann.

Multiple Maus-Icons: Kennzeichnung von bestimmten Objekteigenschaften durch ein oder mehrere Icons, die direkt beim Mauszeiger erscheinen, wenn der Zeiger über das Objekt geführt wird.

Navigation, Web-Navigation: Aktionen, die Benutzer durchführen, um sich virtuell von einer *Ressource* zu einer anderen zu bewegen und um so bestimmte Informationen zu finden oder bestimmte Ziele zu erreichen.

Prefetching: Ein technisches Verfahren, um im Hintergrund von einem *Hypertext*-Dokument aus Daten der – per *Hyperlinks* referenzierten – *Ressourcen* zu laden. Dies geschieht bereits während sich der Anwender das Ausgangsdokument ansieht. Auf diese Weise wird es möglich, das Link-Ziel schneller darzustellen oder auch *Link-Previews* zum Ziel bereitzustellen, ohne dass Anwender einem Link folgen müssen (s. Kapitel 4.1.1).

Ressource: siehe *Knoten*.

Scone: Ein Konzept für ein system- und plattformunabhängiges *Software-Framework* zur softwaretechnischen Erweiterung des Webs, das im Rahmen dieser Arbeit konzipiert und realisiert wurde. Es wurde zur prototypischen Entwicklung von Werkzeugen zur Vereinfachung der Orientierung und *Navigation* im Web ausgelegt (s. Kapitel 8).

Tooltip: Ein kleines grafisches Fensterelement, das erscheint, wenn der Mauszeiger über ein aktives Objekt einer Benutzungsschnittstelle gehalten wird und das ergänzende Informationen oder Hilfen zu dem Objekt enthält (s. Abschnitt 5.3.7).

URI (Uniform Resource Identifier): Der URI ist ein Mechanismus zur Identifikation von abstrakten oder physischen *Ressourcen*. URIs unterteilen sich in ***URNs*** und ***URLs***. Erstere bezeichnen *dauerhafte, ortsunabhängige Ressourcen*. Dies können beispielsweise digitale Dokumente oder die Elemente einer Sprache zum Datenaustausch sein. *URLs* werden hingegen primär für die (teilweise recht flüchtigen) Ressourcen im World Wide Web verwendet und bestehen aus einem Zugriffsprotokoll (z. B. HTTP), dem Ort (ein Domain-Name) und der Adresse einer Ressource. In dieser Arbeit wird URI als Oberbegriff verwendet, da sich man-

che Bezeichnungen im Web wie „mailto-Adressen“ weder dem einen noch dem anderen Unterschema eindeutig zuordnen lassen.

Web-Applikation, Web-Anwendung: Ein Dienst im Internet, der Benutzern eine bestimmte Funktionalität zur Verfügung stellt und der sich vollständig über den Webbrowser bedienen lässt. Beispiele hierfür sind webbasierte E-Mail-Systeme, Online-Shops und soziale Netzwerke.

Webseite: Eine per *URI* adressierbare *Ressource* im Internet, die im Format *HTML* oder *XHTML* vorliegt, andere Objekte einbinden kann und zumeist auf weitere Ressourcen im Web mithilfe von *Hyperlinks* verweist.

Website, Site: Eine (häufig kohärente) Sammlung von verknüpften Web-*Ressourcen*, die auf einem oder mehreren Servern im Internet angeboten werden und in der Regel über denselben Domain-Namen verfügen und demselben Anbieter zuzuordnen sind.

Anhang: Konzepte, Schnittstellen und Standards für Hypertext und Hyperlinks

Der Anhang geht detailliert auf einige grundlegende Aspekte von Hypertext ein und gibt einen Überblick über die Ideen und Ziele der geistigen Väter des Hypertext-Konzepts, richtungsweisende Hypertext-Systeme und die wichtigsten Hypertext-Standards.

Erst werden in Anhang A vier maßgebliche Pioniere der Hypertext-Forschung und ihre Konzepte und Visionen präsentiert. Ihr gemeinsames Ziel war es, die geistige Arbeit, den Wissensaustausch und die interpersonelle Kooperation zu verbessern. Ihre Schwerpunkte waren hingegen unterschiedlich, und die von ihnen erdachten bzw. realisierten Systeme wiesen entsprechend andere Merkmale auf. Ihre Ideen und Arbeiten führten letztendlich zur Entwicklung des World Wide Web.

Anhang B geht auf die Konzepte und Realisierungen von 15 wegbereitenden *Hypertext-Systemen* ein. Die Analyse legt ein besonderes Augenmerk auf die Navigationsmöglichkeiten sowie die Benutzungsschnittstelle der Hyperlinks in den Systemen. Die Untersuchung zeigt, wie verschiedenartig und fortschrittlich diese frühen Projekte bereits waren (s. auch Tabelle 5 auf Seite B-41). Einige der damaligen Erkenntnisse lassen sich auf das Web übertragen und können für seine weitere Entwicklung hilfreich sein.

Anhang C stellt drei *Hypertext-Standards* vor, die entwickelt wurden, um eine gemeinsame technische Ebene für die Entwicklung und den Datenaustausch zwischen unterschiedlichen Hypertext-Systemen zu haben. In dieser Arbeit wird insbesondere auf die Möglichkeiten und die Ansätze zur Realisierung von Hyperlinks in diesen Standards eingegangen.

Die in diesem Anhang vorgestellten Untersuchungen dienten als wissenschaftliche Grundlage für zahlreiche Kapitel und Konzepte der Arbeit, beispielsweise für die Klassifikation von Links (s. Kapitel 4.5.1.1) und Link-Aktionen (s. Kapitel 4.5.1.3) und für die Kategorisierung der Konzepte für die Darstellung von Link-Markern (s. Kapitel 5.2) und Link-Informationen (s. Kapitel 5.3). Sie waren zudem Inspiration für die in dieser Arbeit entwickelten Konzepte zur Erweiterung der Link-Schnittstelle *HyperScout II*, insbesondere für die *Link-Overlays-on-Demand* (s. Abschnitt 6.4.1) und und die *multiplen Maus-Icons* (s. Abschnitt 6.4.2).

A Die Konzepte der Hypertext-Pioniere

Seit Langem ist es ein Bestreben von Menschen, Maschinen zu entwickeln, die sie *bei der Arbeit* und beim *Denken* unterstützen. Während Ersteres inzwischen sehr erfolgreich realisiert wurde, stellte sich die technische Unterstützung geistiger Arbeit als wesentlich problematischer heraus. Ein wissenschaftlicher Ansatz hierfür findet sich in der *künstlichen Intelligenz*, die aber nach ihren Höhenflügen in den 1980er Jahren schnell ihre Grenzen er-

reichte und insgesamt die in sie gesetzten hohen Erwartungen bis heute nicht erfüllen konnte.

Ein anderer, erheblich erfolgreicherer Ansatz liegt darin, die Informationen nicht durch Maschinen verarbeiten (sie „denken“) zu lassen, sondern die Effizienz der Informationsaufnahme für den Menschen mithilfe von Maschinen zu vereinfachen und so die geistige Arbeit zu erleichtern. Dieser Abschnitt stellt die Konzepte und Visionen der vier wichtigsten Pioniere aus diesem Wissenschaftsbereich vor und beleuchtet ihre persönlichen Hintergründe. Sie sind Schlüsselfiguren für die Entwicklung einer Systemklasse, die wir heute als Hypertext-Informationssysteme bezeichnen und die als World Wide Web aus unserem täglichen Leben kaum mehr wegzudenken sind.

A.1 Vannevar Bushs Visionen zur Erweiterung des menschlichen Geistes

Als Urvater des Hypertext-Konzepts wird zumeist Vannevar Bush (1890-1974) angesehen. Der amerikanische Elektroingenieur forschte in den 1920-er Jahren am MIT auf dem Gebiet der analogen Computer. Bereits 1931 wurde unter seiner Leitung der „*Rockefeller Differential Analyzer*“ fertiggestellt, ein analoger Rechner, mit dem man effizient differenzielle mathematische Gleichungen, beispielsweise zur Berechnung von Flugbahnen, lösen konnte.

Früh in den 1930er Jahren war eine der wissenschaftlichen Hauptinteressen von Bush die Entwicklung von Strategien, um die wachsenden Mengen an Informationen handhabbar zu halten. Er kritisierte die schlechte Eignung von Bibliotheken, um an Informationen zu gelangen und vertrat die Meinung, dass andere, menschengerechtere Wege gefunden werden müssten, um dies zu vereinfachen.

> „The difficulty seems to be, not so much that we publish unduly in view of the extent and variety of present-day interests, but rather that publication has been extended far beyond our present ability to make real use of the record. The summation of human experience is being expanded at a prodigious rate, and the means we use for threading through the consequent maze to the momentarily important item is the same as was used in the days of square-rigged ships” (Bush 1945a).

Um das Jahr 1932 begann er die Idee für eine Maschine zu entwickeln, die er später *Memex* nannte (Nyce & Kahn 1991: 42). Sein Ziel war es, eine Technik zu erschaffen, die dem Menschen bei der Recherche hilft, indem sie Informationen für sie auf eine natürlichere Art und Weise zugreifbar macht. Bushs Vorbild war der menschliche Verstand und das durch Assoziationen geprägte Denken und Lernen. Er publizierte seine Konzepte aber erst nach dem Krieg im Jahre 1945, nach seiner Zeit als wissenschaftlicher Berater von Präsident Roosevelt (s. Abb. 165). In dem oft zitierten Artikel „As We May Think“ (Bush 1945a) beschreibt er den Kern seiner Vision wie folgt:

> „Consider a future device for individual use, which is a sort of mechanised private file and library. It affords (…) associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another” (Bush 1945a).



Abb. 165: Vannevar Bush, 1942 (Quelle: Library of Congress Prints & Photographs Division).

Memex sollte ein persönliches System sein, mit dem man sowohl auf Mikrofilm erworbene Informationen studieren als auch selbst Dokumente erstellen und speichern könne. Vor allem wäre es damit auch möglich, *individuelle Pfade* zwischen den Dokumenten festzulegen. Diese Pfade sollten zu inhaltlich verwandten Informationen verweisen und so dem Benutzer helfen, aktuelle Assoziationen festzuhalten und frühere Assoziationen wieder aufzugreifen. Im Gegensatz zu Gedanken und Assoziationen im Gedächtnis *verblassen* diese Pfade nicht („*trails do not faint*"), sondern blieben persistent gespeichert. Man sollte auch die Möglichkeit besitzen, solche Pfade anderen Menschen zukommen zu lassen.

Die Ansätze für Memex waren zwar visionär, aber dennoch fundiert, und basierten mit Mikrofilmen und analogen, mechanischen Techniken auf den Möglichkeiten der Zeit (Nyce & Kahn 1991: 56). Bush hielt offensichtlich eine baldige Umsetzung seiner Konzepte für realistisch. Hierzu kam es aber insbesondere nicht, weil sich die analoge Computertechnik nicht in dem Maße entwickelte, wie er es erhofft hatte.

A.2 Doug Engelbarts Konzepte zur Unterstützung der kooperativen kreativen Arbeit

Ein weiterer bedeutender Visionär bei der Entwicklung der Hypertext-Konzepte war Douglas („Doug") Carl Engelbart (geb. 1925 in Portland, Oregon). Engelbart hatte Elektrotechnik studiert und arbeitete im Krieg für das US-Militär als Radartechniker (Bardini 2000).

Ende der vierziger Jahre wandte er sich einer akademischen Laufbahn zu und studierte Ingenieurswissenschaften. Er beobachtete während dieser Zeit besorgt, dass die Informationsmenge in einem stetig wachsenden Tempo zunahm und es somit immer dringlicher wurde, das Problem des Zugriffs auf die Informationen zu lösen. Beeindruckt von Bushs Artikel, beschloss er 1951 auf der Basis digitaler Computer ein Werkzeug wie Memex zu entwickeln (Bardini 2000). Dieses Vorhaben kann als ausgesprochen ambitioniert angesehen werden – vier Jahre nach der Erfindung des Transistors, sieben Jahre *vor* dem ersten Mikrochip und fast 20 Jahre vor den ersten Computernetzen. Seine kühne Vision lag wohl auch

darin begründet, dass er sich zu der Zeit mit den entsprechenden Techniken nur peripher auskannte (Gillies & Cailliau 2001). Um seiner Vision näher zu kommen, ging er zurück zur Universität und promovierte. Er wählte die Universität von Berkeley aus, da es hier ein Computerprojekt gab – wenn auch noch keinen Computer (Gillies & Cailliau 2001). Da an der Computerfakultät keiner seine Vorstellung zukünftiger Mensch-Computer-Interaktion nachvollziehen konnte, musste er für seine Promotion ein technisches Thema wählen.

Anfang der sechziger Jahre erhielt Engelbart am *Stanford Research Institute* die Möglichkeit, seine Ideen umzusetzen, obwohl sie ihrer Zeit um Jahrzehnte voraus waren. Er griff dabei die Ideen von Vannevar Bush auf, hatte Briefverkehr mit ihm und schrieb 1963 den Artikel „A Conceptual Framework for the Augmentation of Man's Intellect", in dem er die Vision aufbaute, die Menschheit mithilfe digitaler Computer in eine neue Phase der Evolution zu führen. Computer sollten Menschen in die Lage versetzen, in kürzester Zeit Informationen und Konzepte interaktiv und kooperativ zu manipulieren (Conklin 1987). Sein Projekt sollte ursprünglich *H-LAM/T* heißen („***H**uman using **L**anguage, **A**rtifacts and **M**ethodology, in which he is **T**rained*"), erhielt dann später aber den Namen *Augment*. Der Kern des Projekts war das System *NLS* („*O**N** **L**ine **S**ystem*", s. Abschnitt B.2). *NLS* sollte eine EDV-Arbeitsumgebung bieten, die alle für die Arbeit einer Arbeitsgruppe wichtigen Dokumente, Memos und Artikel enthielt, als auch das Planen, die Kooperation und die Kommunikation unterstützte (McKnight, Dillon et al. 1991: 9). Das *Augment*-Projekt konzentrierte sich dabei auf die gemeinsame Erstellung strukturierter Informationen (Engelbart 1984).

Da für Engelbart und seine Gruppe die Benutzbarkeit ihrer Systeme ein ganz besonderes Anliegen war, entwickelten sie eine Reihe von speziellen Ein- und Ausgabegeräten, die die Interaktion mit *NLS* optimieren sollten. Die Bildschirme der Terminals waren flimmerfreie, hoch auflösende grafische CRTs mit Zeilendarstellung. Da dies aufgrund der zu langsamen Technik eigentlich noch gar nicht realisierbar war, verwendeten sie einen Trick: Die Bildschirme der Anwender erhielten ihre Daten direkt von Videokameras, die andere CRTs mit Vektorgrafik aufzeichneten. So konnten neben detaillierten, flimmerfreien Computerausgaben auch die Kooperationspartner dargestellt werden. Bush und sein Team entwickelten zudem zwei neue Eingabegeräte: das *Chord Key Set* und die *Maus* (Engelbart 1984).

NLS wurde erstmals im Dezember 1968 einem größeren Forscherkreis auf der *Joint Computer Conference* in San Francisco vorgestellt (Engelbart & al. 1968). Diese Vorführung kann für die damalige Zeit in Inhalt und Ausführung als bahnbrechend angesehen werden und wurde dafür als „*Mother of all Demos*" bezeichnet (Gillies & Cailliau 2001). In einer Halle bekamen ca. 3000 Zuschauer das System live vorgeführt, wobei sie dank Videoprojektoren Engelbart und die Bildschirmausgaben sehen konnten, der mittels eines Head-Sets seine Aktionen kommentierte (s. Abb. 166).

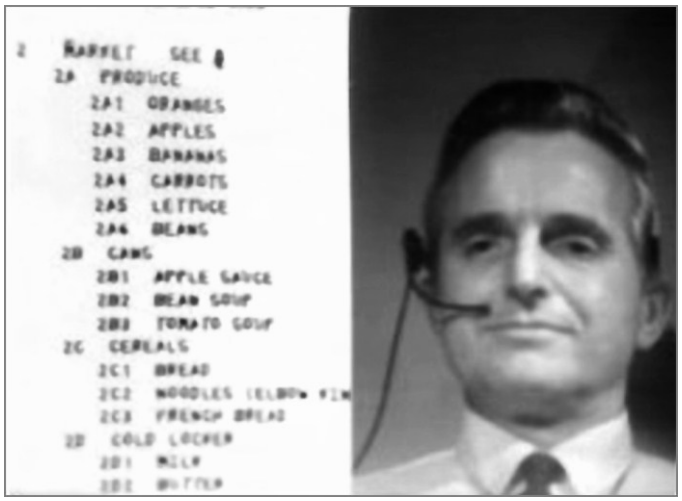


Abb. 166: NLS-Bildschirmfoto mit Doug Engelbart (bei der *„Joint Computer Conference Demo“*, siehe: Engelbart & al. 1968).

Obwohl die Demonstration offensichtlich viele Personen begeisterte und Engelbart große Bekanntheit einbrachte, blieb die erhoffte Flut von Angeboten zur Weiterentwicklung des Systems aus (Gillies & Cailliau 2001). Es war aus heutiger Sicht seiner Zeit wohl zu sehr voraus: Computer waren noch zu teuer und komplex, als dass Firmen tatsächlich so etwas wie persönliche Computer zur Unterstützung geistiger Arbeit als realistisch ansehen konnten, um hier ein mögliches Geschäft zu wittern.

Das *NLS/Augment*-System blieb über 20 Jahre in Betrieb. Kurz vor seiner Abschaltung hat Engelbart es 1987 noch auf der ersten Hypertext-Konferenz vorgeführt. Bis zu diesem Zeitpunkt hatte es ca. 2000 Benutzer gehabt, und seine Datenbank speicherte über 100.000 Dokumente. Im Endeffekt „starb“ das Projekt an fehlenden Förderungsgeldern und seinen für die damalige Zeit zu innovativen und damit zu teuren Konzepten (Nielsen 1988b).

A.3 Ted Nelsons Vision eines revolutionären globalen Hypertext-Systems für alle

Theodor („Ted“) Holm Nelson (geb. 1937 in den USA) ist ein bis heute aktiver Visionär und „Guru“ der Computerwelt, der sich mit seinen oft weit vorausschauenden Visionen und seiner messerscharfen Kritik an den gegenwärtigen Computersystemen einen Namen gemacht hat (s. Abb. 167).

Nelson stammt aus einer gut situierten Künstlerfamilie und studierte erst Philosophie und später Soziologie in Harvard. Ebenso wie Engelbart, verfügte er anfänglich über keine Computerkenntnisse, war aber begeistert von der Idee der „Allzweck-Maschine“. Auch er hatte den Artikel von Vannevar Bush gelesen. Während seiner Zeit an der Harvard University reifte in ihm die Vision, ein Werkzeug zu entwickeln, mit dem er seine Notizen nicht nur katalogisieren konnte, sondern das zudem alle seine Modifikationen festhielt und es ihm so erlaubte, Änderungen rückgängig zu machen und die Versionen zu vergleichen.



Abb. 167: Ted Nelson auf der Hypertext-Konferenz 2003.

Seine erste Publikation zu dieser Vision stammt aus dem Jahre 1965. Er schuf für diese Art von Informationssystemen den Begriff *Hypertext*, sein Projekt nannte er etwas später *Xanadu* (s. auch Abschnitt B.5). *Xanadu* sollte ein „allumfassendes" Informationssystem sein, in dem sämtliche jemals publizierten Werke aufgenommen werden, und so für alle Menschen einfach und preiswert zugreifbar sind (Nelson 1972). Er wollte unter dem Motto „Computerbefreiung" Menschen aus allen Schichten und Fachgebieten die Möglichkeit geben, Informationen weltweit zu suchen und zu publizieren (Nelson 1974).

Die Konzepte von *Xanadu* sahen vor, dass nie etwas gelöscht würde und sämtliche Änderungen nachvollziehbar seien. Zudem sollte man mittels sogenannter „*Transclusions*" (s. Abschnitt B.5) die Möglichkeit erhalten, Teile anderer Dokumente in eigene Publikationen einzufügen, wobei für alle Teildokumente immer ein Verweis auf den ursprünglichen Urheber erhalten bliebe. Nelson sah die Wahrung der Urheberrechte als elementar an: Ein Mechanismus mit dem Namen *Royalties* diente dazu, die Urheber für die Nutzung ihres geistigen Eigentums automatisch zu entlohnen (Nelson 1974). Nicht zuletzt sollte *Xanadu* natürlich im Sinne von Bush „Trails" auch mittels Hyperlinks beliebige Werke miteinander verknüpfen können.

Nelson hat bis in die 1990er Jahre mehrfach angekündigt, dass *Xanadu* demnächst vollendet und einsatzbereit sei. Obwohl dies nie eingetroffen ist und lediglich einige Prototypen unter Unix existieren, haben zweifellos seine oft publizierten Visionen vielen anderen Menschen als Inspiration gedient (Gillies & Cailliau 2001).

A.4 Tim Berners-Lee: Globaler, plattformunabhängiger Hypertext zum Wissensaustausch

Timothy („Tim") John Berners-Lee (geb. 1955 in London, s. Abb. 168) unterscheidet sich von den anderen Vätern des Hypertextes in der Hinsicht, dass er weniger als Visionär denn als

Praktiker und brillanter „Hacker“[1] beschrieben wird (Gillies & Cailliau 2001). Er ist der Vater des *World Wide Web*, dem bisher mit Abstand umfangreichsten und erfolgreichsten Hypertext-System, wenn auch gleichzeitig viele andere Menschen für die Erfolgsgeschichte des Web mitverantwortlich sind (Berners-Lee & Fischetti 2000).



Abb. 168: Tim Berners-Lee bei der World-Wide-Web-Konferenz 2006.

Die ersten konzeptionellen Überlegungen zum World Wide Web stellte Tim Berners-Lee bereits 1980 bei seinem ersten sechsmonatigen Aufenthalt am CERN[2] an (Berners-Lee & Fischetti 2000). Ihm fiel auf, dass das CERN eine ausgesprochen heterogene Computerausstattung aufwies, weil die vielen speziellen Anforderungen der Physiker oft auch spezielle Hard- und Software voraussetzten. Da die Systeme zumeist über andere Dateisysteme, Datenformate und sogar Zeichensätze verfügten und dies eine zeitaufwendige Konvertierung der Daten erforderte, wurde der Datenaustausch zwischen den einzelnen Gruppen und Personen am CERN wesentlich erschwert. Vor diesem Hintergrund entwickelte Berners-Lee 1980 ein kleines Hypertext-System, das ihm einen Überblick über die Forscher, ihre Projekte und die verwendeten Computer gab (Berners-Lee 2010).

1984 kehrte er zum CERN zurück und begann recht bald mit der Entwicklung eines neuen Programms zur Unterstützung des systemunabhängigen Informationsaustausches zwischen den Forschern. Sein System sollte als *Kommunikationsmedium* dienen und zu einer *effizienterer Teamarbeit* zwischen Wissenschaftlern führen. Dazu sollte jeder Benutzer auf seinem eigenen Computer schnell und einfach alle Dokumente des CERN lesen, bearbeiten und abspeichern können (Berners-Lee 1996).

Sein Projekt hatte daher folgende Anforderungen zu erfüllen:

[1] Es heißt, dass ihm während seines Physikstudiums am Queen's College in Oxford zeitweise der Computerzugang verwehrt wurde, da man ihn und einen Freund beim Hacken eines Systems erwischte (Lawson 2003).

[2] Das CERN ist die europäische Forschungseinrichtung für Kernforschung in der Nähe von Genf, die für ihre großen Teilchenbeschleuniger bekannt ist. Die Abkürzung CERN leitet sich vom französischen Namen *Conseil Européen pour la Recherche Nucléaire* ab.

- *Plattformunabhängigkeit*: Das System musste auf beliebigen Plattformen lauffähig sein und den Datenaustausch zwischen verschiedenen Systemen zulassen.
- *Verteilter Zugriff*: Der problemlose entfernte Zugriff auf alle Dokumente des Systems sollte gewährleistet werden, egal auf welchem Server (im Internet) sie gespeichert sind.
- *Hypertext*: Benutzer sollten zu eigenen als auch fremden Dokumenten Hyperlinks hinzufügen können, sodass sie sich mit assoziativen Pfaden im Sinne von Vannevar Bush verknüpfen lassen.

Tim Berners-Lee nannte sein Projekt ursprünglich „*Mesh*“. Er stellte im März 1989 hierfür seinen ersten Forschungsantrag mit dem Titel „*Information Management: A Proposal*“ (Berners-Lee 1989). Das Projekt wurde aber offensichtlich von den Verantwortlichen am CERN nicht als unterstützenswert angesehen; er erhielt noch nicht einmal eine Antwort auf seinen Antrag (Rademacher 2001). Sein direkter Vorgesetzter Mike Sendall gab ihm aber zu verstehen, dass er an dem Projekt weiterarbeiten sollte. Ihm wurde eine NeXT-Workstation zur Verfügung gestellt, und darauf entwickelte er von September bis Weihnachten 1989 den ersten Webbrowser und Server. Den Browser nannte er zuerst *WorldWideWeb*, später aber *Nexus*, als er 1990 dem Gesamtprojekt den Namen „World Wide Web“ gab (Berners-Lee 1998b).

Nexus verfügte über eine grafische Benutzungsschnittstelle und hatte einen WYSIWYG-Editor, mit dem es möglich war, Dokumente direkt zu editieren sowie neue Hyperlinks zu erstellen. Das Betriebssystem *NeXTStep* eignete sich für Berners-Lees Implementation besonders gut, da die objektorientierte grafische Benutzungsoberfläche ein editierbares „Textobjekt“ bot, auf dessen Basis sein Prototyp einfach zu realisieren war. Nexus kannte zu diesem Zeitpunkt bereits URIs, die Daten wurden per HTTP übertragen und das Dokumentformat hieß HTML (s. Abb. 193 auf Seite B-33).

Zu dieser Zeit begann sich auch *Robert Cailliau*, ein Arbeitskollege von Berners-Lee, für das Projekt zu interessieren. Er übernahm in der Folge große Teile der Planung und Organisation. Im November 1990 reichten sie gemeinsam einen weiteren Antrag für ihr World-Wide-Web-Projekt ein. Sie benötigten Mittel für zusätzliche Entwickler, um das System von der Nischenlösung *NeXTStep* auf andere Plattformen – vor allem *X Windows* für Unix – zu übertragen. Da auch dieser zweite Antrag abgelehnt wurde, konnten sie niemanden für die Portierung von Nexus einstellen. Berners-Lee beschloss daraufhin, die Bibliothek „*libwww*“ mit grundlegenden Funktionen für das Web in der Sprache C zu entwickeln und der Open-Source-Community zur Verfügung zu stellen, sodass hiermit andere einen X-Windows-Browser erstellen konnten.

Ende 1990 kam die Mathematikstudentin *Nicola Pellow* als Praktikantin zum CERN und schrieb basierend auf „*libwww*“ einen zeilenorientierten Webbrowser, der erstmals über Telnet den Zugriff von fast jedem Rechner auf das Web erlaubte.

Zu dieser Zeit reichte Tim Berners-Lee einen Bericht über sein Projekt zur Konferenz *Hypertext'91* in San Antonio (Texas, USA) ein. Der Beitrag wurde aber von den Mitgliedern des Programmkomitees abgelehnt, weil er ihrer Ansicht nach andere wissenschaftliche Arbeiten nicht ausreichend berücksichtige und das Projekt nicht den architektonischen Prinzipien offener Hypertext-Systeme entspräche (s. Abschnitt B.15), zumal es ein eigenes Dokumentenformat erforderte. Er stellte sein Projekt daher auf der Konferenz als System-Demo vor (Berners-Lee & Fischetti 2000: 39). Offensichtlich wurden die Stärken des Systems damals nicht erkannt, die vor allem in den bis dato unerreichten Möglichkeiten zur Verteilung und Skalierung bestanden.

Trotz der zahlreichen Rückschläge erlangte das World-Wide-Web-Projekt von nun an zusehends an Popularität, und die Open-Source-Gemeinde entwickelte in den Jahren 1992 und 1993 zahlreiche Browser, beispielsweise *Erwise*, *ViolaWWW*, *Cello* und *Lynx* (s. Abschnitt B.15).

Den eigentlichen Durchbruch erlangte das World Wide Web aber erst durch den Browser Mosaic, der bis heute prägend für die Benutzungsschnittstelle von Webbrowsern ist (s. Abschnitt B.15). Die erste Version von Mosaic wurde Anfang 1993 von einer Gruppe um *Marc Andreessen* freigegeben, die am *NCSA*, dem *National Center for Supercomputing Applications* der Universität von Illinois, arbeitete. Mosaic konnte als erster Browser Grafiken innerhalb von Webseiten darstellen und verlinken (Hendler, Berners-Lee et al. 2002), wodurch das Web wesentlich an Attraktivität gewann.

Mit der Popularität des Web stieg auch der Traffic, den das Web-Protokoll HTTP ausmachte, stetig an. Im März 1993 belegte HTTP nur 0,1% des NFS-Internet-Backbone, bis Oktober 1993 hatte sich der Anteil bereits verfünfhundertfacht. Ab Ende 1993 nahm die Zahl der Veröffentlichungen in allen möglichen Medien zum World Wide Web und zu Mosaic sprunghaft zu. Dies war auch die Zeit, in der die Wirtschaft zusehends Interesse an dem Projekt zeigte (Berners-Lee & Fischetti 2000).

1994 wurde das „Jahr des Web". Es gab die ersten beiden World-Wide-Web-Konferenzen, und bis zum Ende des Jahres stieg die geschätzte Anzahl der Server auf 10.000 (Gray 1996) und die der Benutzer auf über 10 Millionen (Salamon 1998). Im September 1994 bekam Tim Berners-Lee vom MIT das Angebot, das neu gegründete *World Wide Web Consortium (W3C)* zu führen, einer internationalen Organisation, die die Standards des World Wide Web definiert. Er folgte dem Angebot und ging nach Boston, wo er bis heute das W3C leitet (Berners-Lee 1998b).

Auch in den Folgejahren wuchsen das Web und seine Benutzerzahl in ähnlichem Tempo weiter. Seine Entwicklung wurde nun zunehmend durch finanzstarke Firmen, die dominierenden Browser-Hersteller – anfangs *Netscape Communications* und später *Microsoft* – und das *W3C* bestimmt. Aber immer wieder hatten auch scheinbar „aus dem Nichts" kommende

New-Economy-Firmen wie *Yahoo!* und *Google* nachhaltigen Einfluss auf das Erscheinungsbild des Web.

Entscheidend für den Erfolg des Web waren rückblickend einerseits die einfachen, praxisnahen und effizienten Konzepte von Tim Berners-Lee sowie andererseits sein für die damalige Zeit zukunftweisender Glaube an Open Source Software mit dem Verzicht auf Patente und Lizenzgebühren. So wurde das Web zu einem System, an dem nicht nur viele Menschen mitentwickeln konnten; es gab ihnen auch die Möglichkeit, auf einfache Weise zum Autor global zugreifbarer Informationen zu werden – eine Faszination, die bis heute geblieben ist.

A.5 Vier Visionen – ein Konzept

Vannevar Bush war der Wissenschaftler, der die grundlegenden Ideen für Hypertext-Systeme lieferte, Doug Engelbart war der Erste, der ein funktionierendes System baute und Ted Nelson erfand den Namen Hypertext. Er publizierte auch als Erster die Vision eines global vernetzten Informationssystems für alle; aber die technischen Konzepte, die dies letztendlich ermöglichten, erarbeitete Tim Berners-Lee. Allen vier Hypertext-Vätern war dabei eine zentrale Motivation gemein: Der Umgang mit und der Austausch von Informationen sollte für Menschen ergonomischer und effizienter werden. Dazu sahen sie jeweils vor, dass sowohl *Autoren* als auch *Leser* in ihren Systemen *assoziative Verknüpfungen* zwischen beliebigen Dokumenten erstellen konnten.

Genauer betrachtet hatten die vier Pioniere jeweils einen etwas anderen Fokus: Vannevar Bush wollte *den menschlichen Geist erweitern*, um den Zugang zu Informationen zu vereinfachen, Douglas Engelbart beabsichtigte, die *Kollaboration* innerhalb von Arbeitsgruppen zu unterstützen, und Ted Nelson strebte ein *weltweites Archiv aller verfügbaren Publikationen* zum einfachen Zugriff für alle Menschen an. Tim Berners-Lee wollte ebenfalls ein globales System erschaffen, aber er hatte primär eine Vereinfachung des *Wissensaustausches* und der *Kooperation zwischen Wissenschaftlern* im Sinn. Dabei setzte er auf offene Standards und vermied bewusst Lizenzrechte für seine Konzepte.

Eine weitere Gemeinsamkeit zeichnet die vier Hypertext-Väter aus: Obwohl ihre Ideen und Arbeiten letztendlich zu einer technischen Revolution am Ende des Zwanzigsten Jahrhunderts führten, ist keiner von ihnen wirklich reich oder berühmt damit geworden.

Betrachtet man das gegenwärtige World Wide Web, so sind viele der Visionen der Hypertext-Pioniere wahr geworden. Der Zugang zu Informationen sowie der Datenaustausch ist für seine Nutzer schnell und einfach geworden. Allerdings unterscheidet sich das Web in einigen Punkten von den ursprünglichen Vorstellungen: Assoziative Querverweise, die zu inhaltlich verwandten Informationen führen, sind im Inhaltsbereich der meisten Seiten eher rar (s. Abschnitt 4.5.1.1), und die Benutzer sind bis heute nicht in der Lage, Links oder Annotationen zu beliebigen Dokumenten hinzuzufügen (s. Abschnitte 2.2.2.2 und 2.2.3).

B Benutzungsschnittstellen von Hyperlinks in assoziativen Hypertext-Systemen

Ebenso wie die vier Pioniere der Hypertext-Entwicklung unterschiedliche Motivationen für ihre Arbeiten hatten (s. Abschnitt A), gab es zahlreiche Hypertext-Systeme, die die Idee zur Unterstützung der geistigen Arbeit unterschiedlich realisierten und eigene Schwerpunkte setzten. Dies bezieht sich nicht nur auf die technischen Möglichkeiten, sondern auch auf die Benutzungsschnittstelle der Systeme und die Gestaltung und die Funktionalität der Hyperlinks.

Viele der Konzepte und Ideen früherer Hypertext-Systeme sind auch heute noch interessant, da sie andere Ansätze wählten und in ihren Möglichkeiten bereits über die des Web hinausgingen. Beispielsweise waren in *NLS* alle Dokumente hierarchisch strukturiert, *HyperTies* bot eine eingebaute Suchfunktion, und bei Systemen wie *Intermedia* und *Hyper-G* gab es keine inkonsistenten Links. Die Vielfalt der Möglichkeiten zeigt sich auch bezüglich der Visualisierung von Link-Ankern, der Darstellung von Link-Informationen und der Interaktion mit Links. Die unterschiedlichen Systemen haben nicht nur in Form umfangreicher Erfahrungen zum wissenschaftlichen Forschritt beigetragen, vielfach wurde die Benutzung der Systeme auch systematisch untersucht. Aus den damaligen Arbeiten kann man daher bis heute wichtige Erkenntnisse ziehen.

Dieser Teil des Anhangs präsentiert eine Analyse der bedeutendsten Hypertext-Systeme. Es werden nicht nur die jeweils charakteristischen Innovationen betrachtet, sondern es wird besonderes Augenmerk auf die Realisierung und die Möglichkeiten der Benutzungsschnittstelle der Hyperlinks in den Systemen gelegt. Die damals entwickelten Konzepte waren Grundlage für die Klassifikation von Hyperlink-Schnittstellen in Kapitel 4.4 und 4.5 und für die Konzeption der erweiterten Link-Schnittstelle für das Web mit dem Namen *HyperScout* (s. Kapitel 5.5ff).

B.1 Memex: Auf assoziativen Pfaden durch die persönliche Bibliothek

Vannevar Bushs *Memex* ist der *virtuelle* Vorläufer aller Hypertext-Systeme. Memex sollte als umfassende persönliche Bibliothek dienen und zudem Platz für persönliche Anmerkungen, Fotos und Entwürfe bieten (vergl. Abschnitt A.1). Memex wurde zwar nie realisiert, Bush beschreibt aber diverse Aspekte der Benutzungsschnittstelle von Memex und schildert dabei auch, wie er sich das Anlegen und Folgen von Verweisungen vorstellte.

Bushs Systementwurf verfügte über zwei berührungssensitive Bildschirme. Die Eingabe von Daten erfolgte mittels dieser „Touchscreens“ und einer nicht genauer spezifizierten Tastatur sowie Knöpfen und Hebeln. Das wesentliche Charakteristikum von Memex waren die sogenannten „*Trails*“: Sie sollten Pfade zwischen inhaltlich zusammengehörigen Dokumenten

darstellen und helfen, beim Lesen auftretende Assoziationen später nachvollziehen zu können. Diese Verweise sollten mittels eines „einfach zu handhabenden", aber nicht genauer spezifizierten Mechanismus erstellt werden. Ein Verweis bezieht sich bei der Erstellung auf die beiden gerade dargestellten Dokumente, kann aber auch mehrere Dokumente umfassen. Trails waren als *benannte* Verknüpfungen zwischen beliebigen Dokumenten der Bibliothek konzipiert; nach heutigem Verständnis war demnach eine freie Typisierung (s. Abschnitt 4.2.2) der Pfade vorgesehen (s. Tabelle 5).

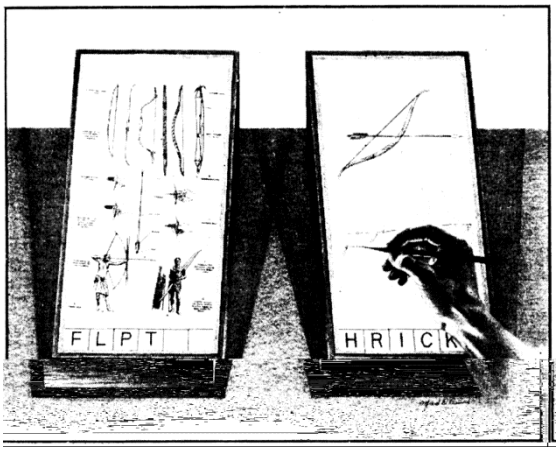


Abb. 169: Die zwei berührungsempfindlichen Bildschirme eines Memex (aus Bush 1945b: 124; Nachdruck in Nyce & Kahn 1991: 110).

Die Darstellung der Pfade (bzw. Link-Anker) wird nur vage beschrieben: Ein *Code-Bereich* am unteren Bildschirmrand sollte dem Benutzer die Auswahl eines Pfades zu dem aktuell dargestellten Dokument erlauben. Abb. 169 zeigt einen Systementwurf, wobei die Code-Bereiche durch die Zeichenfolgen „`FLPT`" und „`HRICK`" symbolisiert werden.

Genau betrachtet sind Bushs Pfade somit keine einfachen „Goto-Verweise", wie man sie heute im Web kennt, sondern sie sind eher mit *Rundgängen* durch inhaltlich zusammengehörige Dokumente vergleichbar (s. Abschnitt 3.1.5).

Die Pfade wurden in einem *Code Book* gespeichert, aus dem sie später wieder abgerufen werden konnten. Benutzer sollten die Möglichkeit haben, die eigenen Pfade auf ein externes Medium zu übertragen, damit sie sie anderen Personen zukommen lassen könnten. Das Erstellen von Pfaden sah Bush als derart bedeutsam für die weitere Entwicklung des Lesens und Lernens an, dass nach seiner Vorstellung Menschen dies (als sogenannte „*Trailblazer*"[1]) sogar als Hauptberuf ausüben sollten (Bush 1945a).

Bush wusste, dass es für eine Realisierung seines Memex noch diverser technologischer Durchbrüche bedurfte. Er hatte – mangels existierender digitaler Computertechnik – damals die Vorstellung, ein solches System mithilfe von Mikrofilmen und analogen, mechanischen und optischen Finessen zu realisieren.

[1] Er fasst das Erstellen von Pfaden mit dem Begriff „*trailblazing*" zusammen, der sich wohl am besten als das *pionierhafte Finden von Pfaden* übersetzen lässt.

B.2 NLS / Augment: Kollaborativer Hypertext für den Knowledge Worker

Eines der ersten tatsächlich realisierten Hypertext-Systeme war *NLS*, ein Akronym für *oN Line System*. Es wurde unter der Leitung von Douglas Engelbart (s. Abschnitt A.2) ab 1962 am *Augmented Human Intellect Research Center* des *Stanford Research Institute* entwickelt.[2]



Abb. 170: Links ein Arbeitsplatz des NLS-Systems, rechts Maus und Chord Keyset.[3]

NLS diente der kollaborativen Erstellung von strukturierten Dokumenten. Die Forschungsgruppe von Engelbart nutzte es beispielsweise für die gemeinsame Erstellung von wissenschaftlichen Berichten (Van Dam 1971: 110). Es erlaubte nicht nur, synchron an einem Dokument zu arbeiten, sondern dabei sogar gleichzeitig mit den Mitarbeitern *audiovisuell* zu kommunizieren. Um die Arbeit mit dem System zu vereinfachen und zu beschleunigen, bot *NLS* neben der Tastatur zwei speziell entwickelte Eingabegeräte: die *Maus* als Zeigegerät, mit der man auch Links auswählen konnte, und das *Chord Keyset*, welches die schnelle Eingabe von Daten und Kommandos mit einer Hand ermöglichte (s. Abb. 170).

Dokumente wurden in der Datenbasis von NLS gespeichert und waren mittels eines Codes eindeutig identifizierbar. Jedes Dokument war streng hierarchisch in *Statements* aufgeteilt, die jeweils bis zu 2000 Zeichen Text enthalten konnten (Meyrowitz & Van Dam 1982b), und jedes Statement hatte einen *Identifier*, der seine direkte Adressierung zuließ. Diese Identifier waren einfache Zahlen-Buchstaben-Kombinationen der Form „1“, „1a“, „1a1“, „1a1a“ für Statements aus vier Hierarchiestufen, bzw. „1b1“, „1b2“, „1b3“, „1b4“ für sequenziell auf-

[2] *NLS* war ein technisch sehr aufwendiges Projekt, dessen Entwicklung zahlreichen Verzögerungen unterworfen war. Es lief auf einem der ersten time-sharing Computer, der *SDS 940*, sowie weiterer speziell für das Projekt konstruierter Hardware.

[3] Die Grafiken sind der Website des „*Doug Engelbart Institute*“ entnommen, die die Leistungen von Douglas Engelbart zusammenfasst und öffentlich zugänglich macht. Siehe: http://dougengelbart.org/.

einanderfolgende Statements. Mittels dieser Adressierung konnten beliebige Verknüpfungen zwischen Dokumenten und Statements erzeugt werden (s. Tabelle 5).

Diese Art der Adressierung barg allerdings das Problem, dass ein späteres Bearbeiten eines Dokuments zur Änderung von Statement-Identifiern führen konnte. Als Folge konnten Links zur falschen Stelle im Dokument verweisen – ein „Vorgänger" fehlerhafter Links im Web. Aus diesem Grunde wurde später zusätzlich eine Adressierung von Statements mittels *Text und Inhalt* eingeführt (z. B. das Auftreten eines bestimmten Wortes in einem Statement als Zielkriterium), sowie *relative Adressen* definiert (beispielsweise „nächste Sektion" oder „übergeordnete Sektion", siehe: Engelbart 1984).

Eindrucksvoll ist zudem, dass NLS bereits Bildschirmbereiche für die gleichzeitige Anzeige mehrerer Statements bot, eine Technik, die mit den *Fenstern* heutiger grafischer Benutzungsschnittstellen vergleichbar ist. Abb. 171 zeigt ein Bildschirmfoto von Augment – wie das Systems später umgetauft wurde – das auf dem Terminal Emulator eines IBM PCs läuft. Auf dem Bildschirm sind gerade fünf Fenster zu sehen: links Fenster eins und zwei, rechts Fenster drei bis fünf.[4]

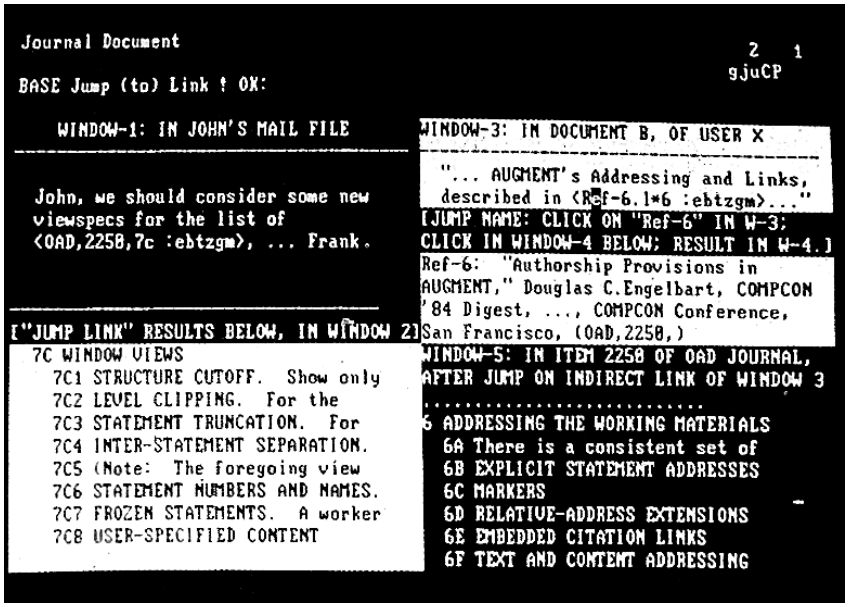


Abb. 171: Fünf Fenster auf dem Bildschirm des NLS/Augment-Systems.

Im ersten Fenster links oben („WINDOW-1") ist ein charakteristischer Link in NLS sichtbar. Er wird mittels spitzer Klammern ausgezeichnet und beinhaltet Daten über das *Link-Ziel* sowie eine *Link-Aktion* (vergl. Abschnitt 4.5.1.3). Das gezeigte Beispiel `<OAD,2250,7c:ebtzgm>` verweist zu Abschnitt *7c* im Dokument *2250* des Journals *OAD*. Der folgende Code „`ebtzgm`" ist die *ViewSpec*, welche die Funktionsweise des Links genauer identifiziert und definiert, wie das Link-Ziel dargestellt werden soll. So kann beispielsweise das komplette Statement,

[4] Eine Gruppe um Douglas Engelbart und Brad Neuberg entwickelte eine Adaption von NLS für das Web namens *HyperScope*. Dabei wurde versucht, das Verhalten des *NLS*-Systems mithilfe von JavaScript und Ajax im Webbrowser nachzuahmen. Siehe: http://www.dougengelbart.org/about/hyperscope.html.

lediglich die ersten drei Zeilen oder die Struktur des Abschnitts und seiner Unterabschnitte angezeigt werden. Im Beispiel ist eine hierarchische Ansicht der Gliederung von Sektion 7c vorgesehen, die bereits im Fenster zwei (unten links) zu sehen ist (Conklin 1987). Eine weitere Besonderheit sind die *unsichtbaren Links*[5] in *NLS*, die erst nach dem Drücken einer Tastenkombination auf dem Chord Keyset dargestellt wurden. (Engelbart & al. 1968). Sie waren damit die erste Form von *Links-On-Demand* in einem Hypertext-System (s. 6.4.1).

Die Hyperlinks in *NLS* boten somit viele Möglichkeiten und gagen dem erfahrenen Benutzer bereits im Link-Anker Informationen über das Link-Ziel und die Art der Darstellungsweise des Zielobjekts, wobei die Präsentation dieser Informationen bezüglich ihrer Benutzbarkeit für Anfänger sicherlich als problematisch anzusehen ist.

B.3 HES / FRESS: Textverarbeitung mit bidirektionalen Links und Annotationen

Ab Ende 1967 entwickelten Andries („Andy“) van Dam und einige Studenten der Brown University[6] in Kooperation mit Ted Nelson (s. Anhang A.3) das „*HES*“ („*Hypertext Editing System*“), welches mit seinem Namen entscheidend zur Verbreitung des Begriffes *Hypertext* beitrug. *HES* war als *Textverarbeitungssystem* konzipiert, das zusätzlich die Erstellung von Verweisen zwischen den Dokumenten erlaubte. Das ursprüngliche Anwendungsziel von *HES* war die Entwicklung von Lehrmaterialien für Literaturkurse: In der Lehre war es wegen seiner komplizierten Bedienung allerdings nicht besonders erfolgreich (Conklin 1987; Van Dam 1988). Der Brown University gelang es aber, *HES* an das *Houston Manned Spacecraft Center* – besser bekannt als das Apollo-Team der NASA – zu verkaufen, wo es der Erstellung von Dokumentationen für die Astronauten diente. Es heißt allerdings, dass das Projekt Ted Nelson enttäuschte und er sich neuen Zielen zuwendete (Nielsen 1993a).

Bereits Anfang 1969 begann die Gruppe von Andy van Dam mit der Entwicklung des Nachfolgers „*FRESS*“ („*File Retrieval and Editing Systems*“). Auslöser für *FRESS* war die *NLS-Demonstration* von Douglas Engelbart (s. Abschnitt A.2), die Andy van Dam begeisterte. Nachdem er mit Engelbart über sein Projekt diskutiert hatte, versuchte er viele der Konzepte von *NLS* auch in *FRESS* zu realisieren, wie beispielsweise die Mehrbenutzer- und Kollaborationsunterstützung (Van Dam 1988).

FRESS war in vielerlei Hinsicht innovativ: Benutzer konnten die Dokumente in einem Repository komfortabel speichern und auf sie zugreifen. Zur Bearbeitung stand ein Editor mit automatischem Zeilenumbruch und Undo-Funktionalität zur Verfügung (DeRose 1998), wodurch FRESS auch das erste Textverarbeitungssystem mit derartigen Möglichkeiten war

[5] Dies kann bereits als die erste Realisierung von „Links-On-Demand“ angesehen werden. Anders als beim HyperScout-Konzept werden hierbei aber *zusätzliche* Links eingeblendet.

[6] Die Brown University ist eine der ältesten Universitäten Amerikas und liegt in Providence, Rhode Island, an der Ostküste der USA.

(Nelson 1987: 28). Ursprünglich sollte *FRESS* sogar sämtliche Änderungen am Text aufzeichnen (ebenso wie Ted Nelson dies für *Xanadu* konzipiert hatte) und so ein beliebiges Undo gewährleisten. Dies scheiterte allerdings an der (für die damalige Zeit) zu hohen Komplexität des Vorhabens und dem beschränkten Arbeitsspeicher der Großrechner (Van Dam 1988).

FRESS bot zwei Arten von Hyperlinks: *Tags* und *Jumps*. *Tags* waren gerichtete Links, mit denen man bestimmte Elemente mit einer Anmerkung, wie z. B. einer Definition oder Fußnote, versehen konnte. Diese Information wurde zusätzlich angezeigt, ohne dass man das ursprüngliche Dokument verließ (Yankelovich, Meyrowitz et al. 1985). Sie dienten zur *Annotation* von Textpassagen. Nach dem Lesen der Anmerkung gelangte der Leser automatisch zurück zum Ausgangsdokument (s. Tabelle 5).

Jumps führten den Leser hingegen zu einem anderen Dokument. Dieser Link-Typ besaß immer genau einen Quell- und einen Zielanker, wobei sich die Anker nicht nur auf ganze Dokumente, sondern auch auf genaue Positionen in Dokumenten beziehen konnten. Jumps konnten in beide Richtungen verfolgt werden. FRESS war entsprechend das erste Hypertext-System mit *bidirektionalen* Links. Um dies zu ermöglichen, wurden Jumps in einem eigenen Link-Repository gespeichert (DeRose & Dam 1999). Zu jedem Jump gehörte ein Titel, und es konnten zusätzlich Schlüsselwörter und Autoren angegeben werden. Jumps waren somit frei typisiert (s. Abschnitt 4.2.2). Die Typisierung erlaubte es Benutzern, die Verweise nach bestimmten Kriterien zu *filtern*; beispielsweise konnte man sich als Student nur die Verweise und Anmerkungen von Professoren anzeigen lassen, die von Kommilitonen aber ignorieren (Van Dam 1988).

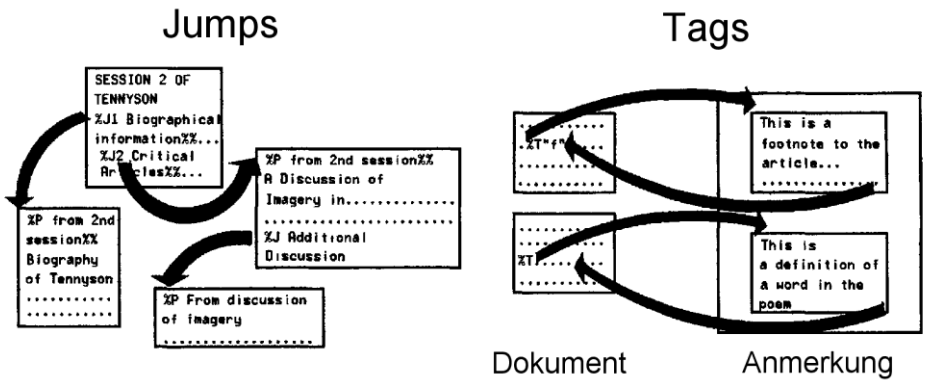


Abb. 172: Tags und Jumps in FRESS (nach Meyrowitz & Van Dam 1982a).

Link-Marker konnten bei *FRESS* im Fließtext eingebettet sein und wurden mit speziellen Zeichenfolgen hervorgehoben (s. Abb. 172): Tags erschienen als „**%T**“ im Text, *Jumps* wurden als „**%J**“ dargestellt, gefolgt von einem Link-Titel, der mit einem doppelten Prozentzeichen abschloss. Zielanker von Links wurden mittels „**%P**“ visualisiert (für „Pmuj“ – „Jump“ rückwärts gelesen). Auf diese Weise konnte man auch Link-Ziele sehen und Links rückwärts verfolgen (Meyrowitz & Van Dam 1982a).

Ebenso wie *NLS* bot *FRESS* bereits eine experimentelle Implementation von Links in Grafiken an. Dafür wurde Anfang der 1970er Jahre eine Schnittstelle zu dem damals „ultramodernen" *IMLAC Graphik Minicomputer 16 bit PDP-8* realisiert, der für den Großrechner als intelligentes Terminal arbeitete.[7] IMLAC war ein Vektorgrafik-System und konnte daher beliebige Zeichen und Grafiken darstellen. Da sich die Maus noch nicht durchgesetzt hatte, wurde zur Eingabe ein Light-Pen verwendet (DeRose 1998).

FRESS wurde für mehr als 10 Jahre an der Brown University für persönliche Hypertext-Bibliotheken eingesetzt. Es gab mehrere kommerzielle Ableger, und es wurde als erstes EDV-System gezielt in der universitären Lehre genutzt (DeRose & Durand 1994: 32). Nach Aussage der Lehrenden wurde *FRESS* von den Studierenden mit unerwarteter Selbstverständlichkeit angenommen, und sein Einsatz führte sogar zu einer wesentlich größeren Menge an von den Studierenden produziertem Material und durchschnittlich deutlich besseren Leistungen und Noten. Obwohl dabei von den Studierenden nur recht wenige Links erstellt wurden,[8] erkannte van Dams Team bereits damals das Lost-In-Hyperspace-Phänomen als Schwachpunkt der Benutzbarkeit von Hypertext-Systemen (s. Abschnitt 3.2.1), da die Anwender oft über Probleme bei der Orientierung klagten (Keep, McLaughlin et al. 1993).

B.4 ZOG / KMS: Hypertext auf Karten mit hierarchischen und assoziativen Links

Das *ZOG*-System war eine Entwicklung der Carnegie Mellon University, die im Jahr 1972 beann. Es war ursprünglich nicht als Hypertext-System konzipiert, sondern als „einfaches" Dokumentationssystem.[9] In *ZOG* wurden alle Dokumente als sogenannte *Frames* („*Rahmen*") repräsentiert, die alle dieselbe Größe hatten und genau den Bildschirm ausfüllten (Trigg & Weiser 1986). Alle Frames waren in einer Datenbank gespeichert und hierarchisch strukturiert. Jedes Frame hatte einen eindeutigen Namen, einen Titel, eine Beschreibung. Am unteren Bildschirmrand zeigte eine Zeile die möglichen Befehle des *ZOG*-Systems an (s. Abb. 173).

Frames konnten beliebige Texte und zwei Arten von Menüpunkten enthalten: Anfänglich waren nur *strukturelle Verweise* zu untergeordneten Frames mithilfe der *Tree Buttons* möglich. Spätere Versionen von *ZOG* boten auch die Möglichkeit *assoziativer Querverweise* (s. Abschnitt 4.5.1.1) mittels der *Annotation Buttons* (s. Tabelle 5). Alle Links waren unidirektional und das Ziel eines Links war immer ein ganzer Frame.

---

[7] *FRESS* lief auf IBM/360 Mainframes mit 128K RAM.

[8] Dies ist möglicherweise darauf zurückzuführen, dass den Studierenden nur ein einziges Grafik-Terminal zur Verfügung stand und das Erstellen von Links mit zeilenorientierten Terminals recht aufwendig war.

[9] Es gab über die Jahre unterschiedliche Versionen von *ZOG*, die konzeptionell aufeinander aufbauten. Die erste Version lief auf IBM Großrechnern, später wurde es auf PERQ Workstations portiert.

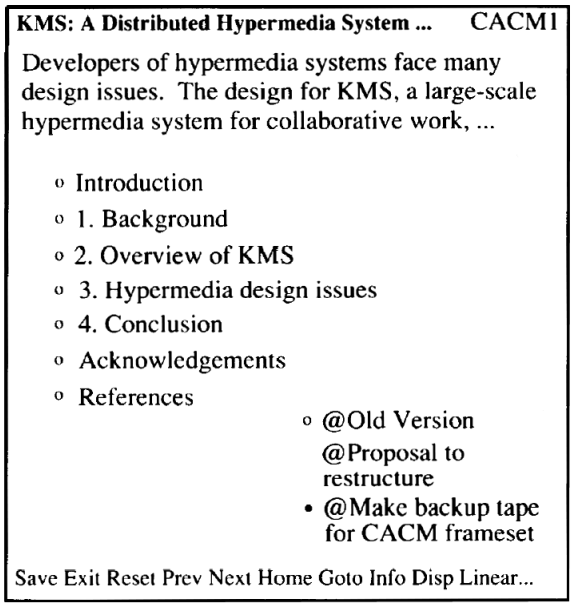


Abb. 173: *KMS* Screenshot, der die unterschiedlichen Link-Typen zeigt. Die letzte Version von *ZOG* hatte eine vergleichbare Schnittstelle (aus Akscyn, McCracken et al. 1988).

Die unterschiedlichen Arten von Links wurden durch spezifische Symbole dargestellt (vergl. Abb. 173): *Tree Buttons* hatten einen Kreis vor dem Link-Text. *Annotation Buttons* hatten zusätzlich zum Kreis ein @-Zeichen vor dem Text. Sie konnten zudem Programme oder Kommandos ausführen, dann wurden sie mit einem ausgefüllten Kreis dargestellt. War das Ziel eines Links nicht verfügbar – handelte es sich also um einen defekten Link –, so erschien kein Kreis vor dem Menütext (s. Abb. 173).

1982 wurde *ZOG* an das US-Militär verkauft, welches es auf dem atomgetriebenen Flugzeugträger „USS Carl Vinson" für Online-Dokumente verwendete (Conklin 1987). Zu dieser Zeit wurde bereits der Nachfolger von *ZOG* entwickelt, das *„Knowledge Management System"* (*„KMS"*). Es lief auf Sun und Apollo Workstations und stellte damit den Übergang zur zweiten Generation von Hypertext-Systemen dar (s. Kapitel 2.2). Es bot ebenfalls Frames, drei Arten von Hyperlinks und baute auf denselben Prinzipien wie *ZOG* auf, war jedoch in der Performanz, den Grafikfähigkeiten und der Benutzbarkeit deutlich verbessert worden. So ließen sich zwei Frames nebeneinander auf dem Bildschirm darstellen, und sie konnten einfache Grafiken enthalten (s. Abb. 174). Die Größe der Frames war aber ebenso wie bei *ZOG* nicht veränderbar, und Scrolling wurde nicht unterstützt (Nielsen 1993a).

KMS war fast komplett durch die Maus zu steuern, und seine Benutzungsschnittstelle setzte für die damalige Zeit neue Maßstäbe. Die meisten Aktionen waren durch einfache Mausklicks auszulösen, wobei alle drei Knöpfe der Maus eine unterschiedliche Bedeutung hatten. Der Benutzer konnte immer anhand einer Art Tooltip beim Mauszeiger erkennen, welchen kontextabhängigen Funktionen die drei Knöpfe im Moment gerade zugeordnet waren (s. Abb. 174). Dies erleichterte die Bedienung des Systems insbesondere für Anfänger.

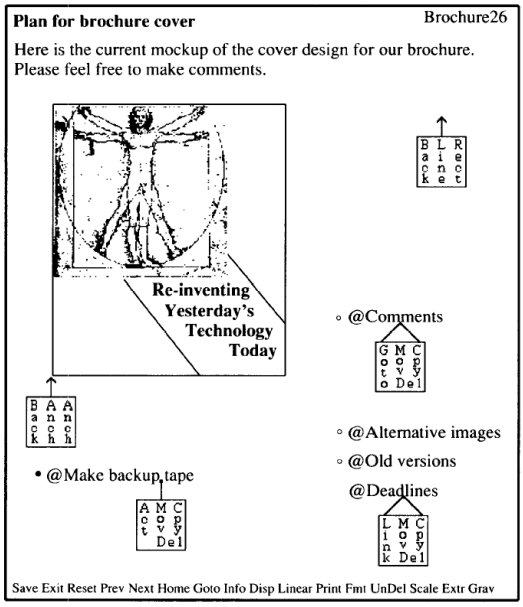


Abb. 174: Fünf Beispiele für Tooltips beim Mauszeiger von *KMS*. Die Texte repräsentieren die Funktionen der drei Mausknöpfe (aus Akscyn, McCracken et al. 1988).

B.5 Xanadu: Ein globales Informationssystem mit Versionsverwaltung und Micro-Payments

„The longest-running vaporware project in the history of computing – a 30-year saga of rabid[10] prototyping and heart-slashing despair" (Wolf 1995).

Xanadu ist wohl eines der bekanntesten, niemals verwirklichten Software-Projekte. Es blieb bis heute „imaginär", da es von seinem Schöpfer Ted Nelson oft detailliert beschrieben und angekündigt, aber nie realisiert wurde (s. Abschnitt A.3).

Ted Nelson begann mit seinen Überlegungen zu *Xanadu* bereits Mitte der 1960er Jahre an der Brown University, die Ausgestaltung der Konzepte erfolgte aber erst in den folgenden Jahren. *Xanadu* kann als ein Lebenswerk von Ted Nelson angesehen werden, da er es bis in die 1990er Jahre hinein ständig weiterentwickelte. Beispiele für seine Adaptionen sind grafische Benutzungsschnittstellen und das Arpanet, deren Möglichkeiten er früh für sein Projekt berücksichtigte. Es ist daher schwer, von „dem einen" *Xanadu*-Konzept zu sprechen.

Xanadu bot eine Vision, die für die damalige Zeit als ausgesprochen ehrgeizig angesehen werden kann: Es sollte einmal *sämtliche von Menschen erschaffenen Dokumente* fassen können. In Xanadu sollte zudem nie etwas gelöscht werden, stattdessen sollte es *sämtliche Änderungen und Erweiterungen protokollieren* und so später nachvollziehbar machen (Nelson 1987).

Realisiert werden sollte dies mithilfe des Konzepts der *Transklusionen*. Sie erlaubten das Einfügen beliebiger Teile existierender Dokumente in eigene Publikationen, wobei eine Referenz auf den ursprünglichen Urheber erhalten blieb. Aus technischer Sicht musste dadurch jeder Textteil nur ein einziges Mal gespeichert werden, egal in wie vielen Dokumenten er existiert.

[10] **rabid**: ['ræbid] toll, rasend, wütend; … fanatisch, übereifrig; … (Weis, Weis et al. 1967).

Für die Autoren sollten Transklusionen das Erstellen von neuen Dokumenten vereinfachen und beschleunigen, indem sie andere Dokumentteile effizient wiederverwenden und in eigene Arbeiten einbinden können, ohne dass sie sich selbst explizit um Copyrights zu kümmern hatten. Stattdessen sollte *Xanadu* die Wahrung der Urheberrechte sicherstellen, indem die Verwendung von Materialien automatisch mittels sogenannter *Royalties* – Ted Nelsons Variante von *Micro-Payments* – vergütet wurde (Nelson 1987). Ted Nelson sieht Transklusionen als eine spezielle Variante von Hyperlinks und propagiert sie bis heute als ein essenzielles Feature von Hypertext-Systemen.

In *Xanadu* lassen sich natürlich auch assoziative Verknüpfungen zwischen Dokumenten erstellen. In seinem Buch *Dream Machines* (Nelson 1974) definiert Nelson drei Kategorien von Hypertext mit charakteristischen Link-Eigenschaften (s. Tabelle 5):

- Die erste namens *Chunk Hypertext* bietet *Referenz-Links*, die zu einem anderen Objekt verweisen und das aktuelle Dokument ersetzen.
- Die zweite Variante nannte er *Stretchtext*: Sie stellt *Expansion Links* zur Verfügung, die direkt unterhalb des Links das Zielobjekt einblenden. Dieses Konzept wurde später beispielsweise vom Guide-System realisiert (s. Abschnitt B.7).
- Die dritte Hypertext-Form bezeichnete er als *Transpointing Windows* und sollte die Bedeutendste in *Xanadu* darstellen: Sie visualisiert Links zwischen Objekten mittels Linien (s. Abb. 175), die auch über die Fensterränder hinaus unterschiedliche Textstellen visuell verknüpfen können (s. Abb. 176).

Alle Links sind in Nelsons Projekt frei typisiert. Sie besitzen einen Startpunkt, einen Endpunkt und einen Verweis auf einen Text, der den Typ des Links definiert (Nelson 1974). Bemerkenswert ist dabei, wie Ted Nelson das Problem inkonsistenter Links handhabt: Da Dokumente nie gelöscht werden, können Links auch nie veralten und fehlerhaft werden.

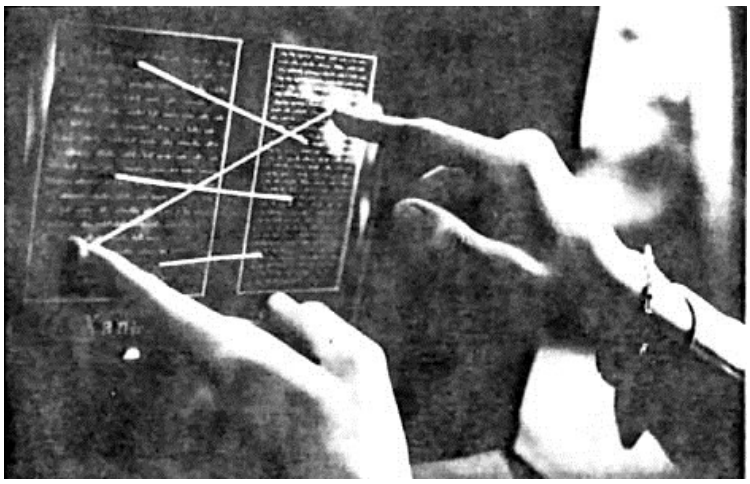


Abb. 175: Mockup der Transpointing Windows in Xanadu (aus Nelson 1972).

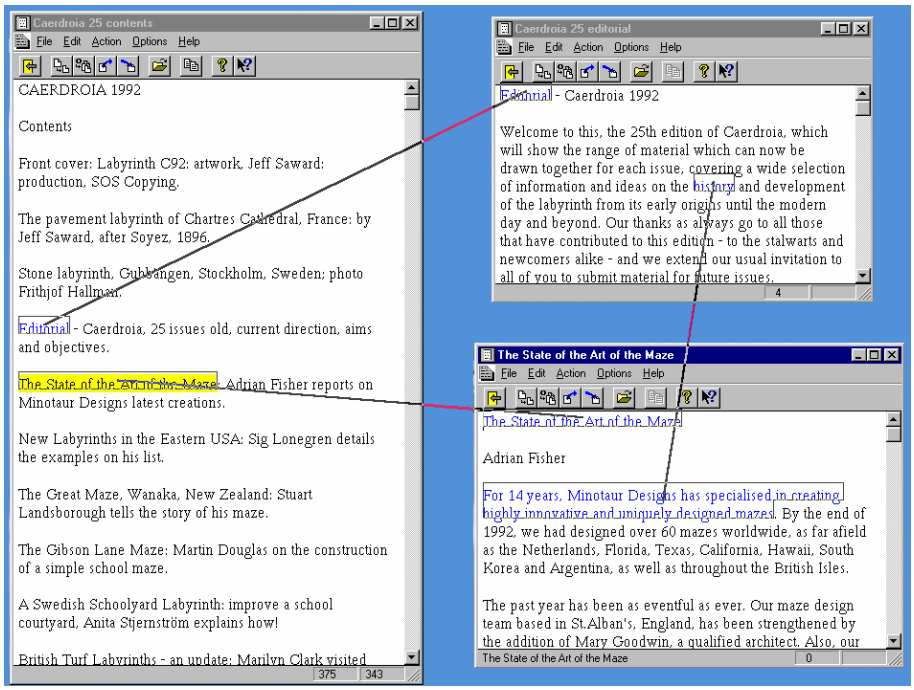


Abb. 176: Ein Xanadu-Prototyp mit "Transpointing Windows" (aus Nelson 1999).

Nelson hat in den letzten 45 Jahren mehrfach angekündigt, dass *Xanadu* demnächst vollendet und einsatzbereit sein solle. Obwohl dies bis heute nicht eingetroffen ist und er dafür auch Spott erntete (s. Zitat am Anfang des Abschnitts), haben seine Entwürfe und Publikationen zahlreichen Forschern als Inspirationen gedient. Das World Wide Web realisiert viele der Ideen von *Xanadu*, obgleich Nelson das Web gerne energisch kritisiert. In (Nelson 1999) geht er ausführlich auf die Schwächen des Web ein, wobei er vor allem die fehlenden Versionierungsmechanismen, fehlende Konzepte für die Wahrung von Urheberrechten und die vielen kaputten Links beanstandet. So fordert Ted Nelson auch „feiner granulierte" Mechanismen für die Einbindung von Objekten im Web und eine umfassende Realisierung von Micro-Payment-Systemen (Nelson 1999).

Er setzt mit diesen Forderungen hohe Maßstäbe, andererseits zeigen auch seine Konzepte Schwächen: Es gibt offenkundige Probleme bezüglich der Skalierbarkeit, beispielsweise bei *Transklusionen* und *Royalties*, in der von ihm vorgeschlagenen feinen Granularität. Zudem haben einige der ihm bis heute vertretenen User-Interface-Konzepte (beispielsweise die *Transpointing Windows*) einen eher eigenwilligen und wenig gebrauchstauglichen Charakter.

B.6 TextNet: Typisierte Links zur Unterstützung des wissenschaftlichen Diskurses

Das Hypertext-System *TextNet* wurde Anfang der 1980er Jahre von Randal Trigg[11] von der University of Maryland entwickelt. Ziel von *TextNet* war es, ein Werkzeug zu erschaffen, das den wissenschaftlichen Diskurs in der akademischen Gemeinschaft fördert. Um dies zu er-

[11] Randal Trigg gilt als der „Erfinder" streng typisierter Hyperlinks (s. Abschnitt 4.2.3). Seine Dissertation (Trigg 1983) wird in diesem Zusammenhang als erste Publikation zu dem Thema angesehen.

reichen, sollte es möglich sein, gemeinsam Texte zu erstellen, zu annotieren und zu kritisieren (Trigg & Weiser 1986). Obwohl *TextNet* Terminals mit reiner Textausgabe[12] verwendete, bot es als erstes Hypertext-System ein User Interface mit *überlappenden Fenstern* und *scrollbaren Fensterinhalten*.

TextNet verfügte über zwei Arten von Knoten: „*Tocs*“ („*Table of Contents*“) dienten der *hierarchischen Strukturierung* der Informationen, wogegen „*Chunks*“ Dokumente mit textuellem Inhalt darstellten. Links wurden in *TextNet* als *eigene Objekte* realisiert und besaßen einen Start- und einen Zielanker sowie einen Typ (Trigg & Weiser 1986).

Die grundlegende Neuerung von *TextNet* war die systematische Typisierung der Links. TextNet verfügte über zwei Basistypen von Verweisen: normale Links und Links zur Annotation („*Commentary Links*“). Diese beiden Klassen waren weiter unterteilt und bildeten eine Typhierarchie mit über 80 Link-Typen (s. Abb. 28 auf Seite 83 und Tabelle 5). Die von Trigg in seiner Dissertation vorgeschlagene *Link-Taxonomie* sollte zur kollaborativen kritischen Argumentation in TextNet genutzt werden. Sie basiert auf der Beobachtung, dass es im Allgemeinen nur eine beschränkte Anzahl von Kommentar- und Anmerkungsarten auf Argumente geben kann.

Für jede dieser Arten existiert in seinem System auch ein Link-Typ. Beispielsweise gibt es Links zu Aussagen, die ein Dokument unterstützen als auch Typen für Links zu Gegenargumenten (Trigg 1983). Diese sind weiter differenziert in Links, die zeigen, dass ein Punkt irrelevant ist („*Pt-irrelevant*“), Daten unangemessen zitiert wurden („*D-inadequate*“) oder (stilistisch) von Thema abschweift („*S-rambling*“, s. Abschnitt 4.2.3 und Abb. 28 auf Seite 83).

Trigg diskutiert in seiner Arbeit bereits die Vor- und Nachteile einer solchen strengen Typisierung (Trigg 1983): Einerseits sieht er das Defizit, dass die limitierte Zahl von Link-Typen das *Anwendungsgebiet beschränkt*, auf der anderen Seite erhält man Möglichkeiten, das Dokument spezialisiert zu *verarbeiten* (Trigg 1983). Beispielsweise sollte *TextNet* den Hypertext zu einem Argumentationsstrang *linearisieren* können und so implizit das Konzept der Pfade implementieren (Zellweger 1989). Des Weiteren konnten Dokumente auch als geordnete Listen zusammengefasst werden, um sie dem Benutzer am Bildschirm zum Lesen zu präsentieren oder um sie auszudrucken (Conklin 1987).

Das Bildschirmfoto von *TextNet* (s. Abb. 177) zeigt rechts oben das Menü zu einem *Chunk* (Dokument) mit dem Titel „`conclusion`“. In dem Menü werden Meta-Informationen zum Dokument sowie eingehende und ausgehende Verweise mit ihrem jeweiligen Namen angezeigt. Der Benutzer hat einen Verweis zum Objekt namens „`sacred`“ ausgewählt, das im linken Fenster zu sehen ist. In der Mitte dieses Fensters werden drei Links mittels des Codes „`RF`“ für *Reference* dargestellt.

[12] *TextNet* lief auf einem VAX 11/780 Großrechner unter Unix und war in Lisp programmiert.

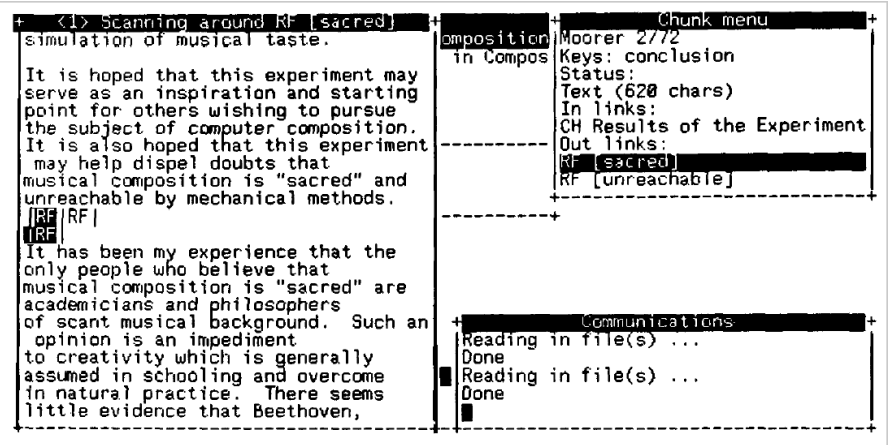


Abb. 177: Ein Bildschirmfoto des Systems *TextNet* (aus Trigg & Weiser 1986).

Bemerkenswert ist, dass der jeweilige Link-Typ in dieser Standard-Ansicht von *TextNet* **nicht** erkennbar ist, aber die Namen der Objekte zu sehen sind. Eine weitere Auswertung der Link-Typen (wie konzeptionell eigentlich vorgesehen) erfolgte hier offensichtlich auch nicht; Trigg schrieb hierzu (Trigg 1983):

> "For instance, readers could provide information to the effect that they desire a quick walk-through excluding extended examples and justification. By inspecting link types, the system could tailor the default path to those desires. TextNet currently does not do this."

Offensichtlich blieb somit *TextNet* in der Realisierung typisierter Links deutlich hinter den in seiner Dissertation beschriebenen Konzepten zurück.

B.7 Guide: Dokumentenverwaltung mit Stretchtext und adaptivem Mauszeiger

Das *Guide-System* entstand ab 1982 an der Universität von Kent in Canterbury. Ursprünglich lief es auf PERQ-Workstations, später entwickelte es die schottische Firma OWL (Office Workstations Limited) weiter und portierte es 1986 auf Macintosh-Computer und ein Jahr später auf IBM-PCs. Guide war das erste kommerzielle Hypertext-System für Macintosh-Computer (Keep, McLaughlin et al. 1993; Gillies & Cailliau 2001).

Die Entwickler von *Guide* hatten sich ein System zum Ziel gesetzt, mit dem auf effiziente Weise Dokumente am Bildschirm *erstellt* und *gelesen* werden konnten. Guide verfügte daher über einen integrierten Editor zum Bearbeiten von Dokumenten und Links. Dokumente waren hierarchisch strukturiert, sie konnten beliebig lang sein und wurden in scrollbaren Fenstern dargestellt.

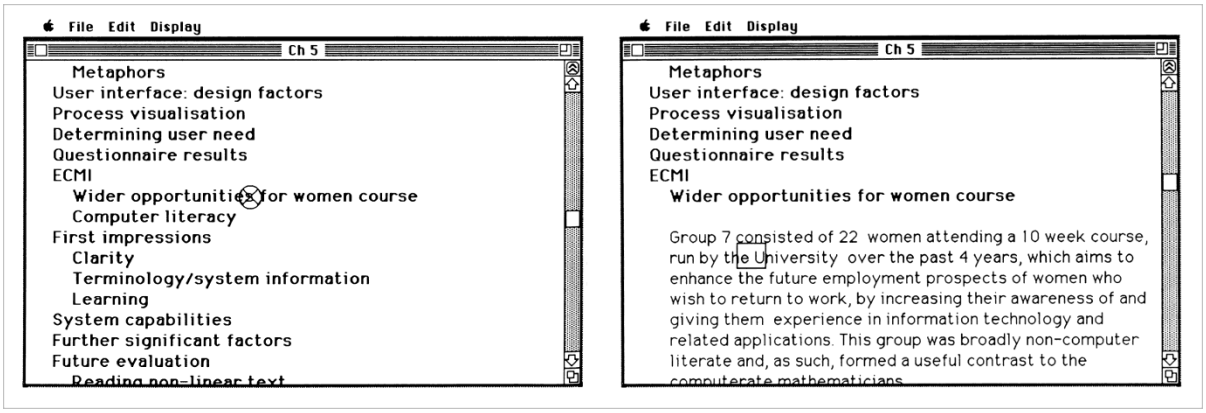


Abb. 178: Ein *Inline Link* im Guide-System: Links sieht man die Übersicht, rechts die expandierte Version des ausgewählten Links (aus Rada 1991: 48ff).

Die ersten Versionen von *Guide* konzentrierten sich auf sogenannte *Inline Links*: Bei ihrer Auswahl wurde im aktuellen Dokument unterhalb des Hyperlinks ein Bereich eingefügt, in dem die zusätzlichen Informationen erschienen. Ein weiterer Link-Klick lies diesen Bereich wieder verschwinden (s. Abb. 178). Auf diese Weise bekam der Benutzer eingangs nur die obersten Hierarchiestufen eines Dokuments zu sehen und konnte interaktiv Details ein- und ausblenden und so im Text navigieren (Brown 1987). Ted Nelson machte diese Art von Hypertext als *Stretchtext* bekannt (vergl. Abschnitt B.5; Nelson 1974). Sie wird bis heute insbesondere im Bereich adaptiver Hypertext-Systeme als bedeutsam angesehen (Brusilovski 1996).

Spätere Versionen von *Guide* unterstützten *vier* unterschiedliche Link-Arten: Neben den bereits erwähnten *Inline Links* kamen zusätzlich *Note Links* hinzu, die in einem eigenen Popup-Fenster erschienen, solange man die Maustaste drückte. Zudem gab es noch *Replacement Links*, bei denen das Quelldokument durch das Link-Ziel ersetzt wurde und *Command Links*, die zum Starten von (externen) Programmen dienten (s. Tabelle 5).

Guide stellte ähnlich wie *HyperTies* Link-Anker als Teil des Fließtextes dar, hob sie aber typografisch hervor. Einige Versionen von Guide verwendeten unterschiedliche Methoden der Hervorhebung für unterschiedliche Link-Arten: *Replacement Links* und *Command Links* erschienen als kursiver Text, *Note Links* in Fett und *Expansion Links* wurden unterstrichen. Zudem visualisierten unterschiedliche Mauszeiger den Link-Typ vor dem Anklicken eines Links (s. Abb. 56 auf Seite 151).

In den 1990er Jahren wurden die Möglichkeiten von *Guide* weiter ausgebaut: Es entstand ein Hypertext-Programm mit multiplen Fenstern, komfortablem Editor, History-Funktion, integrierter Volltextsuche und Annotationswerkzeugen (s. Abb. 179).



Abb. 179: Das Guide-System für Windows aus dem Jahre 1996.

B.8 HyperTies: Embedded Menues, Link-Previews und viele Navigationswerkzeuge

Im Jahre 1983 begannen Ben Shneiderman und sein HCI-Lab an der University of Maryland mit der Entwicklung des Projekts „*TIES - The Interactice Enceclopedia System*“. Ursprünglich war *TIES* als einfaches Online-Nachschlagewerk konzipiert, aber später um Hypertext-Funktionalität erweitert und in *HyperTies* umgetauft (Shneiderman & Kearsley 1989). Die ersten Versionen von *HyperTies* waren noch rein textbasierte DOS-Programme. Ende der 1980er erhielt *HyperTies* Grafikfähigkeiten, und es kam eine X-Windows-Version für Workstations hinzu.

HyperTies war das erste in größerem Umfang eingesetzte Hypertext-System auf PCs und wurde ab Mitte der 1980er Jahre kommerziell vertrieben. Die Anwendungsgebiete von *HyperTies* waren vielfältig, unter anderem gab es darauf basierende interaktive Enzyklopädien, Online-Wartungshandbücher und Museumsführer.

Bekannt machte *HyperTies* das Konzept der *Embedded Menues* (Koved & Shneiderman 1986): Links waren Abschnitte des Fließtextes und wurden lediglich durch eine charakteristische Textfarbe hervorgehoben (s. Abb. 176). Der Text des Link-Ankers informierte den Benutzer über die Bedeutung des Links. Dieses Konzept sollte gegenüber konventionellen Menüs in Listenform einen natürlicheren Umgang mit dem Computer erlauben. Dabei wurde bereits damals eine „dezente“ Hervorhebung der Links als wichtig angesehen, damit der Lesefluss des Benutzers möglichst wenig gestört würde (Marchionini & Shneiderman 1988).

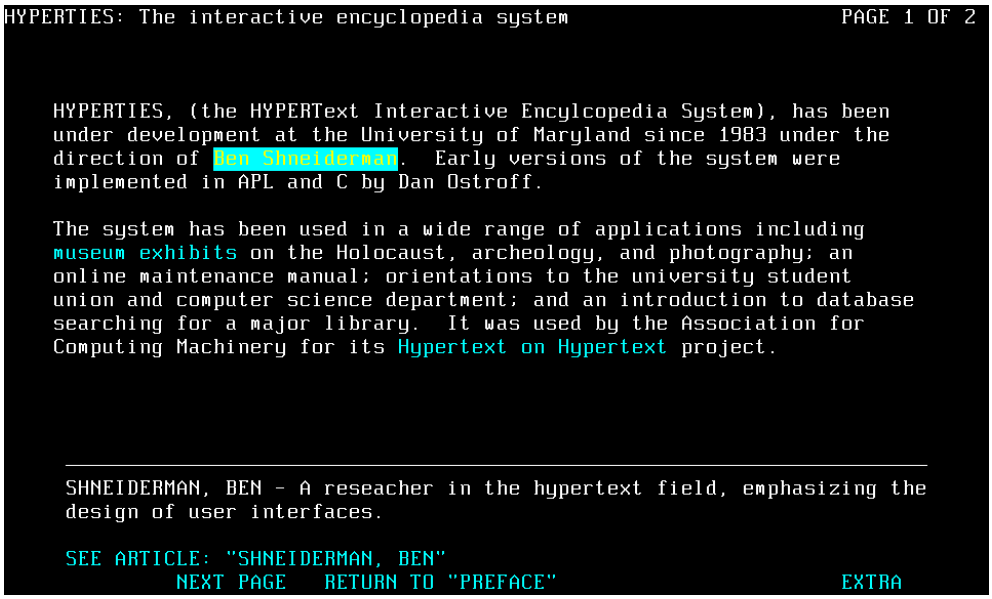


Abb. 180: Eine Text-Seite in der DOS-Version von HyperTies (aus Shneiderman & Kearsley 1989).

HyperTies unterschied Objekte verschiedener Medientypen, wie textliche und grafische Knoten, wobei erstere auch über mehrere Bildschirmseiten gehen konnten, zwischen denen man blätterte. *HyperTies* kannte allerdings keine Link-Typisierung, und die Auswahl eines Links führte immer zu einem neuen Dokument, welches das angezeigte ersetzte (s. Tabelle 5). Da die Links in den Knoten-Objekten eingebettet und uni-direktional waren, trat hier bereits das Problem defekter Links auf. Bereits damals erkannten die Entwickler die Notwendigkeit von Werkzeugen, mit denen solche „*Dangling Links*“ vermieden werden (Shneiderman & Kearsley 1989).

Abb. 180 zeigt das Beispiel eines Textdokuments in *HyperTies*. Der Text enthält drei Links, wobei der erste („Ben Shneiderman“) angewählt ist. Die Anwahl eines Links erfolgt mittels der Cursor-Tasten, einer Maus oder eines Touchscreens. Eine Innovation war der *Link-Preview* von *HyperTies*: Nach der Anwahl eines Links erhielt man erst eine kurze Beschreibung des Zieldokuments im unteren Bildschirmbereich.[13] Erst in einem zweiten Schritt sprang man zum Ziel des Links. So konnte der Benutzer zusätzliche Informationen zur Bedeutung eines Links erhalten, bevor er das Quelldokument verließ.

Am unteren Bildschirmrand sind drei zusätzliche Navigationsmöglichkeiten in *HyperTies* zu sehen. Mittels „`NEXT PAGE`“ wurde eine sequenzielle Navigation zur nächsten Seite eines Knotens angeboten. Der Menüpunkt „`RETURN TO...`“ entsprach dem *Back-Button* heutiger Browser und gewährleistete jederzeit die „Reversibilität der Benutzeraktionen“ (Shneiderman & Kearsley 1989). Diese Funktion sah man als so wichtig an, dass für Sie die „ESC“-Taste reserviert war.

[13] In Abb. 180 ist unten der Hinweistext „SHNEIDERMAN, BEN – A researcher in the hypertext field, emphasizing the design of user interfaces.“ zu sehen, der zum gerade aktiven Link gehört.

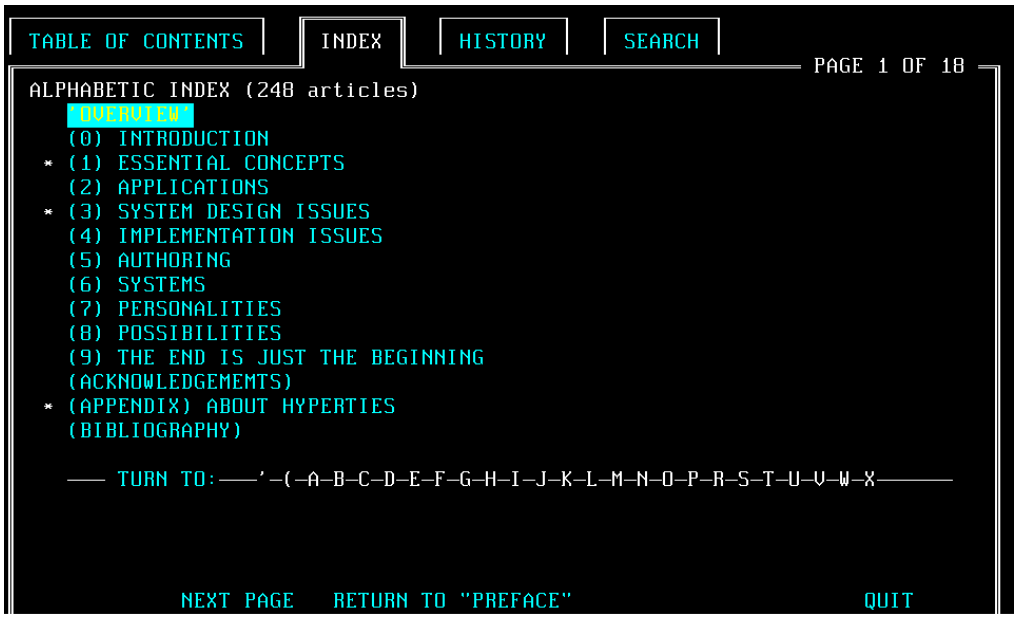


Abb. 181: Die „Extra-Funktionen“ von HyperTies (aus Shneiderman & Kearsley 1989).

Der dritte Menüpunkt „`Extra`“ führte zu weiteren Navigationsfunktionen von *HyperTies* (s. Abb. 181). Diese erweiterten Navigationsfunktionen umfassen einen *Index aller Seiten*, eine *History-Liste* der zuletzt besuchten Dokumente sowie eine integrierte *Volltext-Suchfunktion*. Diese Navigationswerkzeuge sind in *HyperTies* integriert und werden dem Benutzer automatisch zur Verfügung gestellt. Im Unterschied zu anderen frühen Hypertext-Systemen, bot es aber keine hierarchischen Strukturierungs- oder Navigationsmöglichkeiten.

Konzeptionell lassen sich viele Parallelen zum World Wide Web erkennen: Links werden ebenfalls als hervorgehobene Textabschnitte angezeigt, sie sind unidirektional und in die Dokumente eingebettet. *HyperTies* hatte ebenfalls eine Back- und eine History-Funktion. Mit seiner integrierten Volltext-Suche, dem Seitenindex und der Link-Vorschau ging es teilweise sogar über die Konzepte des Web hinaus.

B.9 NoteCards: Übersichtskarten und programmierbare Links

Die Arbeit an *NoteCards* begann Mitte 1984 bei *Xerox PARC*. Randall Trigg (s. Abschnitt B.6: „TextNet“) traf dort nach seiner Dissertation auf Frank Halasz und Tom Moran, die gemeinsam zu den leitenden Entwicklern des Projekts wurden (Conklin 1987).

Konzeptionell basierte *NoteCards* auf der Metapher der Karteikarte (Halasz, Moran et al. 1987). Diese Karten – die damals ebenso wie das Gesamtprojekt „*NoteCards*“ genannt wurden – konnten unterschiedliche Größen haben, ein Scrolling in den Karten war allerdings nicht möglich. Das System war in der Lage, beliebig viele Karten überlappend auf dem Bildschirm darzustellen (Halasz, Moran et al. 1987). *NoteCards* sollte ursprünglich dem Erfassen, Strukturieren und Bearbeiten von neuen Ideen und Konzepten dienen, wurde aber auch für andere kreative Aufgaben verwendet, wie der Erstellung von Forschungsberichten (Trigg & Irish 1987).

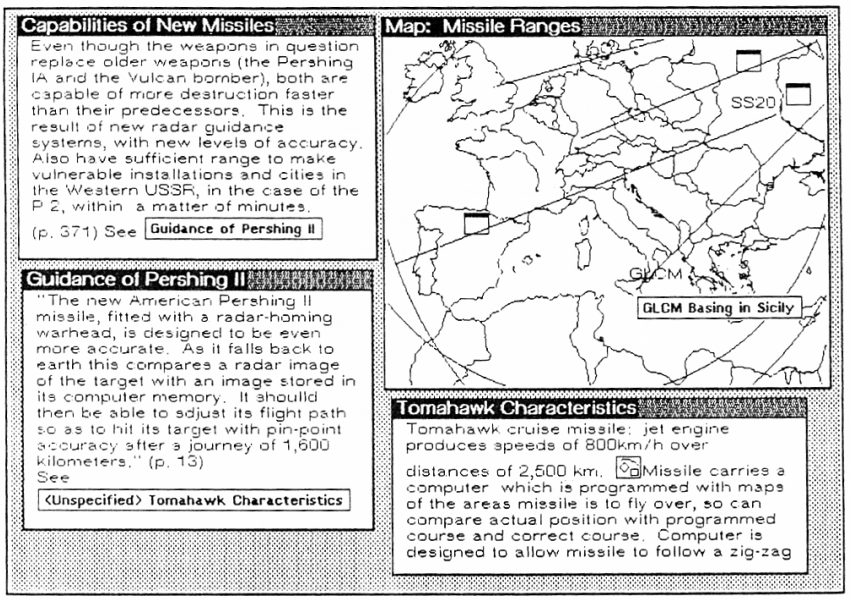


Abb. 182: Ein NoteCards-Bildschirmfoto mit drei Textkarten und einer Karte vom Typ „Map" (aus Halasz 1988).

In *NoteCards* hatte jede Karte einen *Titel* und einen *Typ*. Es gab über 40 verschiedener solcher Kartentypen, die jeweils die Einbindung unterschiedlicher Medienarten wie Texten, Zeichnungen oder Grafiken ermöglichten (s. Abb. 182). Die Bearbeitung der Karten war direkt durch den integrierten Editor möglich (Rada 1991).

Karten wurden mit beliebigen anderen Karten mittels *unidirektionaler typisierter Links* verknüpft. Startanker konnten innerhalb einer Karte positioniert sein, das Link-Ziel war hingegen immer eine ganze Karte. Der Link-Typ lies sich mittels eines Titels definieren, wodurch beliebige neue Typen möglich waren (Keep, McLaughlin et al. 1993). Bemerkenswert ist hierbei, dass Randal Trigg in *NoteCards* auf eine vordefinierte Typhierarchie (wie in seinem vorherigen Projekt *TextNet*, vergl. Abschnitt B.6) verzichtet hatte (s. Tabelle 5), da NoteCards *universeller* im Einsatz sein sollte (Halasz, Moran et al. 1987).

Die Darstellung von Link-Ankern erfolgte entweder mittels eines kleinen *umrahmten Icons* oder in Form eines *Kastens* mit dem Titel der Zielkarte als Ankertext. Wurde ein Link-Typ angegeben, so erschien dieser zusätzlich in spitzen Klammern vor dem Titel des Zielobjekts (s. Abb. 182, links unten[14]). Nach dem Anklicken eines Links wurde ein neues Fenster für das Zielobjekt geöffnet bzw. das Fenster mit dem Objekt nach vorne geholt, falls es schon geöffnet war. Man konnte beliebig viele dieser Fenster auf dem Bildschirm darstellen, allerdings stellte sich heraus, dass Benutzer schnell die Orientierung auf einem mit vielen Fenstern gefüllten Desktop verloren (Halasz 1988).

[14] Bemerkenswert ist, dass die Autoren des Systems es selbst bei diesem Beispiel-Bildschirmfoto offensichtlich versäumten, einen Typ für den Link zu definieren. Folglich erscheint als Typangabe „<Unspecified>".

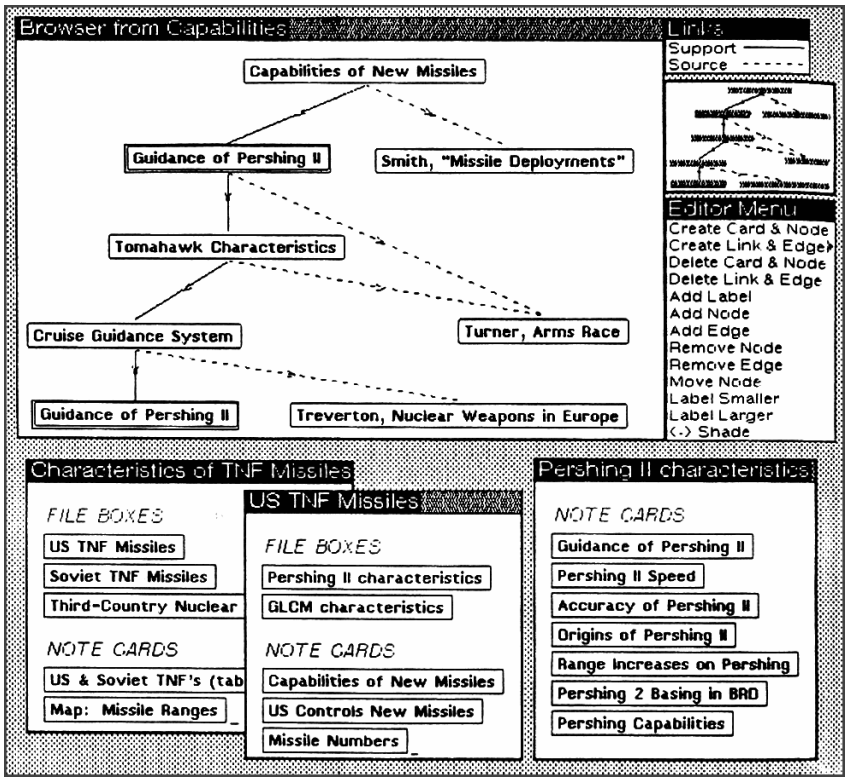


Abb. 183: Eine „Browser Card" und mehrere „Fileboxes" in NoteCards (aus Halasz 1988).

Innovativ war an *NoteCards* auch seine große Flexibilität. Da das System in der Interpretersprache LISP[15] programmiert war, konnte es über LISP-Funktionen beliebig angepasst und erweitert werden. So ließen sich neue Kartentypen definieren sowie Links mit Funktionen verknüpfen, um neue Verknüpfungsmöglichkeiten zu realisieren (Trigg & Irish 1987). Eine solche Erweiterung war die experimentelle Umsetzung multipler Links in *NoteCards* (Halasz, Moran et al. 1987).

Jede Karte war in eine hierarchische Struktur eingebunden. Die *Filebox* – selbst ebenfalls ein Kartentyp – diente als „Container" für andere Karten (Abb. 183 unten).

Ein Novum von *NoteCards* waren die lokalen und globalen Übersichtskarten des Hypertextes innerhalb der *Browser Cards* (vergl. Abb. 183 oben). Sie erlaubten es, einen grafischen Überblick der Knoten und Link-Strukturen zu erhalten. Unterschiedliche Link-Typen wurden mittels verschiedener Linienmuster dargestellt. Die *Browser Card* war interaktiv, sodass man die angezeigten Knoten und Links auch auswählen und bearbeiten konnte.

Als weiteres Navigationswerkzeug bot *NoteCards* eine einfache Suchfunktion, mit der allerdings nur die Titel der Seiten durchsuchbar waren. Zudem gab es die Möglichkeit, Pfade entlang mehrerer Dokumente als *Guided Tours* zu definieren (Trigg 1988). Eine besondere History-Funktion waren *Tabletops*, mit denen man eine gesamte Desktop-Ansicht abspeichern und später wieder herstellen konnte.

[15] *NoteCards* wurde in einem speziellen LISP-Dialekt für hochauflösende *Xerox D Grafik-Workstations* implementiert.

Obgleich *NoteCards* außerhalb von Xerox PARK aufgrund der besonderen Hardwarevoraussetzungen kaum verbreitet war, fanden doch die realisierten Konzepte erhebliche Beachtung: Insbesondere die Übersichtskarten des Hypertextes und die Erweiterbarkeit von *NoteCards* können als wichtige Inspirationen für die Entwicklung vieler folgender Systeme angesehen werden.

B.10 Intermedia: Ein Mehrbenutzer Client-Server-Framework für Hypermedia

Das erste verteilte Hypertext-System war gleichzeitig eines der umfassendsten Projekte in der Geschichte der Hypertext-Forschung: *Intermedia*. Seine Entwicklung begann 1983, als Andy van Dam (s. Abschnitt B.3) zusammen mit William S. Shipp und Norman Meyrowitz[16] das *Institute for Research in Information and Scholarship (IRIS)* gründeten. Ziel ihres Projekts *Intermedia* war es, ein erweiterbares, objektorientiertes *„Multiuser Hypermedia Framework"* zu erschaffen, das die Kollaboration mehrerer Benutzer unterstützte und die Integration beliebiger Medientypen und Applikationen erlaubte (Haan, Kahn et al. 1992). *Intermedia* kann genau genommen nicht als *eine* Anwendung gesehen werden, sondern steht für ein Projekt, das ein eigenes Desktop-System beinhaltet und eine ganze Reihe von Client-Server-Applikationen mit Multi-User-Support umfasst.[17]1985 wurde die erste Version veröffentlicht, ab 1989 war Intermedia auch kommerziell erhältlich (Gillies & Cailliau 2001).

Intermedia war anfänglich primär für die akademische Lehre konzipiert, wo es auch mit Erfolg systematisch eingesetzt wurde (Beeman, Anderson et al. 1987). Über den bloßen Wissenserwerb hinaus sollte das System die ganzheitliche, kritische Einordnung von Lehrmaterialien unterstützen. Als *Mehrbenutzersystem* bot es daher auch drei Arten von Zugriffsrechten: Nur Lesen, Lesen mit der Möglichkeit, Anmerkungen hinzuzufügen, sowie volle Lese- und Schreibrechte (Meyrowitz 1986).

Ähnlich wie in *NoteCards* hatten die Karten in *Intermedia* verschiedene Typen, wie „Artikel", „Anmerkung" oder „Illustration". Die Karten konnten unterschiedlich groß sein und wurden in einem scrollbaren Fenster variabler Größe dargestellt. Die Client-Programme zur Anzeige der Dokumente *InterWord* (Texte), *InterDraw* (Grafiken), *InterVideo* (Animationen) und *InterVal* (zeitliche Abläufe) waren auch gleichzeitig als Editoren zu verwenden. *Intermedia* war das erste Hypertext-System, das Filme und Animationen einbinden und mit Links versehen konnte.

Ein wichtiges Konzept zur Unterstützung der Lehre war in *Intermedia* das „*Web*". Der Begriff stand für die unabhängige Speicherung von Dokumenten und Links in eigenen Datenbanken. Diese Trennung von Inhalt und Struktur ermöglichte es, die gleichen Materialien für

[16] Norman Meyrowitz gilt auch als Vater der Multimedia-Autorensysteme *Macromedia Shockwave* und *Adobe Flash*.
[17] Implementiert wurde Intermedia unter A/UX 1.1, Apples damaliger Version von Unix für Macintosh-Computer (Meyrowitz 1986).

unterschiedliche Kurse zu verwenden, indem verschiedene Link-Strukturen die Materialien jeweils themengerecht zugänglich machten. Da in *Intermedia* Link-Anker auch zu schreibgeschützten Dokumenten hinzugefügt werden konnten, waren Studenten in der Lage, sämtliche Materialien selbst mit Verknüpfungen und persönlichen Annotationen zu versehen (Nielsen 1995b: 51ff). Mithilfe mehrerer Link-Datenbanken konnten spezielle Navigationsmöglichkeiten entsprechend der Benutzeraufgaben und -bedürfnisse angeboten werden.

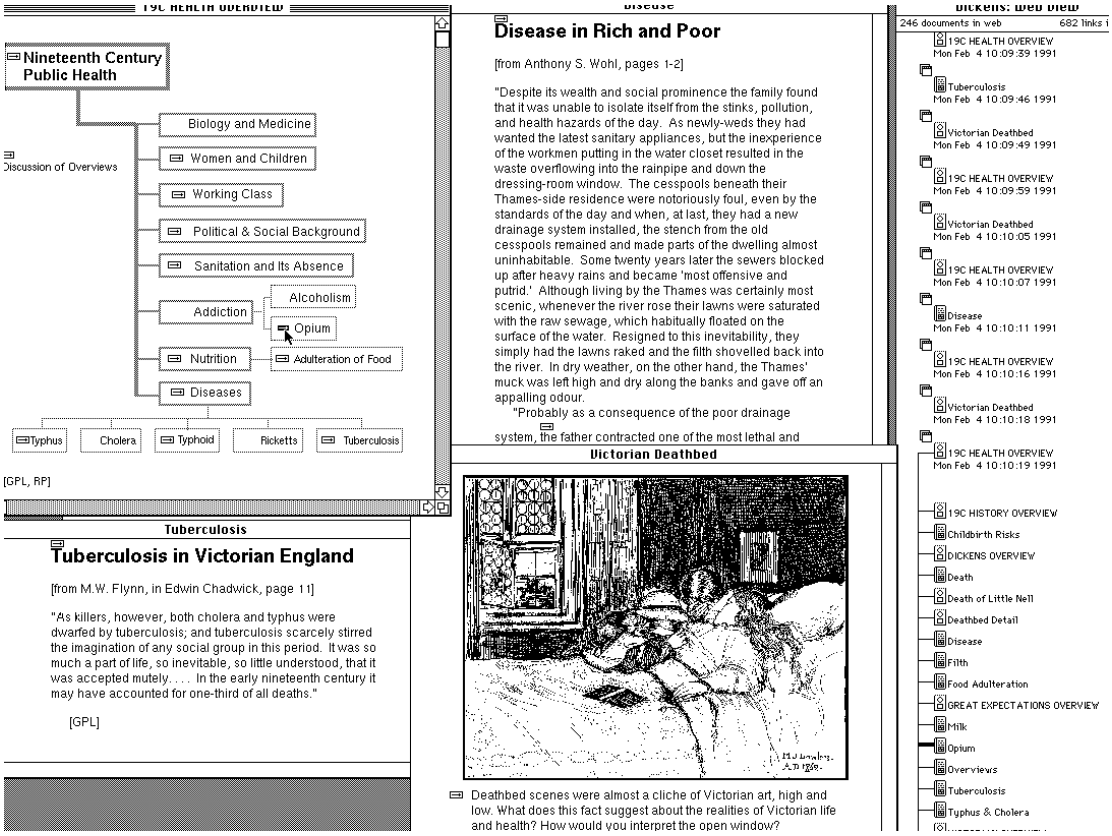


Abb. 184: Ein Screenshot von Intermedia (aus Landow 2004).

In *Intermedia* waren Hyperlinks als eigene Objekte mit Referenzen auf *zwei Anker* realisiert, die ebenfalls als Objekte repräsentiert wurden. Links waren generell bidirektional und konnten in beide Richtungen verfolgt werden. Zudem verfügten sie über mehrere Attribute wie Titel, Autor, Erstellungsdatum und Schlüsselwörter, die zur *Filterung* der Links dienten. Link-Anker waren an bestimmte Positionen innerhalb der Dokumente gebunden, wobei die Adressierung der Anker-Position konzeptionell so realisiert war, dass kleinere Änderungen an den Dokumenten nicht zu fehlerhaften Links führten (Nielsen 1993a: 109).

Link-Marker erschienen grundsätzlich als kleines Icon in Form eines Pfeils in einem Kasten, der oberhalb der Ankerposition dargestellt wurde (s. Abb. 184, Abb. 185 und Tabelle 5). Dieses Link-Icon wurde ebenfalls für Grafiken und Animationen verwendet und oberhalb des verknüpften Objekts platziert. Aufgrund dieser einheitlichen und abstrakten Darstellung war für Benutzer oft nicht zu erkennen, welchen Zweck und welches Ziel ein Link hatte. Link-Anker verfügten daher zusätzlich über einen *Link Extent*, der einen Umfang für Link-Anker definierte (Haan, Kahn et al. 1992). Dieser Extent wurde allerdings erst nach Akti-

vierung einer Funktion im Menü von *Intermedia* angezeigt. Solche *Link Extents* konnten sich auch überlappen; dies war im System zwar nicht erkennbar, es gab aber den Vorschlag, Überlappungen zukünftig mittels einer anderen Schattierung anzuzeigen (s. Abb. 185; Garrett, Smith et al. 1986).

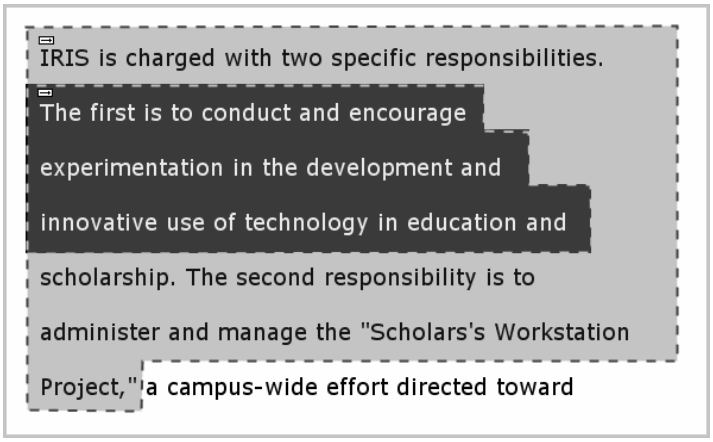


Abb. 185: Ein Konzept für überlappende Link-Extents in Intermedia (nach Garrett, Smith et al. 1986).

Eine weitere Innovation der Link-Anker in *Intermedia* war, dass jeder Anker zu mehreren Links gehören konnte, wodurch sich *multiple Links* ergaben. Nach dem Anklicken eines Link-Ankers, auf den mehrere Link-Objekte referenzierten, wurde dem Benutzer eine Dialog-Box mit den Link-Titeln zur Auswahl angeboten.

Für *Intermedia* existierten mehrere Client-Anwendungen, die unterschiedliche Übersichten des Hyperspace anboten, um die Navigation und Orientierung zu vereinfachen (Meyrowitz 1986). Dazu gehörten *lokale Karten* (mit den ein- und ausgehenden Links zu einem Knoten), *Konzeptkarten* und *globale Übersichtskarten*. Die globalen Karten wurden aber später entfernt, weil Benutzer sie als zu unübersichtlich kritisierten (Utting & Yankelovich 1989). Eine besondere Art der Übersicht bot der *WebView*, der neben einer Auflistung der zuletzt betrachteten Dokumente auch eine Vorschau auf die Objekte bot, zu denen vom aktuellen Dokument aus Links existierten. Dabei wurde für jedes Objekt ein Icon (für den Objekttyp) und der Titel dargestellt (s. Abschnitt 4.1.1, Abb. 24 auf S. 77 und Utting & Yankelovich 1989).

Mithilfe des *InterLex*-Servers verwies jedes Wort in *Intermedia*-Dokumenten zu einer Beschreibung im *American Heritage Dictionary*. Dies machte Intermedia zum ersten System mit *generischen Links* (Nielsen 1993a). Zusätzlich bot Intermedia *Guided Tours*, eine *History-Funktion* sowie eine *Volltext-Suche* mit einem Relevanz-Ranking der Suchtreffer an.

Trotz der Client-Server-Architektur war *Intermedia* als lokales verteiltes System für Arbeitsgruppen mit einer beschränkten Benutzerzahl konzipiert. Der Hauptgrund lag darin, dass es relativ schlecht skalierte. Der Schwerpunkt lag eher bei einer guten Umsetzung neuer Konzepte und Ideen, um die Benutzbarkeit und Nützlichkeit von Hypertext zu optimieren. Trotz seiner Fortschrittlichkeit erreichte *Intermedia* aber außerhalb des akademischen Bereiches nur geringe Popularität, zumal es für die damalige Zeit hohe (d. h. teure) Hardwareanforderungen stellte (Gillies & Cailliau 2001).

Intermedia war wohl seinerzeit das mit Abstand fortschrittlichste Hypertext-System und bot viele Hypertext-Features, die bis heute im Web nicht zu finden sind. Da allerdings das verwendete Betriebssystem A/UX ab Mitte der 1990er Jahre nicht weiterentwickelt wurde, läutete dies auch das Ende von Intermedia ein. Als Nachfolger von Intermedia wird oft *Storyspace*[18] erwähnt (Bernstein, Bolter et al. 1991), das aber bei weitem nicht die Möglichkeiten von *Intermedia* erreicht.

B.11 HyperCard: Ein flexibles Hypertext-Autorensystem

HyperCard wurde von Bill Atkinson für Apple Computer Inc. entwickelt und von 1987 bis 1992 kostenlos den Macintosh-Computern beigelegt.[19] Als erstes Hypertext-System ist es daher einer vergleichsweise großen Zahl von Benutzern zugänglich geworden. Die vielseitige Einsetzbarkeit von *HyperCard* machte es schnell sehr populär (Nielsen 1993a).

HyperCard verwendet wie *KMS* (s. Abschnitt B.4) und *NoteCards* (s. Abschnitt B.9) die Metapher der Karteikarte, um Informationen zu repräsentieren. Diese Karten hatten eine einheitliche Größe und konnten neben Text und Grafiken auch typische grafische Schnittstellenelemente wie Eingabefelder und Knöpfe enthalten. Jede Karte ließ sich zudem mit Variablen verknüpfen, wodurch das System auch für einfache Datenverarbeitungsaufgaben einsetzbar war. Zusammengehörige Karten eines Projekts waren jeweils in einer Struktur namens *Stack* zusammengefasst.

Link-Anker waren in *HyperCard* nicht Wörtern oder Objekten innerhalb einer Karte zugeordnet, sondern als *Regionen* auf der Karte definiert, den *Hot Buttons*. So konnte jedes dargestellte Element zu einer Verknüpfung werden. Zumeist haben die Autoren Link-Anker als Schaltflächen oder Kästen dargestellt, wobei ein Icon oder eine Beschriftung als Hinweis auf die Funktion und das Ziel des Links diente (s. Abb. 186). Zusätzlich wurden bei *Tastendruck* alle aktiven Link-Regionen mittels eines *Rahmens* hervorgehoben. Ein Link rief in *HyperCard* grundsätzlich ein Skript in der recht intuitiven Skriptsprache *HyperTalk* auf (Nichols 1987). Diese zumeist kleinen Programme definierten das Verhalten der HyperCard-Anwendung: Der Klick auf einen Link konnte somit nicht nur einfach zur Darstellung einer neue Karte führen, sondern auch Karten mit bestimmten Eigenschaften suchen, Änderungen an der aktuellen Karte abhängig von Eingaben und dem Systemzustand vornehmen oder ein externes Programm aufrufen (s. Tabelle 5; Goodman 1987; Nichols 1987).

Als Navigationshilfen bot *HyperCard* eine *History-* und eine *Suchfunktion* an. Innovativ war an der History-Funktion, dass sie die zuletzt angezeigten Karten in Form von *Thumbnails* darstellte (Weinreich 1997: 49).

---

[18] Storyspace wird von *Eastgate Systems Inc*. vertrieben: http://www.eastgate.com/storyspace/.

[19] Danach verkaufte Apple Computer *HyperCard* als eigenständige Software. Bis 2004 wurde es in der erweiterten Version 2.4 als Multimedia-Autorenwerkzeug vermarktet.

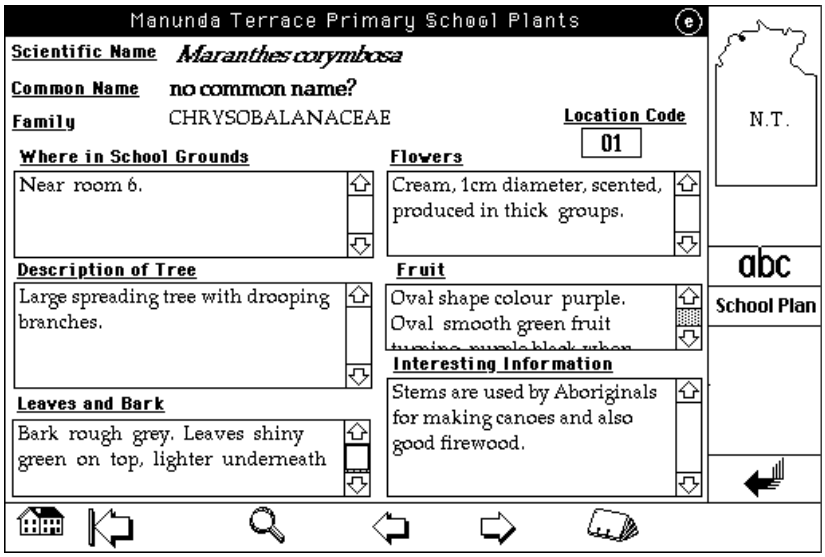


Abb. 186: Knöpfe und Icons als aktive Objekte in HyperCard.

HyperCard prägte aufgrund seiner Popularität zur damaligen Zeit das öffentliche Bild von Hypertext in ähnlicher Weise, wie es das Web jetzt für „das Internet" tut. Vielen Benutzern wurde daher der Eindruck vermittelt, dass alles, was mit *HyperCard* zu tun habe, auch Hypertext sei, und umgekehrt ein Hypertext-System die Funktionalität von *HyperCard* erfordere (McKnight, Dillon et al. 1991: 10). Aus Sicht vieler Hypertext-Experten handelte es sich bei *HyperCard* aber eher um ein recht universelles Prototyping-Werkzeug, das – basierend auf dem Konzept der Karte – die Realisierung interaktiver Systeme ermöglichte, als um ein wirkliches Hypertext-System (Nielsen 1993a). Hier kann eine weitere Parallele zum Web gezogen werden: Auch das Web war ursprünglich als Hypertext-System konzipiert und ist heute – ähnlich wie *HyperCard* damals – zu einer nahezu universellen Plattform für unterschiedlichste Anwendungen geworden.

B.12 Sun's Link Service: Hypertext als integrierbarer Dienst

Das Konzept *offener Hypertext-Systeme* wurde erstmals 1989 im Projekt *Sun's Link Service* realisiert. Das Projekt lief auf Workstations mit *SunOS* und dem *X Window System* und stellte Entwicklern einen Link-Datenbankserver sowie eine Reihe von Bibliotheken zur Verfügung, mit deren Hilfe nahezu beliebige Applikationen um eine Client-Funktionalität für den *Link Service* erweitert werden konnten (Pearl 1989). Somit handelte es sich bei *Sun's Link Service* – wie der Name bereits sagt – eigentlich nicht um ein Hypertext-System, sondern um einen Hypertext-Dienst. Die Integrationsmöglichkeit des Dienstes in existierende Anwendungen gewährleistete gleichzeitig die Unabhängigkeit der Links vom Dokumentformat, wodurch es die entscheidenden Schlüsselkriterien für *offene Hypertext-Systeme* erfüllte (s. Abschnitt 2.2.2.2).

Die anwendungsunabhängige Link-Datenbank des Dienstes war Voraussetzung für die Trennung der Verknüpfungsinformationen von den Dokumenten. Die Links waren relativ einfach modelliert: Jeder Link verfügte über zwei Anker, die jeweils eine Objekt-ID und

Daten über die aufzurufende Applikation zur Darstellung des Objekts enthielten. Weitere Attribute fehlten, Links hatten folglich weder eine Richtung noch waren sie typisiert. Die konkrete Realisierung der Link-Anker und Link-Marker sollte durch die Applikationen erfolgen. Die Objekt-ID zur Identifikation des Link-Ankers konnte infolgedessen nicht nur ganze Dokumente adressieren, sondern theoretisch auch Objekte innerhalb eines Dokuments beschreiben.

Sun's Link Service verfügte über zwei relativ primitive Konzepte zur Gewährleistung konsistenter Links: Sobald ein Benutzer einen Link auswählte, dessen zweiter Link-Anker nicht mehr zugeordnet werden konnte, wurde er auf den Fehler hingewiesen und gefragt, ob der Link gelöscht werden solle. Zusätzlich gab es eine Art *Garbage Collector*, der explizit aufzurufen war und (zeitaufwendig) für alle Links in der Datenbank testete, ob jeweils beide Anker noch existierten.

Die Benutzungsschnittstelle des Dienstes beschränkte sich auf das Wesentliche (s. Abb. 187): Mittels einer Dialog-Box konnte man Links verfolgen („*Display*"), neu erstellen („*Start Link*") oder auch wieder löschen („*Delete*"). Wenn das aktive Dokument mehrere Links bzw. einen Anker mit mehreren Link-Zielen aufwies, so wurde dem Benutzer beim Folgen des Links eine Auswahlbox angezeigt, in der er das gewünschte Link-Ziel selektieren konnte.

Für das Aussehen von Link-Markern in den einzelnen Anwendungen gab es keine Vorschriften. Nach (Pearl 1989) wurden lediglich zwei Empfehlungen gegeben: Erstens sollte die Erweiterung einer existierenden Anwendung durch den Sun Link Service möglichst wenig an dessen Benutzungsschnittstelle ändern:

> „One of the design goals for the Link Service was to have minimal impact on the look and feel of the integrating applications" (Pearl 1989).

Zweitens empfahl man aus *Konsistenzgründen* eine kleine Grafik mit zwei Kettengliedern als Link-Marker (s. Abb. 188 und Tabelle 5).

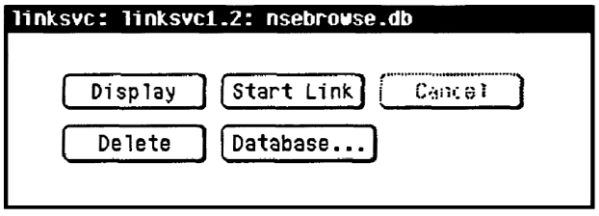


Abb. 187: Sun's Linking Command Panel (aus Pearl 1989).

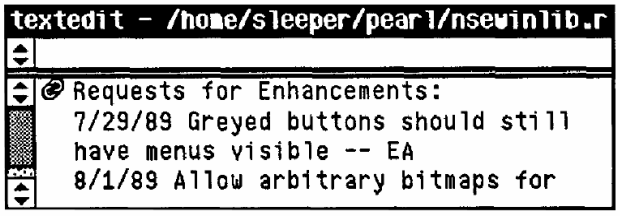


Abb. 188: Einbettung des Link Service in *textedit*. Links oben ist der Link-Marker zu sehen (Pearl 1989).

Sun's Link Service wurde in mehrere Text-Editoren, einen Dateibrowser, ein Projektverwaltungssystem und einige Programme von Drittherstellern integriert. Das System setzte sich aber nie durch.

B.13 Microcosm/DLS: Offener Hypertext, generische Links und multiple Link-Datenbanken

Ende der 1980er Jahre lies sich Wendy Hall von der University of Southampton (GB) durch das *Guide*-System (s. Abschnitt B.7) dazu inspirieren, ein neues Hypertext-Projekt zu initiieren. Im Gegensatz zu Guide sollte ihr System *Microcosm* aber kein eigenes Dateiformat benötigen, sondern all ihre vorhandenen Dokumente miteinander verknüpfen können und zusätzlich Verweise auf externe Quellen ermöglichen (Hall 1997).

Die ersten Publikationen zu *Microcosm* erschienen 1990, erste Demonstrationen und Bildschirmfotos aber erst einige Jahre später. *Microcosm* wurde für *Microsoft Windows 3* implementiert und in unterschiedliche Applikationen integriert, beispielsweise den *Windows Calendar*, *Microsoft Word* und später auch den *Netscape Navigator*.

Ähnlich wie *Sun's Link Service* konnte *Microcosm* Verweise zu beinahe beliebigen Objekten hinzufügen. Es bot zwei Formen der Integration:

- Wenn man Zugriff auf den Quellcode der Applikation hatte oder die Anwendung eine Makrosprache unterstützte, so konnte ein *Microcosm Aware Viewer* erstellt werden, der zusätzliche eigene Hypertext-Menüs aufwies und Link-Marker anzeigte.
- Andere Applikationen mussten auf den *Microcosm Universal Viewer* zurückgreifen, der im Titelbalken der Applikation zusätzliche Icons einblendete, mittels derer man Links folgen und neue Links erzeugen konnte (Davis, Knight et al. 1994).

Abb. 189 zeigt die Verwendung des *Universal Viewer* mit dem *Windows Calendar*. Das erste Icon (die Weltkugel) ruft das Hauptmenü von *Microcosm* auf (was gerade geschehen ist), das zweite Icon (das Kettenglied) dient dem Folgen von Links. Die weiteren Icons sind *optionale* Abkürzungen zu weiteren Hypertext-Funktionen von *Microcosm*.

Da bei *Microcosm* die zu verlinkenden Dokumente nicht geändert werden konnten, war eine Trennung von Links und Dokumenten mithilfe von Link-Datenbanken unumgänglich. Microcosm prägte hierfür den Begriff *Linkbase* (Davis, Hall et al. 1992). Link-Objekte der Linkbase hatten jeweils folgende Parameter: Der *Start* eines Links wurde über einen Schlüsselbegriff oder einen Text definiert und konnte zusätzlich über eine Positionsangabe genauer eingeschränkt werden. Optional konnte noch ein Dateiname (später auch einen URI) für das Dokument des Quellankers angegeben werden. Das *Link-Ziel* erforderte einen Dateinamen (bzw. einen URI) sowie eine Positionsangabe für den Zielanker. Drittens hatte jeder Link noch einen *Textkommentar*, der seine Funktion näher beschreiben sollte.

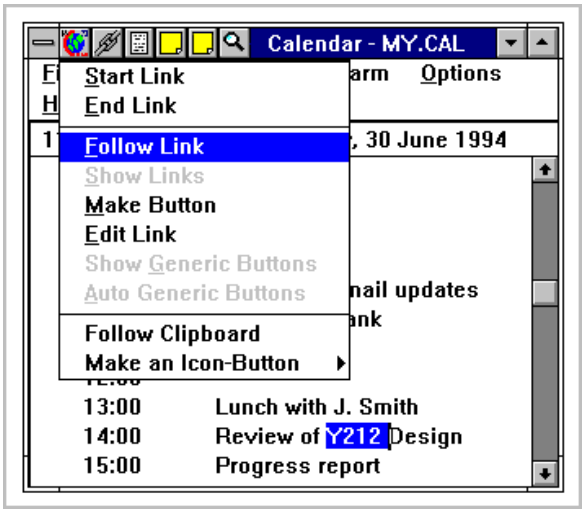


Abb. 189: Der Windows Calendar mit Microcosms Universal Viewer. Der Benutzer folgt gerade dem Link „Y212" (nach Davis, Knight et al. 1994).

Abhängig von der Art des Startankers wurden in *Microcosm* drei Arten von Links unterschieden:

1. *Generische Links* waren für alle Dokumente gültig, die einen entsprechenden Schlüsselbegriff als Startanker enthielten. Diese Anker wurden grundsätzlich *nicht* dargestellt, sondern Benutzer konnten ihnen folgen, indem sie einen Begriff im Text markierte und dann „Follow Link" auswählten (s. Abb. 189).

2. *Lokale Links* bezogen sich auf jedes Auftreten eines bestimmten Textes *innerhalb genau eines Dokuments*. Ansonsten waren sie vergleichbar mit generischen Links.

3. *Spezifische Links* bezogen sich auf einen genau definierten Anker in einem Dokument. Ihre Link-Anker wurden mittels eines Icons oder „*Link-Buttons*" dargestellt.

Für generische und lokale Links führte *Microcosm* den Begriff des *dynamischen Linkens* ein: Erst im Moment der Navigation wurde die Existenz eines Links in der Link-Datenbank überprüft und (wenn vorhanden) das Zieldokument aufgerufen. Gab es für einen Begriff oder eine Position in einem Dokument mehrere passende Links, so wurde dies als multipler Link interpretiert. Es erschien eine Auswahlbox mit den betreffenden Links, aus denen sich der Benutzer das gewünschte Ziel aussuchte (s. Abb. 190 und Tabelle 5).

Die ersten Versionen von *Microcosm* waren lokale Anwendungen und für einzelne Benutzer ausgelegt. Erst Version 2.0 bot eine Mehrbenutzer-Unterstützung an. Für eine verteilte Anwendung mit vielen Anwendern war es dennoch nicht geeignet, da es ebenso wie *Intermedia* Schwächen bei der Skalierbarkeit zeigte (Hall 1997).

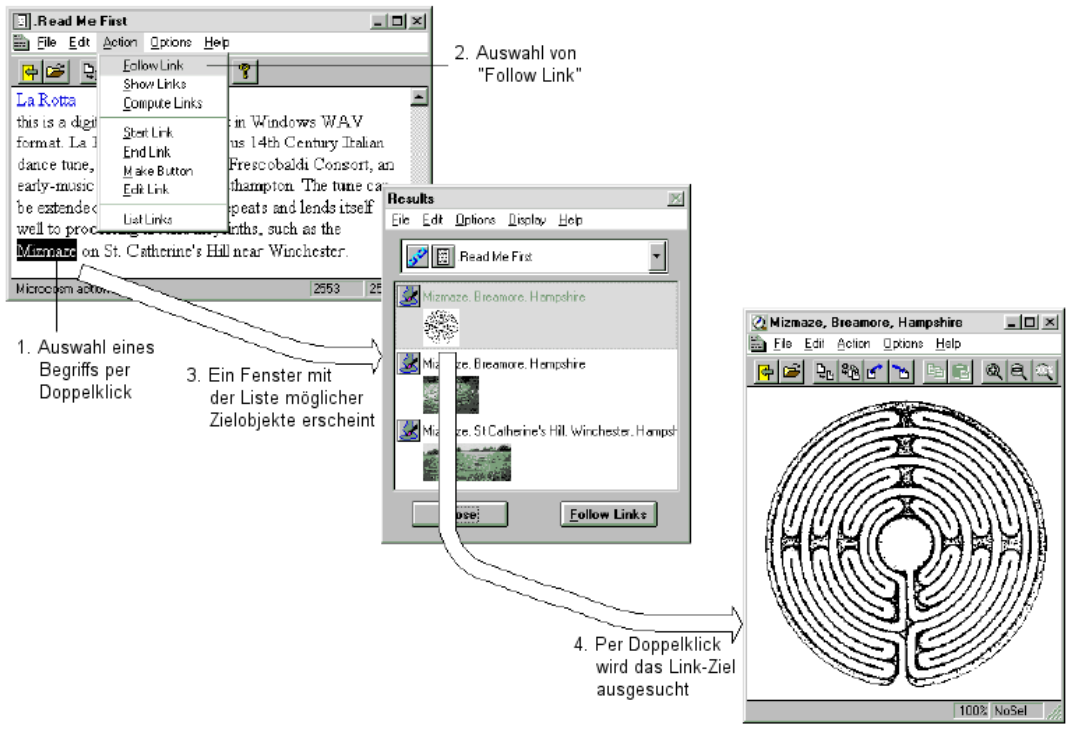


Abb. 190: Vorgehen bei der Auswahl multipler Links in *Microcosm DLS* (Mockup).

Anfang der 1990er Jahre erfuhr *Microcosm* in wissenschaftlichen Kreisen relativ viel Aufmerksamkeit, wurde aber ab 1995 vom World Wide Web in den Schatten gestellt. In der Folge wurde aus *Microcosm* der *Distributed Link-Service „DLS“*, der im Wesentlichen eine Anpassung der Konzepte von *Microcosm* an das Web war (Carr, DeRoure et al. 1995). *DLS* bot Benutzern mittels eines Intermediaries (s. Abschnitt 8.4) beim Browsen im Web ergänzende Verknüpfungen aus mehreren Link-Datenbanken an. Zur Darstellung der zusätzlichen „spezifischen Links“ in den *Webseiten* gab es unterschiedliche Methoden, beispielsweise als fortlaufende Zahl in eckigen Klammern, die hinter dem Link-Anker eingefügt wurde. DLS war allerdings kein System für die breite Masse, da es unter anderem relativ kompliziert zu installieren und konfigurieren war.[20]

B.14 Hyper-G: Ein offenes, verteiltes Hypertext-Content-Management-System

Ende 1989 nahmen sich Hermann Maurer und seine Gruppe von der Universität Graz vor, das „ultimative“ Hypertext-System zu entwickeln. Es sollte sowohl die Möglichkeiten eines *offenen* Hypertext-Systems bieten, als auch *mehrbenutzerfähig*, *verteilt* und *skalierbar* sein. Das

[20] Das *Microcosm*-Projekt hatte noch diverse kommerzielle Nachfolger, die aber mäßig erfolgreich waren: 1995 wurde die Multicosm Ltd. gegründet, die das System *Microcosm Plus* für den Einsatz mit interaktiven elektronischen Handbüchern entwickelte. Ab 1999 kam das Projekt *Microcosm Pro* hinzu, das eine Integration offener Hypertext-Funktionalität in webbasierte Systeme für das computerunterstützte Lernen erlaubte. Die letzte Entwicklung war *Microcosm TNG*, welches als flexible Middleware für offene Hypertext-Systeme fungierte (Hall 1997; Goose, Hall et al. 2000).

hieraus erwachsene Projekt nannten sie *Hyper-G*[21]. Es wurde in C++ als Client-Server-Anwendung für Unix-Plattformen realisiert. Die Client-Applikationen für das *X Window System* waren unter dem Namen *Harmony*[22] zusammengefasst.

Mauerers Gruppe hatte zuvor bereits umfangreiche Erfahrungen bei der Entwicklung des österreichischen „Videotex"-Informationssystems sammeln können (Maurer 1996). Dabei hatten sie gelernt, dass eine plattform- und dokumentenunabhängige Einsetzbarkeit eines solchen Systems die Trennung von Inhalt, Struktur und Präsentation voraussetzt und dass für eine brauchbare Suchfunktion der Einsatz von Metadaten unabdingbar ist (Gillies & Cailliau 2001). Dokumente und Links wurden daher getrennt, als eigene Objekte in einem Datenbanksystem gespeichert und jeweils mit zahlreichen Meta-Informationen versehen. Eine Benutzerverwaltung mit feingranularer Zugriffskontrolle erlaubte die Spezifikation von Lese- und Schreibrechten sowohl für Dokumente als auch für Hyperlinks. So konnten Benutzer z. B. das Recht erhalten, zu Dokumenten ohne Schreibzugriff Links und Annotationen hinzuzufügen (Kappe 1991).

Links waren in *Hyper-G* generell *gerichtet* und *bidirektional*, sodass sie vor- und rückwärts verfolgt werden konnten. Sie ließen sich mittels beliebiger beschreibender Schlüsselwörter wie „Kritik" oder „Beispiel" *typisieren*. Ein explizit vordefinierter derartiger Link-Typ war die *Annotation*. *Hyper*-G erlaubte die *Filterung* von Links nach Link-Typ und Benutzer, wobei die Link-Marker herausgefilterter Links nicht mehr erschienen (vergl. Abschnitt 3.1.6). Zur Wahrung der Link-Konsistenz besaß *Hyper-G* ein Link-Management, das mithilfe eines Server-zu-Server-Protokolls sogar die Konsistenz von Verknüpfungen mit externen Objekten gewährleisten konnte (Kappe 1995). Fehlerhafte Links wurden ausgeblendet.

Die Start- und Zielanker der Hyperlinks wurden zusammen mit den Links verwaltet. Link-Anker wurden erst bei der Darstellung eines Objekts aus der Link-Datenbank abgefragt. Sie konnten beliebige Teile eines Dokuments umfassen, und es war möglich, beliebige Dateiformate mit Verweisen zu versehen (s. Tabelle 5). Harmony bot entsprechende angepasste Client-Programme für PostScript, PDF, VRML und Grafiken.

Diese *Viewer* verwendeten unterschiedliche Methoden zur Anzeige von Link-Markern. Der *Text Viewer* für HTML- und HTF-Dokumente[23] hob Links mittels einer Änderung des Texthintergrunds hervor. Dabei konnten bis zu sechs *überlappende* Links dargestellt werden, indem unterschiedliche Hintergrundfarben in horizontale Streifen aufgeteilt wurden. Abb. 191 zeigt rechts ein Beispiel mit mehreren Link-Ankern im Fließtext des Text Viewers, wobei der Begriff „Graz" zwei überlappende Verknüpfungen aufweist. Link-Marker in Grafiken und Animationen wurden durch Linien dargestellt, die den Bereich des Link-Ankers umrahmten

[21] Das „*G*" in *Hyper-G* steht für Graz, die Heimatstadt des Systems (Maurer 1996).

[22] Es gab auch ein entsprechendes Pendant für Windows 3.11 mit dem Namen *Amadeus*, das aber in seinen Möglichkeiten weit hinter *Harmony* zurückblieb.

[23] HTF, das *Hypertext Format*, ist Hyper-G's SGML-Sprache zur Auszeichnung von Hypertext. Sie ähnelt HTML 1.0 weitestgehend.

(Andrews 1996). So ließen sich ebenfalls Link-Anker *innerhalb* von Bildern identifizieren (s. Abb. 191 unten). Dies bietet bis heute kein Webbrowser für *Image Maps*.

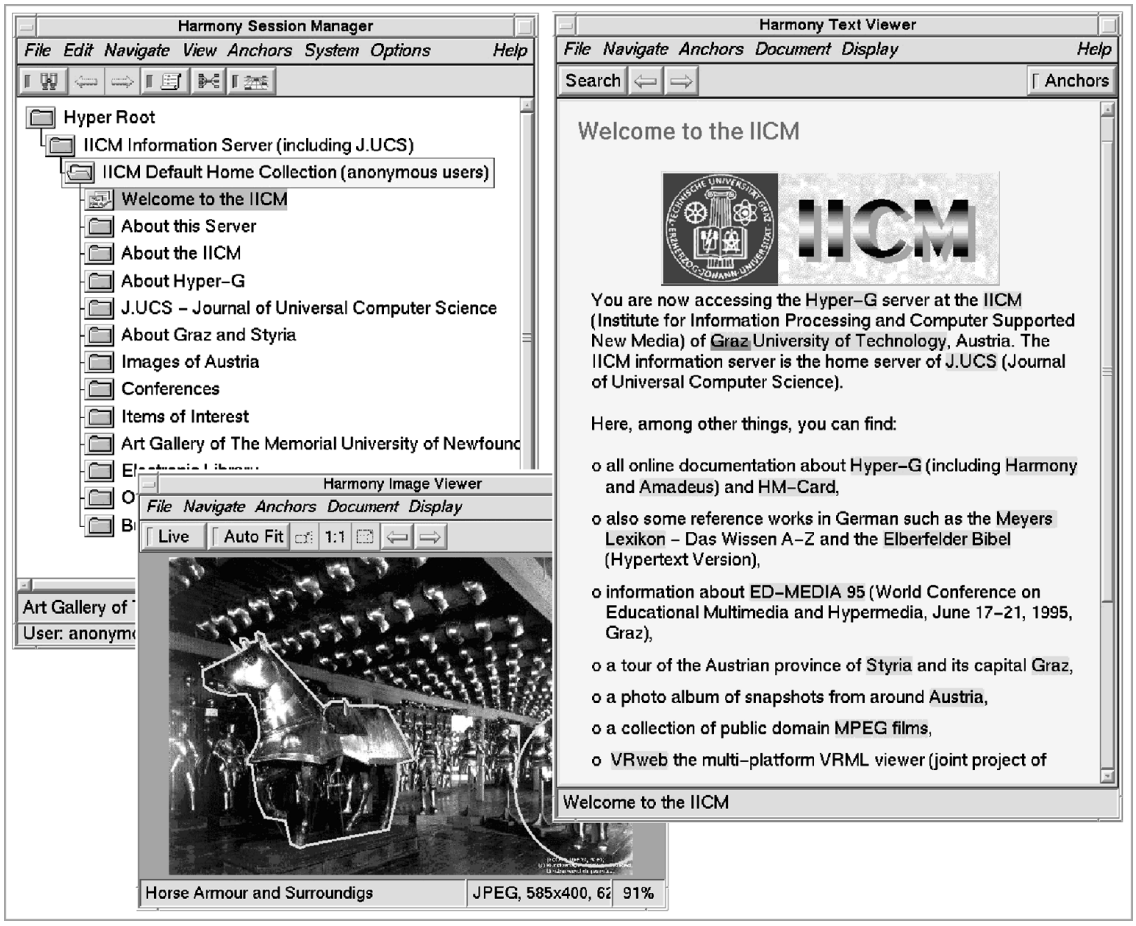


Abb. 191: Beispielansicht der Client-Anwendungen von Harmony (nach Andrews 1996).

Eine weitere Stärke von *Harmony* war die (wenn auch experimentelle) Darstellung von *Zielankern*: Im PostScript-Viewer von Harmony wurde „*Lowlightning*" eingesetzt, um den *nicht* zum Link-Ziel gehörenden Dokumentenbereich auszugrauen (s. Abb. 192).

Neben den umfangreichen Verknüpfungsmöglichkeiten verfügte *Hyper-G* über eine hierarchisch strukturierte Sitzungsverwaltung (den *Session Manager*, s. Abb. 191 links oben), in der alle Objekte[24] eines Servers sogenannten *Kollektionen (Collections)* zugeordnet waren. Dieser hierarchische[25] Gliederungsmechanismus strukturierte alle Informationen eines Servers. Benutzer hatten dadurch jederzeit Zugriff auf eine inhaltlich strukturierte Übersicht aller Inhalte einer Site (Kappe 1991).

[24] Ein weiterer elementarer Strukturierungsmechanismus zum Gruppieren inhaltlich voneinander abhängiger Dateien zu eigenen Objekten waren „*Cluster*". So konnten beispielsweise polyglotte Versionen eines Dokuments und ihre eingebetteten Grafiken zusammengefasst werden. Dies verbesserte die Übersichtlichkeit der Gesamtstruktur eines Servers signifikant (Kappe 1991).

[25] Die Struktur war zwar hierarchisch, aber nicht zwangsweise baumförmig, sondern konnte durchaus ein Graph sein, da Kollektionen und Objekte bewusst *mehreren* Kollektionen zugeordnet werden konnten.

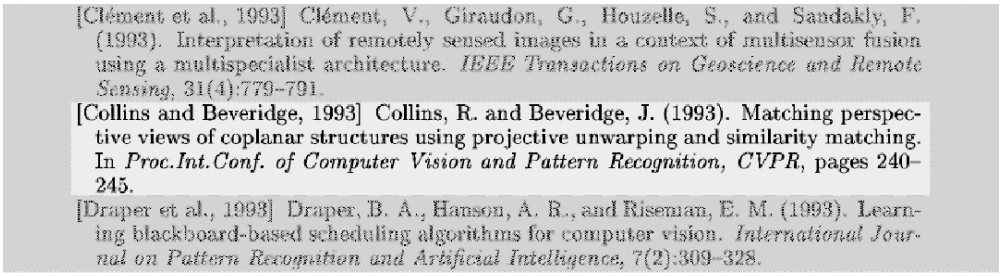

[Clément et al., 1993] Clément, V., Giraudon, G., Houzelle, S., and Sandakly, F. (1993). Interpretation of remotely sensed images in a context of multisensor fusion using a multispecialist architecture. *IEEE Transactions on Geoscience and Remote Sensing*, 31(4):779–791.
[Collins and Beveridge, 1993] Collins, R. and Beveridge, J. (1993). Matching perspective views of coplanar structures using projective unwarping and similarity matching. In *Proc.Int.Conf. of Computer Vision and Pattern Recognition, CVPR*, pages 240–245.
[Draper et al., 1993] Draper, B. A., Hanson, A. R., and Riseman, E. M. (1993). Learning blackboard-based scheduling algorithms for computer vision. *International Journal on Pattern Recognition and Artificial Intelligence*, 7(2):309–328.

Abb. 192: Der *Zielanker* eines Links in einem Postscript-Dokument war nicht ausgegraut (aus Andrews 1996: 62).

Zur weiteren Vereinfachung von Navigation und Orientierung bot *Harmony* unterschiedliche zwei- und dreidimensionale grafische Visualisierung von Strukturinformationen an. Da sämtliche Struktur-Daten des Servers unabhängig gespeichert wurden, konnte beispielsweise schnell eine lokale Karte des Hyperspace oder eine dreidimensionale Ansicht der hierarchischen Gliederung des Servers präsentiert werden.

Jeder *Hyper-G-Server* bot darüber hinaus ein integriertes Suchsystem, das sowohl die Metadaten aller Objekte, als auch den Volltext aller unterstützten Dokumentformate wie HTF, PDF und Postscript berücksichtigte. Der Index wurde aufgrund der Systemarchitektur automatisch konsistent gehalten.

Obwohl *Hyper-G* dem Web konzeptionell deutlich überlegen war, konnte es sich nicht durchsetzen. Hierfür gibt es drei Hauptgründe: Zum einen war die Performanz der verfügbaren Clients zur Anzeige der Dokumente deutlich schlechter als die vergleichbarer Webbrowser (Weinreich 1997). Zweitens war die Installation von *Hyper-G*-Servern und Clients erheblich aufwendiger, und drittens hatte das Web bei Praxisreife von *Hyper-G* bereits einen so großen Bekanntheitsgrad erlangt, dass es gegen den „Hype“ um das Web nicht ankam.

Aus letzterem Grunde versuchten die Entwickler von *Hyper-G* ihr System bereits 1996 als die *„zweite Generation des Web“* anzupreisen (Maurer 1996), da es über eine HTTP-Schnittstelle verfügte, mittels derer man auch mit Webbrowsern auf die Informationen eines *Hyper-G*-Servers zugreifen konnte. Dem Benutzer boten sich dabei aber bei Weitem nicht die gleichen Möglichkeiten wie mit *Harmony*, beispielsweise wurde die Position in der Hierarchie des Servers im Webbrowser unterhalb des aktuellen Dokuments eingeblendet, und es fehlten jegliche Übersichtskarten.

Da das Web seinen Siegeszug fortsetzte, beschloss Frank Kappe – einer der Projektleiter von *Hyper-G* – 1997 mit den gesammelten Erfahrungen und dem erlangten Renommee das kommerzielle Web-Content-Management-System *Hyperwave* zu entwickeln.[26] Es wurde lange als Nachfolger von *Hyper-G* anpriesen und wird bis heute erfolgreich vertrieben.

[26] Die Homepage der Hyperwave AG findet man unter: http://www.hyperwave.com/.

B.15 Das Web: verteilt, skalierbar, universell

Das ab 1989 von Tim Berners-Lee (s. Abschnitt A.4) am CERN in Genf entwickelte *World Wide Web* erscheint vor den Möglichkeiten von *Intermedia* und *Hyper-G* geradezu rückständig: Es benötigt ein eigenes Dokumentenformat, die Links sind in den Code eingebettet, sie sind uni-direktional und konnten leicht inkonsistent werden, wenn man ein Dokument verändert, verschiebt oder löscht (vergl. Abschnitt 3.3.4). Ursprünglich waren bei Web auch keine eingebetteten Grafiken vorgesehen, dafür aber erlaubte der erste Browser das einfache Bearbeiten und Erstellen von Seiten und Links (s. u.).

Dem konnte es einige Stärken entgegensetzen, die andere Systeme dieser Zeit zum Teil vermissen ließen: Es war plattformunabhängig, skalierte auch auf globaler Ebene, war komplett dezentralisiert und daher fehlertolerant (Bry & Eckert 2005) und als Open-Source-Projekt kostenlos erhältlich. Ein weiteres Alleinstellungsmerkmal war die Möglichkeit, mit *externen Links* zu Ressourcen auf anderen Servern zu verweisen. Dies funktionierte ohne eine zentrale Link-Datenbank, die diese Verknüpfungen verwaltete, wodurch Tim Berners-Lee die Skalierbarkeit des Systems sicherstellte (s. Abschnitt 3.3.4).

Zur Adressierung von Objekten (als Link-Ziel) entwickelte Tim Berners-Lee den *URI* (*Universal Resource Identifier*), den er selbst als die bedeutendste seiner Erfindungen im Zusammenhang mit dem Web bezeichnete, da sie es erstmals ermöglichte, beliebige Ressourcen im Internet zu referenzieren – und das nicht nur global im Web, sondern auch in anderen Systemen (Berners-Lee & Fischetti 2000). Gleichzeitig konnte im Zusammenhang mit dem von ihm entwickelten Web-Protokoll *HTTP (Hypertext Transfer Protocol)* jedes Dateiformat als auch beliebige Daten von und zu Online-Applikationen transferiert werden.

Der erste Browser für das Web war *Nexus*, den Berners-Lee im Februar 1991 am CERN vorstellte. Das Programm war aber nur für die NeXTStep-Plattform verfügbar, da es die objektorientierten Bibliotheken des Systems voraussetzte. Er verfügte über einen Editor, mit dem man einfach lokale Dateien im HTML-Format bearbeiten und mit Links versehen konnte. Links wurden unterstrichen dargestellt, auf späteren Farb-Workstations hob *Nexus* sie blau und unterstrichen hervor (s. Abb. 193). Das Ziel eines Links erschien wie bei *NoteCards* in einem neuen Fenster (s. Abschnitt B.9). Der Browser verfügte über ein kleines Fenster mit „Navigationselementen“ (s. Abb. 193 rechts oben), mit denen man zur vorherigen Seite zurückkehren konnte („Back up“) und im aktuellen Fenster den vorherigen und nächsten Link der *Ausgangseite* aufrufen konnte („Previous“ und „Next“), ohne dass man zu ihr zurückkehren musste (Berners-Lee 2010).

Für das World Wide Web sah Tim Berners-Lee von Beginn an eine mit Bookmarks vergleichbare Navigationshilfe vor: Jeder Benutzer sollte Links zu interessanten Dokumenten mit dem Editor von *Nexus* in einer persönlichen Webseite – seiner „Homepage“ – verwalten können. Er nannte dieses Konzept „Private Links“ (Berners-Lee 1989; Berners-Lee 1998b).

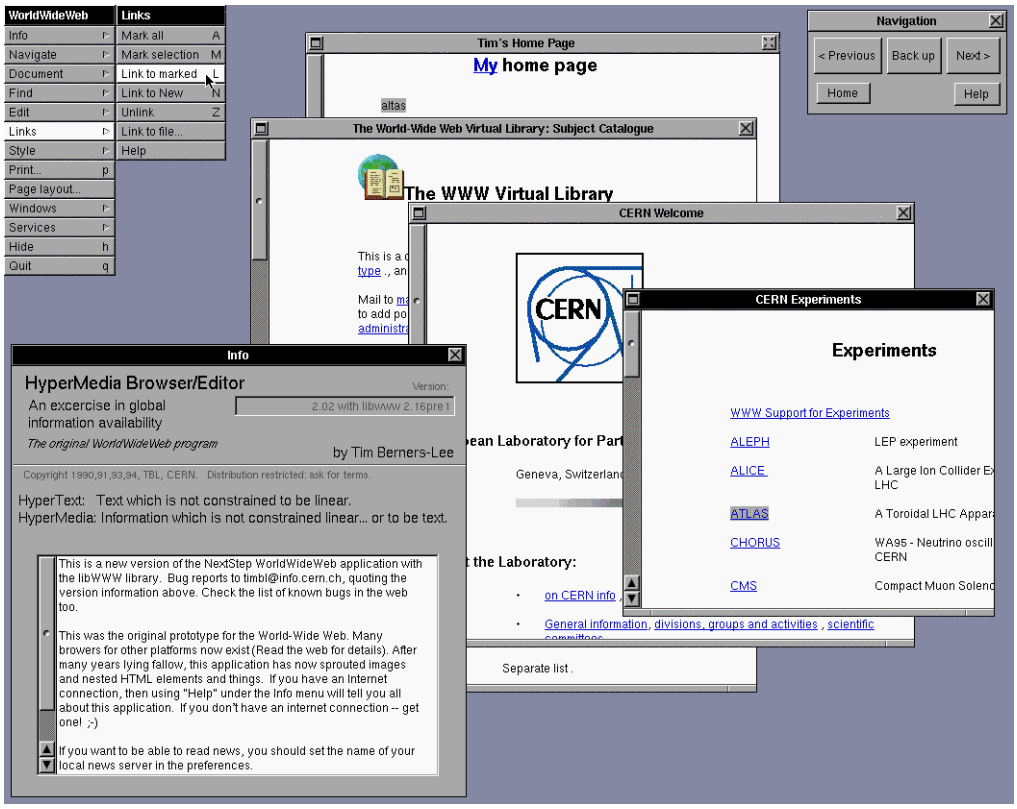


Abb. 193: Der Browser Nexus auf dem NeXTStep-System. Zu sehen ist Version 2 aus dem Jahr 1994 (nach CERN 2008).

Damit das Web auch für Benutzer ohne NeXTStep-Workstation zugänglich wurde, entwickelte Nicola Pellow am CERN zum Frühjahr 1992 die erste Version des *Line-mode Browsers*, der für Text-Terminals konzipiert war. Der Browser kennzeichnete Links durch eine fortlaufende Nummer in eckigen Klammern hinter dem Link-Text (s. Abb. 194), die Auswahl des Links erfolgte durch die Eingabe der Nummer (CERN 2008).

Die ersten Browser für das *X Window System* für Unix Workstations waren *Erwise* und *ViolaWWW*. *Erwise* wurde an der *Universität Helsinki* als studentisches Projekt entwickelt und im April 1992 vorgestellt, danach aber nicht weiterentwickelt. Einen Monat später veröffentlichte Pei Wei, ein Student der University of California, Berkeley, die erste Version seines Browsers *ViolaWWW* (Berners-Lee & Fischetti 2000: 56). Bereits im Jahr 1993 verfügte *ViolaWWW* über eine eingebettete Skript-Sprache (ähnlich wie später *JavaScript* im *Netscape Navigator*, s. u.), Style Sheets (vergleichbar mit *Cascading Style Sheets*) und einen Mechanismus, der das Ausführen von Applikationen im Browser über das Internet erlaubte, wie es mit *Java Applets* erst Jahre später für andere Browser möglich wurde. *ViolaWWW* setzte zu seiner Zeit den Standard für Webbrowser sowohl in technischer Hinsicht als auch bezüglich der Benutzungsschnittstelle (Berners-Lee & Fischetti 2000; Freedman 2001: 629).

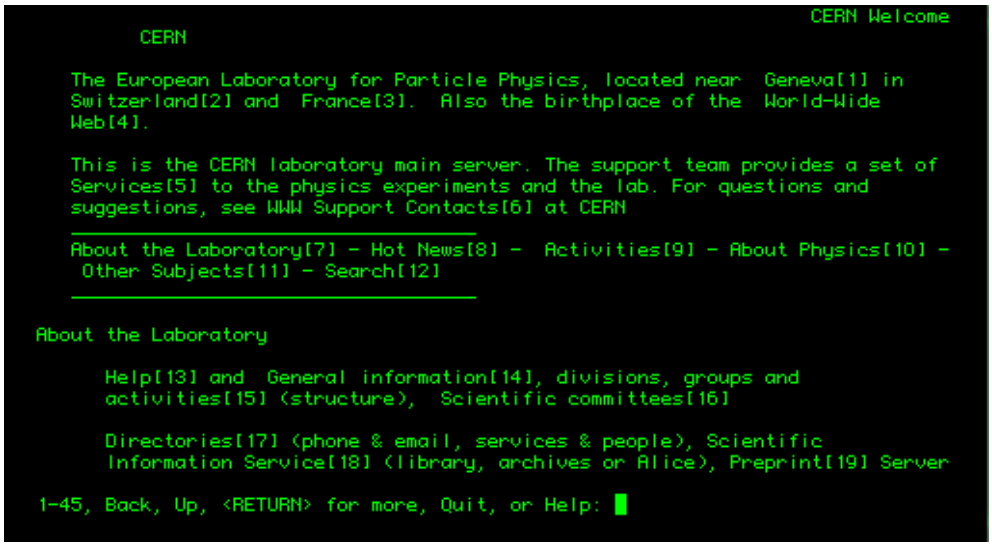


Abb. 194: Der *Line-mode Browser* von Nicola Pellow war anfänglich der einzige verfügbare Browser für viele Benutzer des Web (Quelle: CERN 2008).

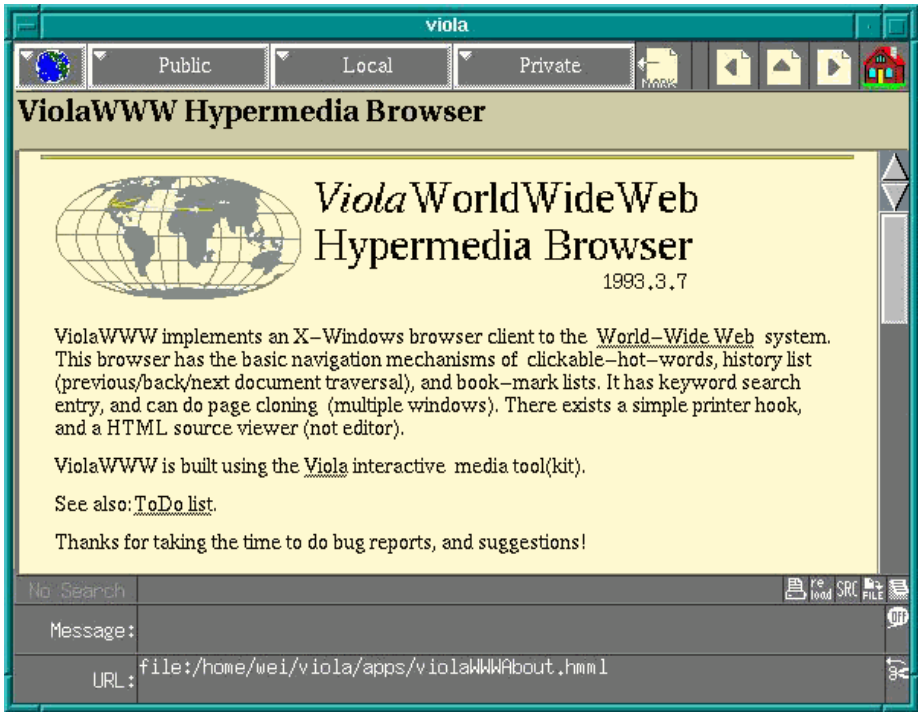


Abb. 195: Ein Screenshot von ViolaWWW. Die dargestellte Version ist aus dem Frühjahr 1993.

Als Link-Marker dienten bei *ViolaWWW* wellenförmige Unterstreichungen (Abb. 195). Der Titel des Dokuments erschien nicht wie bei Nexus und heutigen Browsern in der Titelleiste des Fensters, sondern in einem reservierten Bereich direkt über dem Dokument (vergl. Abb. 195 oben: „*ViolaWWW Hypermedia Browser*“). Der URI der aktuellen Seite wurde zusätzlich ganz unten im Fenster dargestellt (s. Abb. 195). Die Navigationsmöglichkeiten waren vergleichbar mit Nexus, zusätzlich konnte man noch Bookmarks setzen und über eine History auf zuvor besuchte Seiten zugreifen. Der Browser verfügte aber über keine Bearbeitungsfunktion für Webseiten mehr (Connolly, Cailliau et al. 2000; Gillies & Cailliau 2001).

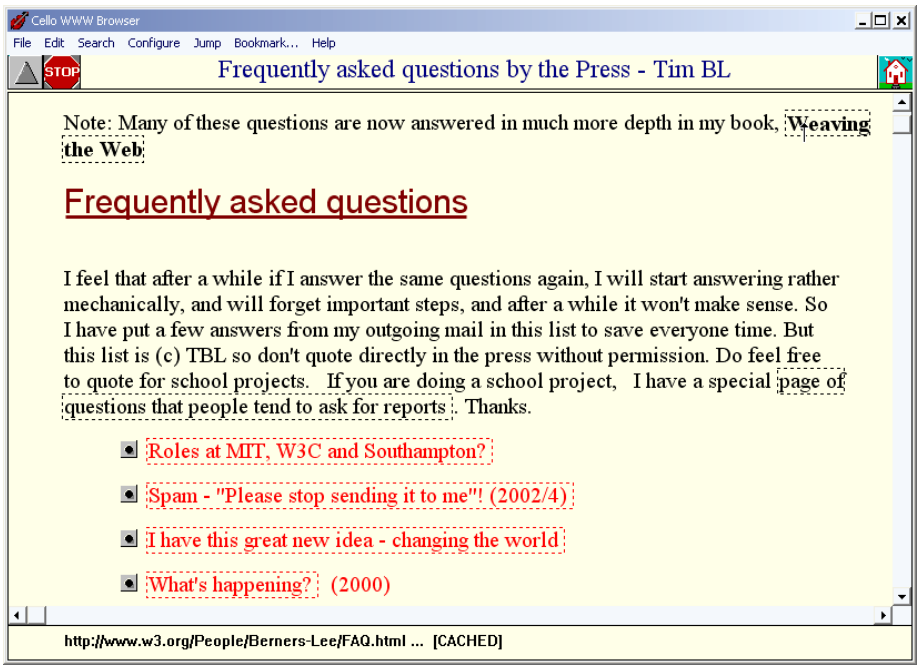


Abb. 196: Eine Webseite im Browser Cello (Version 1.01 aus dem Jahr 1994).

Der erste Browser für Microsoft Windows hieß *Cello*. Er wurde im März 1993 von Thomas R. Bruce von der Cornell University in Ithaca (New York, USA) veröffentlicht (Berners-Lee & Fischetti 2000). Der Browser verwendete Farben und Unterstreichungen für unterschiedliche HTML-Elemente wie Überschriften und Listen. Link-Anker wurden stattdessen durch gepunktete Rahmen hervorgehoben (s. Abb. 196). Er bot ebenfalls einen Backtracking-Knopf mit der Bezeichnung „Up“. Ähnlich wie bei *ViolaWWW* wurde der Titel der aktuellen Seite über und der URI unter dem Dokument dargestellt.

Im Folgemonat April 1993 kam Version 1.0 von *Mosaic* für das „X Window System“[27] heraus, und im September 1993 folgten die ersten Beta-Versionen von *Mosaic* für Windows und Macintosh-Computer. *Mosaic* entwickelte sich innerhalb weniger Wochen zum Standard-Browser für die jeweilige Plattform (Berners-Lee & Fischetti 2000). Er ließ sich einfacher installieren als andere Browser[28] und bot als erster die Möglichkeit, Grafiken direkt in Webseiten einzubetten. Grafiken konnten auch als Link-Anker fungieren und wurden dann durch einen blauen Rahmen hervorgehoben (s. Abschnitt 5.2.9). Ebenfalls neu war eine einfache „Link-Vorschau“, bei der unten im Statusbereich des Browsers der URI des Link-Ziels erschien, wenn man mit dem Mauszeiger über einen Link-Anker fuhr (s. Abb. 55 auf Seite 151 und Abb. 197 unten).

[27] Die ersten beta-Versionen von Mosaic waren bereits ab Januar 1993 erhältlich.

[28] Unter X Windows musste beispielsweise nur eine einzige Datei installiert werden, wogegen frühere Browser recht aufwendig zu installieren waren und zusätzliche Bibliotheken benötigten.



Abb. 197: NCSA Mosaic für Windows. Zu sehen ist Version 0.4 vom September 1994.

Links wurden in *Mosaic* blau und unterstrichen hervorgehoben und änderten die Farbe (je nach Browser-Version in Lila oder Grün, dies war zudem konfigurierbar, s. Abschnitt 5.1.3), wenn ein Link zu einer kürzlich bereits besuchten Seite führte (s. Abb. 197). Da *Mosaic* keinen integrierten Editor für HTML-Seiten bot, stand als Alternative eine „Hotlist“ zur Verfügung, die den heutigen *Bookmarks* bzw. *Favoriten* entsprach (s. Abschnitt 3.1.7). Zahlreiche unterschiedliche Versionen des Open-Source-Browsers *Mosaic* wurden in den Folgejahren entwickelt, beispielsweise eine X-Windows-Version mit einer grafischen History, die Thumbnails für die Repräsentation besuchter Webseiten verwendete (Ayers & Stasko 1995).

Version 1.0 des Browsers *Netscape Navigator* wurde im Dezember 1994 veröffentlicht und entwickelte sich schnell zum neuen Standard-Browser. Die Schnittstelle ähnelte sehr der von *NCSA Mosaic* (s. Abb. 198), zumal im Wesentlichen dasselbe Team die Entwicklung leitete. Er war wesentlich schneller als alle vorherigen Browser, da er gleichzeitig mehrere HTTP-Verbindungen zu einem Webserver aufbauen konnte, um beispielsweise parallel die eingebetteten Bilder einer Seite zu laden, während die Seite selbst noch in der Übertragung war. Ab Version 2.0 bot der *Netscape Navigator* eine in den HTML-Code eingebettete Scriptsprache, die erst *LiveScript* und später *JavaScript* hieß und erstmals interaktive Webseiten ohne Server-Kommunikation ermöglichte. Mit JavaScript wurden viele der erweiterten Link-Möglichkeiten aus früheren Hypertext-Systemen im Web realisierbar.



Abb. 198: Der *Netscape Navigator 1.0* aus dem Jahr 1994. Zu sehen ist die „Welcome Page", die beim Start des Browsers erscheint.

Die ersten Versionen des *Microsoft Internet Explorers* bauten ebenfalls auf *Mosaic* auf und unterschieden sich in der Benutzungsschnittstelle nur unwesentlich. Eine Neuerung bezüglich der Schnittstelle von Hyperlinks wurde in Version 4.0 des *Internet Explorers* Ende 1997 eingeführt: Der Browser zeigte den Inhalt des „`title`"-Attributs von Link-Ankern in einem Tooltip an (Nielsen 1998a; Microsoft 2011) und machte damit erstmals Typ-Informationen von Links für die Benutzer des Web zugänglich (s. Abschnitt 5.3.7, Abb. 58 und Tabelle 5).

Erwähnenswert bezüglich der Weiterentwicklung der Benutzungsschnittstelle von Webbrowsern ist der Browser *Opera*, der als erster[29] das *Tabbed Browsing* integrierte, bei dem innerhalb *eines* Fensters in „Tabs" (Reitern) mehrere Dokumente geöffnet werden konnten (Abb. 199, oben). Zudem führte *Opera* ab Version 5.1 „Mausgesten" an, die es erlauben, den Browser durch Mausbewegungen bei gedrückter rechter Maustaste zu steuern. Als weitere Besonderheit stellte *Opera* den URI des Link-Ziels zusätzlich im Tooltip dar (Abb. 199, rechts).

Im November 2004 wurde Version 1.0 von *Mozilla Firefox* veröffentlicht. *Firefox* wurde schnell zum populärsten Open-Source-Browser für das Web, auch wenn ihm seit 2009 *Google Chrome* zunehmend Marktanteile abnimmt. Eine besondere Stärke von *Mozilla Firefox* sind *Browser Add-Ons*, mit denen sich die Benutzungsschittstelle von Firefox nahezu beliebig erweitern lässt (vergl. Abschnitte 8.3.1 und 8.9). Inzwischen gibt Tausende solcher *Add-Ons*; einige bieten sogar ähnliche Konzepte wie sie HyperScout-Projekt realisiert und evaluiert wurden (s. Kapitel 5.5ff, 6.4ff und 9.2.3).

[29] Die erste *Opera*-Version mit Browser Tabs (auch „Multi Document Interface"/"MDI" genannt) war Opera 4.0 im Jahr 2000. Es gab aber vorher schon Erweiterungen für den Internet Explorer, die dies ermöglichten, beispielsweise *NetCaptor* von 1998.

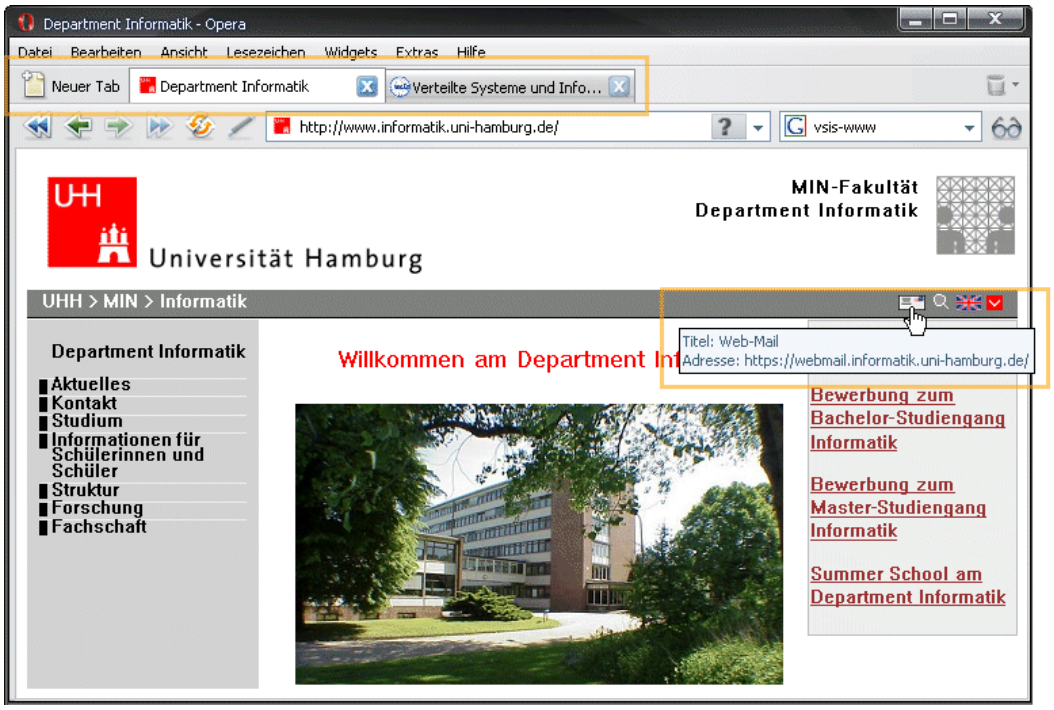


Abb. 199: Opera bot als erster Browser Tabs an (oben) und zeigt im Tooltip neben dem Titel immer den URI des Link-Zieles (rechts).

B.16 Weitere Hypertext-Systeme

Zahlreiche weitere Hypertext-Systeme konnten in dieser Übersicht nicht berücksichtigt werden oder wurden bewusst ausgegrenzt, da sie entweder nicht dem assoziativen Hypertext-Paradigma zuzurechnen sind (s. Kapitel 2.2ff) oder sie aus Sicht des Autors keine wesentlichen neuen Aspekte bezüglich der Möglichkeiten und der Benutzungsschnittstelle von Links und Link-Ankern beitragen. Zwei Beispiele:

- Die ab 1977 am MIT entwickelte *Aspen Movie Map* wird oft als erstes Hypermedia-System angesehen (Nielsen 1993a). Benutzer hatten mit dem System die Möglichkeit, eine virtuelle Tour[30] durch die Stadt Aspen (Colorado, USA) zu unternehmen, vergleichbar mit *Google Street View*[31] heute (Naimark 1997). Es verwendete Aufnahmen aus den Straßen der Stadt, zwischen denen man mittels eines Touchscreens virtuell navigieren konnte. Das System bot aber keine assoziativen Querverweise zwischen einzelnen Knoten.
- Das System *Aquanet* stellte inhaltliche Zusammenhänge durch die Anordnung der Knoten auf einer Karte dar und nicht durch explizite Hyperlinks. Es ist daher dem räumlichen Hypertext-Paradigma zuzuordnen (s. Abschnitt 2.2).

[30] Ein Beispiel-Video des Systems findet man unter http://www.youtube.com/watch?v=Hf6LkqgXPMU

[31] *Street View* ist ein Dienst von *Google*, der für viele Städte und Nationalparks 360°-Panoramabilder der Straßen anbietet, die im Abstand von einigen Metern aufgenommen werden. Man erhält eine Rundum-Ansicht der fotografierten Örtlichkeiten. Mehr Informationen findet man unter: http://maps.google.de/help/maps/streetview/.

Für zusätzliche Informationen über die grafischen Benutzungsschnittstellen zahlreicher Hypertext-Informationssysteme sei auf die Arbeit von Matthias Müller-Prove (Müller-Prove 2001) und die in den Abschnitten aufgeführten Referenzen verwiesen, insbesondere (Conklin 1987; Nyce & Kahn 1991; Nielsen 1993a).

B.17 Resümee

Dieses Kapitel gibt einen Eindruck davon, wie unterschiedlich die Konzepte und Realisierungen früherer Hypertext-Systeme waren. Sie boten nicht nur verschiedene Navigationswerkzeuge, auch die Eigenschaften von Knoten, Links, Link-Ankern und Link-Markern variierten wesentlich (s. auch Tabelle 5 auf Seite B-41).

Als einziges der vorgestellten Systeme ist das World Wide Web bis heute in Benutzung, obwohl es wesentliche Defizite gegenüber seinen Vorgängern aufweist. So gibt es außer dem *Dokumenttitel* keinen allgemeingültigen Standard für weitere *Metadaten*, keine systemimmanenten *Strukturinformationen*, kein integriertes *Suchsystem* und nur sehr primitive *History-Funktionen*. Solche Basisfunktionalität früherer Hypertext-Systeme erfordert im Web bis heute externe Dienste oder zusätzliche Werkzeuge (s. Kapitel 3.1ff und 3.4). Des Weiteren haben Web-Hyperlinks diverse Einschränkungen, wie ihre Unidirektionalität, die mangelnde Konsistenzsicherung und die Notwendigkeit, sie in den Code der Dokumente einzubinden (s. Abschnitte B.15 und 3.3.4). Indes werden inzwischen viele der erweiterten Link-Möglichkeiten früherer Systeme im Web mithilfe von JavaScript- und Flash-Programmen realisiert, wie *multiple Links*, *generische Links* oder *Stretchtext* (s. auch Abschnitte 4.4.6 und 4.5.1.3), ohne dass für diese Anwendungen neue Konzepte für Link-Marker etabliert wurden (s. Abschnitt 5.1.3). Aufgrund der wachsenden Möglichkeiten von JavaScript und serverseitiger Web-Dienste ist das Web zudem in vielen Anwendungsbereichen gleichzeitig zu einer *Online-Applikation* geworden, die zusätzliche Herausforderungen an die Benutzungsschnittstelle stellt (vergl. Kapitel 2.2.5 und 9.2).

Dieser Überblick zeigt darüber hinaus, dass keine sukzessive Entwicklung eines Standards für Link-Marker erkennbar ist, sondern für fast jedes System eine andere Methode gewählt wurde (s. auch Tabelle 5), ohne dass eine systematische Einordnung und Evaluation der Methoden stattgefunden hätte (Weinreich & Lamersdorf 2000; Obendorf & Weinreich 2003). Kapitel 5.2 dieser Arbeit klassifiziert daher erstmals die existierenden Techniken zur Darstellung von Link-Ankern und diskutiert die jeweiligen Vor- und Nachteile.

Bezüglich der Anzeige von Link-Previews und Typ-Informationen für Links ist die Situation ebenfalls unbefriedigend: Viele Systeme – selbst solche mit typisierten Links wie *TextNet* und *Hyper-G* – zeigten die Link-Typen gar nicht oder nur auf Anforderung an. Eine systematische Untersuchung der existierenden Techniken geschah ebenfalls nicht. Kapitel 5.3 führt aus diesem Grund eine Klassifikation der Möglichkeiten zur Darstellung von Link-Informationen in Hypertext-Systemen durch und analysiert die jeweiligen Stärken und Schwächen.

Ausgehend von diesen Analysen der Link-Benutzungsschnittstelle wurden die Konzepte des HyperScout-Systems entwickelt (s. Kapitel 5.5ff).

Der Vielgestaltigkeit früherer Hypertext-Informationssysteme hat zur Entwicklung mehrerer Hypertext-Standards geführt. Sie führten das Fachwissen der Hypertext-Forscher zusammen und sollten als Basis für zukünftige Entwicklungen dienen. Der Autor dieser Arbeit hatte daher die Hoffnung, dass diese Standards auch Hinweise und Emüfehlungen für die Realisierung der Schnittstelle der Hyperlinks böten. Der folgende Abschnitt stellt daher die wichtigsten Hypertext-Standards vor und richtet ein besonderes Augenmerk auf ihre Spezifikation von Links und Link-Ankern.

Tabelle 5: Charakteristische Eigenschaften bedeutender Hypertext-Informationssysteme

System	Kennzeichnende Innovationen	Art der Knoten	Kategorien von Links	Link-Direktionalität	Grad der Links	Link-Typisierung	Link-Granularität	Link-Marker	Link-Preview	Navigationswerkzeuge
Memex *Seite B-1*	Konzept der assoziativen Pfade	Beliebige Dokumente auf Mikrofilm	Trails (Pfade)	Gerichtet	Multiple Verknüpfungen	Benannte Verknüpfungen	Quelle: Knoten Ziel: Knoten	Separat unterhalb des Dokuments	Nein	k.A.
NLS/ Augment *Seite B-3*	Kollaborative Hypertext-Nutzung, Video-Konferenzen, Maus, Chord Keyset	„Statements" mit bis zu 2000 Zeichen Text	Assoziative Hyperlinks mit verschiedenen Link-Aktionen	Unidirektional	Binär	Nein	Q: Inline Z: Knoten	Eingebetteter Code in spitzen Klammern: <OAD, 2250, 7c : ebtzgm>	Codierte Ang. zum Link-Ziel und zur Link-Aktion	Hierarchie
HES/ Fress *Seite B-5*	Bidirektionale Links, integrierter Editor mit Undo-Funktionalität, Annotations-Funktion	Text-Dokumente	2 Kategorien: *Tags*: Für Anmerkungen und Fußnoten *Jumps*: Verknüpfen Dokumente	*Tags*: Unidirektional *Jumps*: Bidirektional, gerichtet	*Tags:* Binär *Jumps*: Binär	*Tags*: Nein *Jumps*: Frei typisiert (Keywords, Anker-Titel, Autor)	*Tags*: Q: Inline, Z: Knoten *Jumps*: Q: Inline, Z: Inline	*Tags*: Eingebetteter Code „**%T**" *Jumps*: Start als „***%J Anker-Titel%%***" Ziel als „**%P**"	*Tags*: Nein *Jumps*: Nein, aber Titel des Startankers	Hierarchie
ZOG/ KMS *Seite B-7*	Grafische Benutzungsschnittstelle, grafische Link-Anker, Tooltips mit möglichen Aktionen	„Frames": Karten fester Größe, kein Scrolling	3 Kategorien: *Hierarchische, assoziative* und *programmatische Links*	Unidirektional	Binär	Nein, aber Angabe eines Link-Textes	Q: Inline Z: Knoten	Als Menüpunkte, gekennzeichnet durch Kreis, Punkt und @-Zeichen	Nein, Anzeige des Link-Titels	Hierarchie
Xanadu *Seite B-9*	Globale Verteilung, Transklusionen, Versionierung, Rechteverwaltung, Micro-Payments	Beliebige Dokumente, Zusammenführen von Dokumenten per Transklusionen	4 Kategorien: *Transpointing Windows*[1], *Transklusionen, Referenz* und *Expansion Links*	*Transpointing Windows:* Bidirektional, gerichtet	*Transpointing Windows:* Binär	Frei typisiert	*Transpointing Windows:* Q: Inline Z: Inline	*Transpointing Windows:* Endpunkt einer Linie, Textanker sind umrahmt (im Prototyp von 1999)	Nein	Multidimensionale Strukturen namens "ZigZag" (Nelson 2004)
TextNet *Seite B-11*	Semantisch streng typisierte Links, überlappende Fenster	*TOCS* zur hierarchischen Strukturierung, *Chunks*: Text-Dokumente mit Scrolling	Assoziative Hyperlinks	Bidirektional, gerichtet	Binär	Streng typisiert: 2 Basistypen, 80 Untertypen	Q: Inline Z: Knoten	Code „RF"	Titel des Ziels (bei der Menü-Ansicht)	Hierarchie
Guide *Seite B-13*	Stretchtext, adaptiver Mauszeiger	Dokumente in Fenstern mit Scrolling	*Inline Links* (s. „*Stretchtext*") Später zusätzlich: *Note, Replacement* und *Command Links*	Gerichtet	Binär	Nein	Quelle: Inline Ziel: Unterschiedlich	Typografisch hervorgehoben, adaptiver Mauszeiger	Nein	Erst in späteren Versionen: Volltext-Suche, History, Annotation etc.
HyperTies *Seite B-15*	Links als "Embedded Menues", History, Suche und Index-Funktion	„*Pages*": Bildschirmfüllend, kein Scrolling. Sequenzen von Seiten möglich.	Assoziative Hyperlinks	Unidirektional	Binär	Nein	Q: Inline Z: Knoten	Textfarbe	Ja, Beschreibung des Link-Zieles	Seiten-Index, Suche, Backtracking, History
NoteCards *Seite B-17*	Hypertext-Karten, programmatische Links, Erweiterbarkeit	Karten variabler Größe, kein Scrolling, über 40 Kartentypen	2 Kategorien: *assoziative Links* (Ziel erschien in neuem Fenster), *programmatische Links*	Unidirektional	Binär	Frei typisiert	Q: Inline Z: Knoten	Link-Icon oder Rahmen um Link-Text	Titel des Link-Zieles, Link-Typ	Hierarchie, grafische Übersichtskarten, Bookmarks („TableTops")
Intermedia *Seite B-20*	Link-Datenbank, Link-Anker als eigene Objekte, Multimedia-fähig, Client-Server-System	Multimediale Dokumente in Fenstern mit Scrolling	2 Kategorien: *assoziative Links* (Ziel erscheint in neuem Fenster), *generische Links* (Worte verweisen auf Glossareinträge)	*Assoziative Links*: Bidirektional, ungerichtet	Binär[2], Anker sind als eigene Objekte repräsentiert	Frei typisiert: Link-Titel, Schlüsselwörter, Autor, Erstellungsdatum	Q: Inline Z: Inline	Symbolisch: 	Preview nur im Client „WebView"	Volltext-Suche, Guided Tour, History, WebView, Übersichtskarten etc.
HyperCard *Seite B-23*	Autorensystem, alle Links sind mit Skripten verknüpft, Integration externer Funktionen und Datenbanken	Bildschirmfüllende Karten, kein Scrolling	Programmatische Links: Alle Links rufen HyperTalk-Skripte auf.	Unidirektional	Je nach Skript	Nein	Q: Bereiche Z: Beliebig (meist ganze Karte)	Rahmen um Bereich des Link-Ankers bei Tastendruck	Nein	Suche, grafische History mit Thumbnails
Sun's Link Service *Seite B-24*	Offener Hypertext-Dienst mit Client-Bibliotheken für die Integration in beliebige Anwendungen	Beliebige Objekte, die darstellende Applikation muss angepasst werden	Assoziative Links	Bidirektional, ungerichtet	Binär[2]	Nein	Q: Knoten Z: Knoten (feinere Granularität denkbar)	Symbolisch: 	Nein	Nein
Microcosm *Seite B-26*	Offener Hypertext-Dienst mit generischen Links	Beliebige Textdokumente, die korrespondierende Applikationen darstellen.	3 Kategorien: *Generische Links,* *lokale Links* (Variation generischer Links), *spezifische Links* (assoziative Links)	*Spezifische Links:* Bidirektional, gerichtet	*Spezifische Links:* Binär[2]	*Spezifische Links:* Frei typisiert mit Link-Kommentar	Q: Position oder Phrase in einem spezifischen oder beliebigen Dokumenten Z: Knoten oder Inline	*Spezifische Links*: Per Link-Icon oder Link-Button. Sonst durch Markieren eines Textabschnitts	Nein	Nein
Hyper-G *Seite B-28*	Global verteiltes, offenes Hypertext-System	Beliebige Objekte, die von den zugehörigen Viewern darstellbar sind	Assoziative Hyperlinks	Bidirektional, gerichtet	Binär[2], überlappende Link-Anker möglich	Frei typisiert: Schlüsselwörter, einzig vordefinierter Typ ist "Annotation".	Q: Inline Z: Knoten oder Inline	Farbiger Texthintergrund, Rahmen um Links in Grafiken und Filmen	Nein	Hierarchie, Suche, History, Backtracking, grafische Übersichtskarten
World Wide Web *Seite B-32*	Global, verteilt, skalierbar, URIs zur Adressierung, HTTP als universelles Protokoll	*Quelle*: HTML-Dokument und Dokumente in Web-fähigen Anwendungen *Ziel*: Alle per URI adressierbaren Ressourcen	Assoziative Hyperlinks	Unidirektional	Binär, weitere Möglichkeiten mit programmatischen Links	Ja, freie und strenge Typisierung;wird aber kaum genutzt (s. Abschnitt 4.4.2)	Q: Inline Z: Knoten oder Inline, sofern Anker im Zielobjekt vorhanden	Unterstrichener farbiger Text, beliebige andere Darstellung möglich	URI des Link-Zieles im Statusbereich	Backtracking, Bookmarks, History. Zusätzliche Dienste (z. B. für die Suche) und Browser-Add-Ons

[1] Transklusionen dienen zur Einbindung anderer (Teil)-Dokumente, Transpointing Windows sind assoziative Links in neuem Fenster, Referenz-Links sind assoziative Links, ersetzt die Quelle, Expansion-Links für Stretchtext

[2] Links haben zwar genau 2 Anker, ein Anker kann aber zu mehreren Links gehören. Somit werden implizit multiple Links möglich.

C Hyperlinks und Link-Anker in Hypertext-Standards

"The nice thing about standards is that there are so many of them to choose from" (Tanenbaum 2003).

Dieses Kapitel gibt einen Überblick über drei bedeutende Hypertext-Standards. Sie sind nicht nur von großer theoretischer Relevanz, denn sie fassen die Forschungsergebnisse und Erfahrungen vieler Experten zusammen und vermitteln so einen fundierten Einblick in Konzepte und Terminologien von Hypertext-Informationssystemen. In dieser Arbeit liegt dabei ein besonderes Augenmerk auf der *Modellierung* von Links und *Link-Ankern*, auf der *Typisierung* von Links und den potenziellen Auswirkungen auf die *Benutzungsschnittstelle*.

Als Erstes wird das *Dexter-Modell* vorgestellt, das Ende der 1980er Jahre die Projekterfahrungen der namhaftesten Hypertext-Experten bündelte (s. Abschnitt C.1). Es galt lange als *der* Standard für Hypertext-Systeme und wurde erst durch den Siegeszug des Webs (das das Dexter-Modell nicht berücksichtigte) in den Hintergrund gedrängt. Es schließen sich das *DeVise Hypermedia Framework* (als umfassendste Implementation des *Dexter-Modells*) und das *Amsterdam-Hypermedia-Modell* an, das das *Dexter-Modell* für das Anwendungsfeld multimedialer Daten erweitert und konkretisiert.

Abschnitt C.2 präsentiert eine Analyse des Standards *HyTime*. Diese SGML-Sprache erlaubt es, verschiedenartige Multimediaobjekte in Beziehung zueinander zu setzen und beliebige multimediale Daten mit Link-Ankern und Hyperlinks zu versehen.

Als Drittes wird die *XML Linking Language* betrachtet (Abschnitt C.3). Sie *soll(te)* im Zusammenspiel mit anderen XML-Sprachen HTML ablösen und bietet viele neue Verknüpfungsmöglichkeiten von Web-Ressourcen. Im Anschluss wird kurz das *Xspect-System* vorgestellt, die bisher weitestgehende Realisierung von *XLink* als Hypertext-System.

Das Kapitel endet mit einer Diskussion über die Möglichkeiten und Grenzen der Hypertext-Standards aus Sicht der Benutzungsschnittstelle und der Benutzbarkeit (Abschnitt C.4). Es wird kritisiert, dass die Standards kaum auf die mögliche Semantik der Links eingehen, sie diverse Aspekte der Interaktion mit Hyperlinks unbetrachtet lassen und Hinweise für eine ergonomische Umsetzung der Standards fehlen.

C.1 Das Dexter-Referenzmodell

Das *Dexter-Hypertext-Referenzmodell* war lange Zeit das im Hypertext-Forschungsbereich anerkannteste Modell für Hypertext-Systeme. Es wurde während zweier Workshops Ende der 1980er Jahre entwickelt. Der erste Workshop fand 1988 im „Dexter Inn" in New Hampshire (USA) statt, dem das Modell seinen Namen verdankt. An den Workshops nahmen Entwickler von nahezu allen damals relevanten Hypertext-Systemen teil (Halasz & Schwartz 1994).

Das Ziel des Dexter-Modells war es, eine *Standard-Terminologie* und ein *Referenzmodell* für Hypertext-Systeme zu spezifizieren, mittels derer ein gemeinsames Daten- und Prozessmodell zur Verfügung gestellt werden könnte. Es sollte damit eine Grundlage bieten, um die zur damaligen Zeit existierenden, recht unterschiedlichen Systeme und Notationen zu vergleichen und zusammenzuführen und so beispielsweise den Datenaustausch zu ermöglichen. Gleichzeitig sollte es Entwicklern eine *Architektur* für zukünftige Hypertext-Systeme an die Hand geben. Daher versuchte man, im Dexter-Modell die wichtigsten Charakteristika möglichst vieler Hypertext-Systeme zu abstrahieren und zu standardisieren.

Im Folgenden werden zuerst die *drei Schichten* des Modells erläutert. Danach wird genauer auf die Konzepte von *Komponenten*, *Links* und *Link-Ankern* im Dexter-Modell eingegangen. Die folgenden Abschnitte betrachten das *DeVise Hypermedia-System* als Anwendung des Dexter-Modells, und das *Amsterdam Hypermedia Model* als Erweiterung des Modells, da sie jeweils mehrere Aspekte der Benutzungsschnittstelle von Links konkretisieren. Den Abschluss bildet eine Zusammenfassung der Stärken und Schwächen des Modells.

C.1.1 Die Schichten des Dexter-Modells

Das Dexter-Modell unterteilt Hypertext-Systeme in drei Schichten. Diese drei Schichten sind entsprechend dem Dienst-Protokoll-Paradigma durch zwei klar definierte Schnittstellen getrennt (s. Abb. 200).

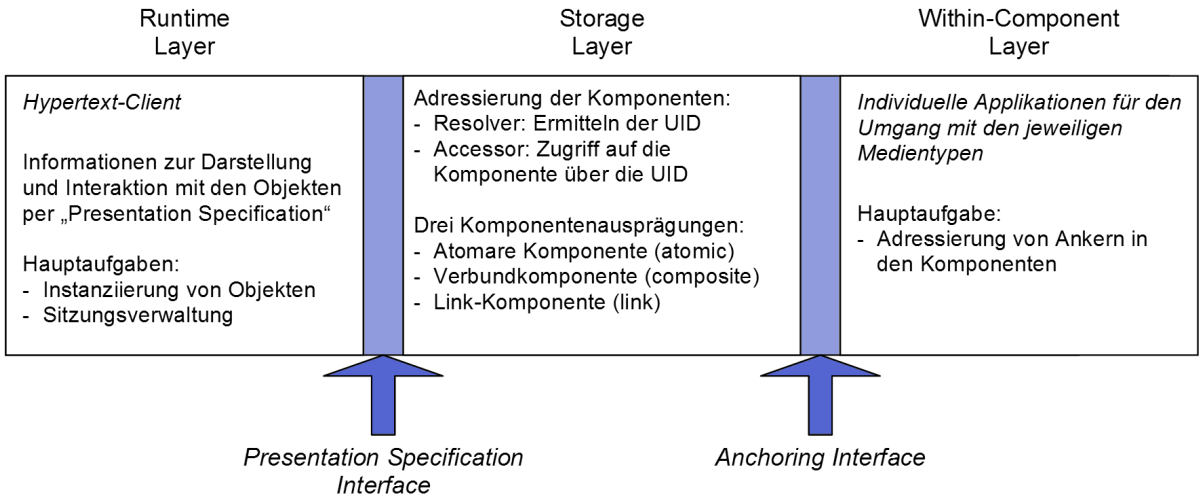


Abb. 200: Die drei Schichten und zwei Schnittstellen des Dexter-Modells.

Die oberste Schicht ist das *Runtime Layer*. Es dient der Darstellung der Dokumente und Links sowie der Interaktion mit dem Hypertext-System. Allerdings werden Präsentations- oder Verhaltensaspekte von Dokumenten und Links nicht konkret spezifiziert, da die Entwickler des Dexter-Modells die möglichen Benutzungsschnittstellen als zu vielfältig ansahen, um sie in einem generischen Modell zu fassen. Nach dem Dexter-Modell obliegt daher die Ausgestaltung der Benutzungsschnittstelle der Client-Anwendung. Das *Runtime Layer* modelliert auch das Konzept einer Benutzersitzung (Session), mittels derer Benutzern eine persönliche Umgebung ermöglicht wird, beispielsweise in Form einer History der besuchten Objekte.

Das grundlegende Konzept im *Runtime Module* ist die „*Instanziierung*" von Komponenten und der damit verbundenen Anzeige für den Benutzer. Es ist vorgesehen, dass die Objekte nach einer Bearbeitung durch den Benutzer abgespeichert werden können.

Eine konkretere Ausgestaltung als das *Runtime Layer* erfuhr im Dexter-Modell die *Schnittstelle* zwischen dem *Runtime Layer* und der darunter liegenden Ebene. Das *Presentation Specification Interface* erlaubt eine Definition von Darstellungsattributen für die Objekte des Hypertext-Systems. So kann beispielsweise festgelegt werden, welche Fenstergröße für ein bestimmtes Objekt zu bevorzugen ist und in welchem Modus ein Objekt instanziiert werden soll, d. h., ob es zur Anzeige oder zur Bearbeitung geöffnet wird.

Die mittlere Schicht nennt sich *Storage Layer* und stellt das Herzstück des Dexter-Modells dar (s. Abb. 200 Mitte). Sie modelliert die Datenbasis des Hypertext-Systems, die aus einer endlichen Menge strukturierter Objekte besteht, welche durch Relationen miteinander verknüpft werden können. Die Repräsentation *aller* Hypertext-Objekte erfolgt dabei durch sogenannte *Komponenten („Components").* Zusätzlich zur Datenbasis definiert das Storage Layer Funktionen zur Manipulation und Navigation in den Daten. Der Zugriff auf die Komponenten erfolgt mittels einer *Resolver Function* und einer *Accessor Function*. Erstere dient dem Auflösen einer *Komponentenspezifikation (Component Specification)* in einen eindeutigen *Komponenten-Identifikator (UID – Unique Identifier)*, zu dem dann die *Accessor Function* das passende Objekt zurückliefert. Neben der Verwaltung der Hypertext-Objekte sieht diese Schicht auch die Persistierung der Objekte vor, wobei sowohl der verteilte Zugriff als auch der Mehrbenutzerbetrieb vorgesehen ist.

Das *Anchoring Interface* definiert, wie sich Positionen innerhalb der Komponenten beschreiben lassen, um so die Endpunkte der Links, die sogenannten *Link-Anker*, beschreiben zu können. Es definiert damit auch den Zugriff auf die unterste Schicht, das *Within-Component Layer* (s. Abb. 200 rechts). Hier geschieht die Adressierung von Links innerhalb der Komponenten des Hypertext-Netzwerkes. Da die Art der möglichen Inhalte und Strukturen nicht eingeschränkt werden sollte, wird diese Schicht – und damit der interne Aufbau und das Format der eigentlichen Daten – nicht spezifiziert. Sie lassen sich in beliebigen anderen Standards repräsentieren, als Beispiele werden SGML (Goldfarb & Rubinsky 1990) und ODA (ISO8613-11 1995) angeführt.

C.1.2 Komponenten im Dexter-Modell

Alle Datenobjekte im Dexter-Modell werden als sogenannte Komponenten (*Components*) repräsentiert. Komponenten können sowohl atomar sein (*Atomic Component*) oder als Verbundkomponente (*Composite Component*) andere Komponenten zusammenfassen.

- *Atomare Komponenten* entsprechen dem Konzept von Knoten im assoziativen Hypertext-Paradigma (s. Abschnitt 2.2.1). Sie können beliebige Medientypen wie Texte, Grafiken oder Animationen enthalten, wobei sie als generische Datencontainer behandelt werden.

Folglich unterscheidet das Dexter-Modell beispielsweise nicht zwischen Text- oder Grafikkomponenten.

- *Verbundkomponenten* dienen der Strukturierung des Hypertextes, z. B. zum Einbetten von Objekten in andere Komponenten oder zur Gliederung in Hierarchien.

Zur Adressierung besitzt jede Komponente einen eindeutigen *Identifikator*, die *UID* (s. Abb. 201, links oben). Zudem verfügen alle Komponenten über eine Reihe von Meta-Informationen. Dies sind Daten zur Darstellung der Komponente (*Presentation Specification*) und weitere, nicht näher spezifizierte Attribute als Name-Wert-Paare. Sie können den Inhalt der Komponente beschreiben, Schlüsselwörter und Typinformationen beinhalten. Um Hyperlinks zwischen Komponenten zu ermöglichen, können in ihnen außerdem beliebig viele *Link-Anker* definiert werden (s. Abb. 201, links).

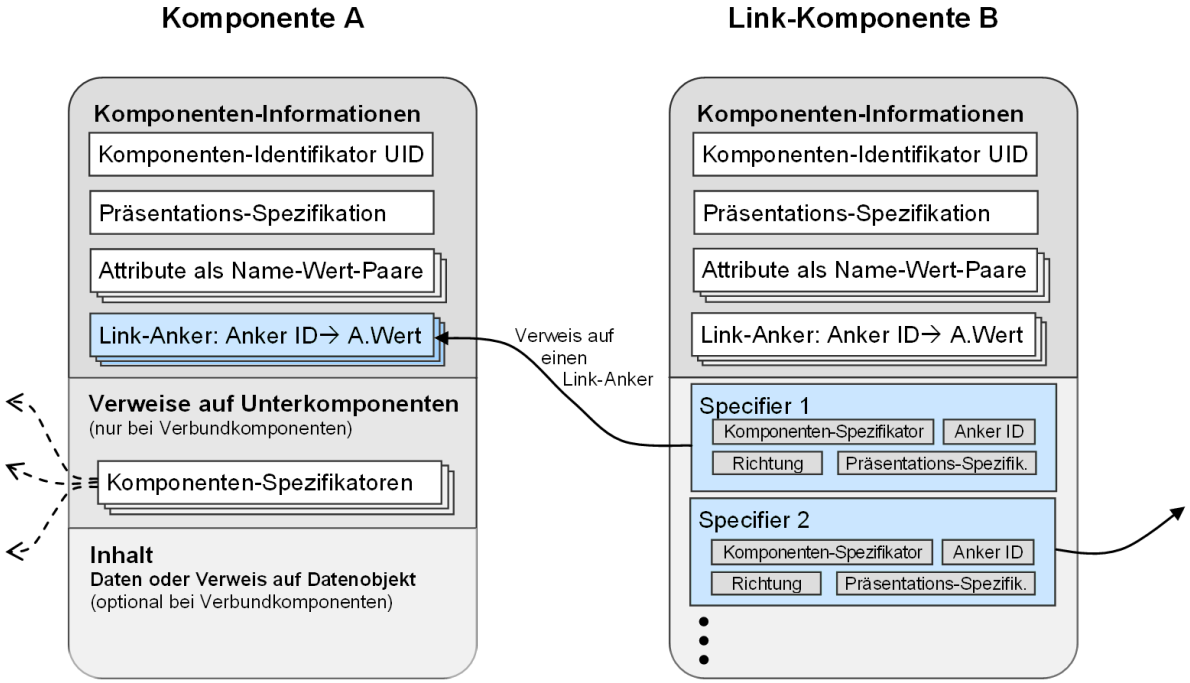


Abb. 201: Der Aufbau von Komponenten im Dexter-Modell.

C.1.3 Link-Anker im Dexter-Modell

Das Dexter-Modell definiert den *Link-Anker (Anchor)* als Punkt oder Objekt innerhalb einer Komponente, der als *Endpunkt* für Links dient (s. Abb. 201, links). Jeder Anker besteht aus zwei Elementen, einem im Dokument eindeutigen Anker-Identifikator *(Anchor ID)* und einem Ankerwert *(Anchor Value)*. Der Identifikator dient zur Adressierung des Ankers durch Links, wogegen der Ankerwert Position und Umfang des Link-Endpunktes in der Komponente spezifiziert. Der Aufbau des Ankerwertes wird allerdings nicht genauer beschrieben, da seine Interpretation durch die Applikation erfolgen soll, die für den Umgang mit dem entsprechenden Dokumenttyp verantwortlich ist.

C.1.4 Hyperlinks im Dexter-Modell

Links sind im Dexter-Modell ebenfalls Komponenten. Sie können daher ebenso wie atomare oder zusammengesetzte Komponenten sowohl Meta-Informationen zur Darstellung und zur Typisierung enthalten.[1]

Die Spezifikation der zu verknüpfenden Ressourcen erfolgt mithilfe einer Liste von zwei oder mehr Verweisen auf Link-Anker, wozu die sogenannten „*Specifiers*“ genutzt werden, die sich jeweils aus vier Attributen zusammensetzen (s. Abb. 201, rechts):

1. Die *Komponentenspezifikation (UID)* dient zur Identifikation des Zielobjekts.
2. Der Endpunkt des Links innerhalb der Komponente wird mittels der *Anker-ID definiert*.
3. Die Richtungsangabe (*Direction*) beschreibt die möglichen Traversierungsrichtungen des Links. Die vier Werte „NONE“, „FROM“, „TO“ und „BIDIRECT“ sind zulässig. Beispielsweise wird ein binärer unidirektionaler Link realisiert, indem der Startpunkt den Wert „FROM“ und der Endpunkt den Wert „TO“ erhält.
4. Ein ebenfalls *Presentation Specification* genannter Parameter beschreibt die Visualisierung des Link-Ankers im Dokument. Die Instanziierung eines Link-Ankers (also seiner Darstellung durch das *Runtime Layer*) heißt im Dexter-Modell *Link-Marker* (s. Abschnitt 2.2.1).

Einer der Hauptvorteile dieses Modells sind die flexiblen und komplexen Verknüpfungsmöglichkeiten, die vielfältige Ausprägungen von Hyperlinks zulassen. Sie reichen vom einfachen unidirektionalen Link mit einem Start- und Endpunkt über bidirektionale Links bis hin zu multiplen Links mit vielen Endpunkten. Mittels der Metadaten können sie dabei flexibel typisiert werden. Ebenfalls lassen sich vielfältige Strukturen mittels dieses Modells ausdrücken, beispielsweise kann ein Pfad (bzw. *Trail*, s. Abschnitte 3.1.5 und B.1) als eine Link-Komponente mit einer Liste von entsprechenden Specifiern beschrieben werden.

Man findet jedoch keine Angaben zur konkreten Interaktion mit Links und der Gestaltung von Link-Markern im Dexter-Modell. Dies wurde der darstellenden Anwendung im *Runtime Layer* überlassen. Im Folgenden werden daher zwei Projekte vorgestellt, die sich genauer mit dem Dexter-Modell beschäftigt haben und es weiter konkretisieren.

C.1.5 DeVise / DHM – Erfahrungen mit dem Dexter-Modell

DeVise war ein industrienahes Forschungsprojekt mehrerer Institute der Universität Århus in Dänemark, das sich mit unterschiedlichen Aspekten der Spezifikation, Entwicklung und Validation von großen verteilten Software-Systemen auseinandersetzte. Im Rahmen dieses Projekts wurde das ***De**Vise **H**ypermedia-System (DHM)* entwickelt, der wohl am besten dokumentierten und umfassendsten Realisierung des *Dexter-Modells*.

---

[1] Link-Komponenten können ebenfalls Link-Anker enthalten. Da Links auf beliebige Komponenten verweisen, ist es im Dexter-Modell auch möglich, Links auf Links verweisen zu lassen.

DHM war als Framework für offene, verteilte Hypertext-Systeme konzipiert, das die asynchrone kollaborative Arbeit mit multimedialen Dokumenten unterstützt. Bei der Implementation von *DHM* stellte sich heraus, dass sich einige der Spezifikationen des Dexter-Modells nicht einhalten ließen, andere zu unpräzise waren, und viele Anforderungen einer solchen Architektur im Standard nur ungenügend berücksichtigt wurden. Beispielsweise setzte die Kooperationsunterstützung in *DHM* eine ganze Reihe wesentlicher Erweiterungen am *Dexter-Modell* voraus, da die gleichzeitige Bearbeitung von Objekten Methoden zum Sperren von Komponenten im *Storage Layer* erforderte und Nachrichten über Änderungen an der Datenbasis benötigt wurden (Grønbæk & Trigg 1994).

C.1.6 Erweiterungen am Link-Modell

Abgesehen von solch grundsätzlichen Erweiterungen am *Storage Layer* wiesen vor allem die Komponenten und Links des Dexter-Modells Schwächen auf. So haben im Modell *Link-Komponenten* immer mindestens zwei *Specifier* (Verweise auf Anker); bei der Realisierung zeigte sich aber, dass auch Links ohne oder mit nur einem Anker unverzichtbar waren. Dies ergab sich aus Beobachtungen von Autoren bei der Arbeit mit dem System: Sie erzeugten häufig beim Schreiben eines Dokuments erst nur Links als „Platzhalter“ mit einem Anfang oder einem Ziel, bis sie Zeit fanden, weitere Verweisobjekte zu bestimmen oder zu erstellen. Zudem sollte das Ändern oder Entfernen eines Dokuments nicht dazu führen, dass automatisch alle Links gelöscht wurden, deren Anker dabei verloren gingen, da die Autoren für diese Links lieber alternative Anker definieren wollten.

Auch die *Link-Anker* des *Dexter-Modells* wiesen Defizite auf. Unter anderem blieb undefiniert, ob und wie Anker von mehreren Links gemeinsam verwendet werden könnten. Fraglich war zudem, was mit Ankern geschehen soll, wenn die auf sie verweisenden Links gelöscht werden, und welche Bedeutung Anker in Verbundkomponenten bezüglich der untergeordneten Komponenten haben.

Bei der Entwicklung des DHM-Frameworks hat sich somit die Unterscheidung von drei Ankertypen als zweckmäßig erwiesen:

- *Whole-Component Anchors* bezeichnen den gesamten Inhalt einer Komponente.
- *Marked Anchors* beziehen sich auf einen bestimmten Bereich innerhalb einer Komponente, und sie sollen für den Benutzer hervorgehoben dargestellt werden.
- *Unmarked Anchors* werden nicht hervorgehoben. Sie haben in der Regel auch keine feste Position in der Komponente, sondern können beispielsweise anhand eines Schlüsselwortes bestimmt werden.

C.1.7 Die Benutzungsschnittstelle des DHM-Systems

Das DHM-System hat diverse Aspekte des Dexter-Modells konkretisiert und zeigt eine mögliche Realisierung der Links nach dem Dexter-Modell auf. Leider wird in den Publikationen

kaum explizit auf die Benutzungsschnittstelle und Erfahrungen bezüglich der Benutzbarkeit eingegangen eingegangen.

Als Link-Marker verwendet das DHM-System Rahmen um den verknüpften Text (s. Abb. 202, rechtes Fenster). Multiple Links öffnen nach dem Anklicken eine Dialog-Box mit den möglichen Zieldokumenten, aus denen eines gewählt werden kann (Grønbæk & Trigg 1999).

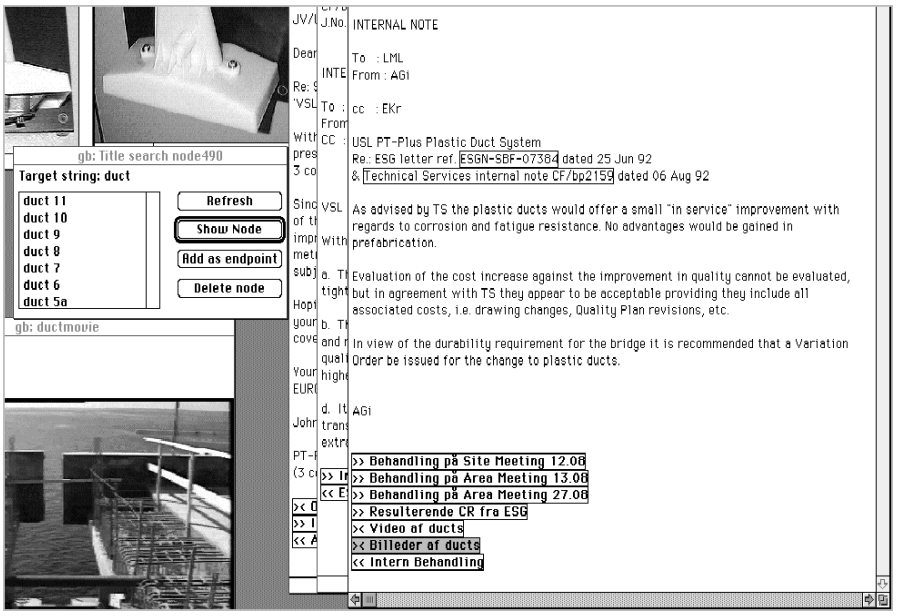


Abb. 202: Link-Anker im DHM. Auf der linken Seite ist eine Auswahl von Link-Zielen zu sehen (aus Grønbæk & Trigg 1999).

Es finden sich Vorschläge für die Darstellung von Link-Zielen, die je nach Umfang als dezenter gepunkteter Rahmen oder bei längeren Ankern als lateraler Balken links vom Text dargestellt werden könnten (vergl. Abschnitt 5.2.6 und Abb. 45, Abb. 46 und Abb. 47 auf S. 136ff).

Die *Whole-Component Anchors* werden am Ende des Dokuments mit dem Titel des Link-Ziels aufgeführt. Dabei geben Pfeile die Richtung der Verweise an (s. Abb. 202, rechts unten). Offensichtlich wird aber von den Typisierungsmöglichkeiten des Dexter-Modells kaum Gebrauch gemacht, zumindest finden sich bei den Links in den Screenshots keine Typ-Informationen. Auch werden die Präsentationsattribute des Modells nicht genutzt.

C.1.8 Das Amsterdam Hypermedia Model

Das *Amsterdam Hypermedia Model (AHM)* wurde Anfang der 1990er Jahre am CWI (Centrum voor Wiskunde en Informatica) in Amsterdam entwickelt (Hardman, Bulterman et al. 1994). Es ist eine Erweiterung des *Dexter-Modells* und die gleichzeitig eine ausführliche systematische Analyse des Modells beinhaltet. Das *AHM* berücksichtigt zusätzlich die komplexen zeitlichen und darstellungsspezifischen Beziehungen in Multimedia-Dokumenten. Die durchgeführten Ergänzungen betreffen alle Ebenen des Dexter-Modells, wobei die ausschlaggebenden Fortschritte bei der *Synchronisation von Komponenten*, der *Definition von Links* und den *Präsentationsspezifikationen* zu finden sind.

Ein Großteil des *AHM* beschäftigt sich mit der Kombination und Synchronisation von multimedialen Inhalten. Dazu favorisiert das Modell den *Composite-Structure-Ansatz*, der Medienobjekte in Verbundkomponenten gruppiert und gemeinsam abspielt, falls ihre separate Nutzung nicht sinnvoll erscheint. Ansonsten erfolgt die Synchronisation durch spezielle Verknüpfungen und *Synchronization Arcs*, die jeweils paarweise Unterkomponenten in eine zeitliche Beziehung setzen.

Im *AHM* wurde auch die Spezifikation von Link-Ankern überarbeitet, da sie für die Verwendung mit multimedialen und zusammengesetzten Objekten Defizite aufwies. Das Modell definiert nun zusätzlich das Konzept des *Link-Kontextes (Link Context)*. Er beschreibt, welche Teile einer Komponente beim Folgen eines Links beteiligt sind und wie sie dabei beeinflusst werden sollen.[2] Der Kontext des Quellankers repräsentiert die Menge an Informationen, die der Benutzer „verlässt", wenn er einen Link auswählt, wohingegen der Zielkontext den relevanten Teil des Zielobjekts umfasst. Anstatt das Quellobjekt einfach durch das Zielobjekt zu ersetzen, kann so ein genaueres Verhalten der Hypermedia-Applikation definiert werden. So könnte dem Benutzer nach dem Folgen eines Links ein Teil des Hauptfilms in einem verkleinerten Fenster mit geringerer Lautstärke weiter dargestellt werden, während er im Vordergrund Detailinformationen angezeigt bekommt.

Eine weitere Neuerung des *AHM* besteht in der Möglichkeit, Komponenten *Ausgabekanälen (Channels)* zuzuordnen. Die Ausgabekanäle sind abstrakte Ausgabegeräte, mittels derer sich benutzerbezogene Charakteristiken für die Medientypen eines Kanals definieren lassen. So kann der Benutzer beispielsweise die Lautstärke aller Audioausgaben festlegen, eine bevorzugte Sprache für Untertitel wählen oder die Darstellung von Link-Markern im Gesamtsystem ändern. Auf diese Weise ist die Gesamtpräsentation an Benutzerbedürfnisse anpassbar und muss nicht für jede Komponente neu justiert werden.

C.1.9 Die Benutzungsschnittstelle des AHM

In den Publikationen zum *AHM* wird leider ebenfalls kaum auf die mögliche Benutzungsschnittstelle für mögliche Realisierungen des Modells eingegangen. Lediglich in der Dissertation der Projektleiterin Lynda Hardman findet man einige Illustrationen (Hardman 1998). Das Beispiel verwendet für die Darstellung von Link-Ankern *Rahmen*, die den Link-Text umschließen. Als interessante Neuerung wird das Verfolgen eines Links mittels eines Überblendeffektes visualisiert (s. Abb. 203). Dies stellt vermutlich primär einen Tribut an die multimediale Ausrichtung des Projekts dar, da bei Animationen und Audio-Dateien ebenfalls kontinuierliche Übergänge empfohlen werden, um dem Benutzer Feedback für seine Navigationsaktion zu geben (ibid.).

---

[2] Diese Definition des Link-Kontextes unterscheidet sich von der im *Intermedia-System* (s. Anhang B.10), bei der ein *inhaltlicher* Kontext für den Link-Anker angegeben wird.



Abb. 203: Die Illustration von Links in einem *AHM*-konformen System sowie das Folgen des Links „Walking Routes“ (aus Hardman 1998).

C.1.10 Zusammenfassende Betrachtung zum Dexter-Modell

Das *Dexter-Modell* hat zweifellos das Verständnis vieler Konzepte und Begriffe der Hypertext-Forschung – wie Link-Anker, Link-Marker oder bidirektionale Links – vereinheitlicht. Dennoch ist es in vielen Bereichen zu abstrakt geblieben, um als Grundlage für die Realisierung von miteinander kompatiblen Hypertext-Systemen auszureichen. Das *Amsterdam Hypermedia Modell* und das *DeVise Hypermedia System* belegen dies unter anderem durch ihre Kritik an den Link-Komponenten des Dexter-Modells und die jeweils anderen Lösungsansätze (Halasz & Schwartz 1994). Da im Standard die meisten Aspekte ausgegrenzt wurden, die für die Darstellung und den Umgang mit Links und Link-Ankern entscheidend sind, sehen beide Arbeiten beispielsweise Erweiterungen und Konkretisierungen der Modellierung von Link-Ankern als notwendig an. Die vorgeschlagenen Lösungen unterscheiden sich allerdings erheblich – dies zeigt, zu welch unterschiedlicher Ausgestaltung ein solch generisches Hypertext-Modell führen kann.

Für *offene, verteilte Hypertext-Systeme* fehlen im Dexter-Modell neben den beim DHM bereits angesprochenen Mängeln der Kooperationsunterstützung (s. Abschnitt C.1.5) darüber hinaus auch Konzepte zum Umgang mit der Verteilung von Dokumenten und Links, wie sie beispielsweise für das World Wide Web unabdingbar wären.

C.2 HyTime

*HyTime (**Hy**permedia/**Time**-based Structuring Language)* ist eine auf SGML basierende Sprachspezifikation, die für offene Hypertext-Systeme konzipiert wurde, um Beziehungen zwischen verschiedenen Datentypen auf vielfältige Weise auszudrücken. Im Jahre 1992 wurde

HyTime als ISO-Standard ISO/IEC 10744 verabschiedet. Fünf Jahre später wurde *HyTime* 2 als ISO-Norm ISO/IEC 10744:1997 aktualisiert aufgenommen (Goldfarb 1997;Goldfarb, Newcomb et al. 1997). Dieser Abschnitt bezieht sich auf den neueren Standard von 1997.

HyTime verfolgt einen möglichst generischen Ansatz, um *Verknüpfungen zwischen Objekten* in einem für Computer verarbeitbaren Format zu repräsentieren. Obwohl *HyTime* selbst SGML verwendet, können damit auch Hyperlinks für andere Datenformate beschrieben werden, insbesondere auch für Multimedia-Dokumente. Aus diesem Grunde bietet es Ausdrucksformen zur Spezifikation von räumlichen und temporalen Verknüpfungs- und Synchronisationsinformationen zwischen und innerhalb von Informationskomponenten an (DeRose & Durand 1994).

Bei *HyTime* handelt es sich um einen ausgesprochen komplexen Standard: Seine Definition umfasst über 450 Seiten, wobei die Zielsetzung oft recht allgemein und abstrakt gehalten ist. Genauer betrachtet stellt sich HyTime als eine Ansammlung von Standards dar, die zwar oft enge Beziehungen untereinander haben, aber aufgrund ihres modularen Aufbaus teilweise auch unabhängig voneinander nutzbar sind. Der *HyTime*-Standard ist in 11 Klauseln (*Modules*) aufgeteilt, wobei die Hauptklauseln die Teile 6 bis 10 sind (Goldfarb 1997). Die technische Sprachdefinition erfolgt per DTDs.

Die grundlegendste dieser Hauptklauseln ist Nr. 6, das ***Base Module***, welches syntaktische Definitionen beinhaltet, die von allen weiteren Klauseln benötigt werden. Die Definition beginnt mit einer Übersicht zum Konzept und der Terminologie von HyTime. Des Weiteren findet man hier die *Hyperdocument Management Facilities*, Konventionen für die Verwaltung und die Adressierung von SGML-Hypertext-Dokumenten. Die *Coordinate Specifications* definieren eine Notation für die räumliche Adressierung von Objekten. Es gibt noch eine Reihe weiterer optionaler Bestimmungen, die beispielsweise der Zugriffssteuerung auf Dokumente dienen oder eine Definition von Referenztabellen für häufig benutzte Elemente und Mechanismen erlauben.

Das ***Location Addressing Module*** (Klausel 7) dient der Adressierung von Link-*Ankern*. Es ist insgesamt das mit Abstand umfassendste Modul von *HyTime* und bietet mehrere ausgefeilte Techniken zur Beschreibung von Bereichen in Ressourcen unterschiedlicher Medientypen. Es ermöglicht sowohl die Adressierung von Teilen von SGML-Dateien als auch von Objekten in anderen Dateiformaten. Hierzu können Koordinatensysteme, Zeitachsen und andere Adressierungssysteme eingesetzt werden.

Das grundlegende Konzept des *Location Addressing* in *HyTime* basiert auf dem Prinzip der sogenannten *Groves*, der ***Gr**aph Presentation **o**f Property **V**alu**es***. Dies ist ein Abstraktionsmechanismus, mit dem sich die Struktur beliebiger Objekte als Graph beschreiben lässt. Seine Teilelemente werden als *Property Values* (dies sind Eigenschaft-Wert-Paare) dargestellt. Für jeden Dateityp ist ein eigenes *Property Set* notwendig, das definiert, wie der Dateityp in

die Graphenrepräsentation überführt werden kann. Der *Grove Plan* dient zur Festlegung der Klassen und Eigenschaften, die in den Grove mit aufzunehmen sind.

Basierend auf der Graphen-Repräsentation von Objekten als Groves, verwendet *HyTime* drei unterschiedliche Basiskonzepte zur Adressierung von Link-Ankern in den Objekten:

- Die Adressierung per Name (*Name-space Locations*) verwendet IDs oder beliebige SGML-Attribute, um Knoten von SGML-Dateien zu referenzieren.
- Die Adressierung über Koordinaten (*Coordinate Locations*) kann Bereiche innerhalb von Objekten beschreiben, beispielsweise Areale in Grafiken und Abschnitte innerhalb von hierarchisch strukturierten Textdokumenten.
- Die Adressierung durch ein semantisches Konstrukt (*Semantic Locations*) erlaubt die Spezifizierung von Positionen durch inhaltliche Kriterien. Dieser Ansatz ist vergleichbar mit den Anfragekonzepten von Datenbanksprachen und beschreibt den gewünschten Anker durch seine Eigenschaften und Werte.

Die Konzepte zur Adressierung von Teilobjekten können in einer *Location Ladder* kombiniert werden. Jeder Ausdruck bezieht sich dabei auf das durch den vorherigen Ausdruck beschriebene Teilobjekt.

Durch die Kombination der Konzepte des *Location Addressing* lässt sich (nahezu) jedes digitale Objekt mit komplexen Link-Ankern versehen, ohne dass es speziell für die Hypertext-Nutzung angepasst oder geändert werden müsste.

Das ***Hyperlinks Module*** (Klausel 8) dient zur Definition von Relationen zwischen Objekten, wobei diese Relationen sowohl innerhalb eines Dokuments als auch zwischen verschiedenen Dokumenten verweisen können. Hyperlinks haben jeweils drei Eigenschaften: Den semantischen Typ des gesamten Links (*link type*), die Verweise auf die *Link-Anker* und die semantischen Rollen, die sie jeweils im Link spielen (*anchor roles*). HyTime 1997[3] grenzt fünf verschiedene *Link-Formen* ab:

- Die grundlegende Link-Form ist der *hylink (Hyperlink)*. Er kann beliebig viele Link-Anker haben, die als *SGML-Attribute* definiert werden. Die Bedeutung dieser Anker wird von der entsprechenden Anwendung interpretiert. Alle weiteren Link-Formen sind von ihm abgeleitet.
- Der *ilink (Independent Link)* ist semantisch identisch mit dem *hylink*, verwenden aber eine alternative Syntax zur Adressierung der Anker.
- Der *agglink (Aggregation Link)* dient zur Aggregation einer Liste von Knoten. Er vereint entweder seine Mitglieder zu einem Verbundobjekt oder gruppiert einzelne Objekte. So

[3] HyTime 1992 und 1997 unterscheiden sich insbesondere in dieser Klausel und den verfügbaren Link-Formen. Beispielsweise war in HyTime 1992 der *ilink* die grundlegende Link-Form. Seine Rolle wurde 1997 durch den *hylink* ersetzt.

lassen sich z. B. alternative Objektversionen (in unterschiedlichen Sprachen etc.) klassifizieren.

- Die einfachste Link-Form ist der *clink (Contextual Link)*, der (vergleichbar mit Links in HTML) in das SGML-Dokument eingebunden ist und über genau einen Start- und Zielanker verfügt. Der Anfang des *clink* befindet sich dabei immer an der Position seiner Definition, das Ende kann an beliebiger Stelle sein, benötigt allerdings einen entsprechenden Zielanker mit ID im Zieldokument.
- Der *varlink (Variable Link)* hat die flexibelste Adressierungssyntax für Link-Anker in HyTime. Anker werden sind nicht direkt im Link spezifiziert, sondern es wird auf zusätzliche Elemente namens *anchspec* verwiesen, die zur Anker-Adressierung dienen.

Das Hyperlinks Module unterstützt mit dem *varlink* auch Hypertext-Strukturen, deren Links mehr als zwei Anker miteinander verknüpfen und deren Anker von mehreren Links adressiert werden. Mit *HyTime* lassen sich somit komplexe Strukturen beschreiben, ohne dass aber die Verwendung der Verknüpfungen und den Umgang mit ihnen spezifiziert wird.

Das ***Scheduling Module*** und das ***Rendition Module*** (Klauseln 9 und 10) dienen zur Definition von Link-Ankern für Hypermedia-Systeme mit zeitlich veränderlichen Objekten. Die Adressierung im *Scheduling Module* basiert darauf, Objekte mit räumlichen und temporalen Achsen zu versehen und dann auf den Achsen Positionen als Link-Anker zu definieren. Das *Rendition Module* baut hierauf auf und unterstützt zusätzlich die Berücksichtigung von Änderungen an den Objekten über die Zeit. Bewegt sich beispielsweise ein verlinkter Bildausschnitt, so lässt sich hiermit ein Link-Anker ausdrücken, der dem Objekt folgt. Diese beiden Klauseln erlauben es somit auch, Audio-Daten, interaktive Präsentationen oder Animationen mit sich wandelnden Link-Ankern zu versehen. Solchen veränderlichen Link-Ankern sind Benutzer bisher nur in prototypischen Hypertext-Systemen begegnet (s. Abschnitt B.14).

C.2.1 Zusammenfassende Betrachtung zu HyTime

HyTime ist als Standard eher wissenschaftlich als praktisch relevant geworden. Die Sprache weist sehr flexible Konzepte auf, die auf hohem abstrakten Niveau definiert sind. Somit ist sie zumindest für Forscher interessant, die eine syntaktisch durchdachte Sprache zum Umgang mit multimedialen Objekten suchen (DeRose & Durand 1994).

Allerdings scheint es so, als hätte gerade die Mächtigkeit und Universalität von *HyTime* – und die damit verbundene Komplexität – seine Etablierung behindert. Die Verabschiedung der ersten ISO-Norm liegt schon über 17 Jahre zurück, es gab aber in Expertenkreisen noch lange Diskussionen über den Standard. Die hier vorgestellte zweite Auflage des Standards von 1997 ist zudem in entscheidenden Teilen inkompatibel mit dem Vorgänger.

Diese Umstände haben sicherlich auch dazu beigetragen, dass es bis heute keine vollständige Implementation von *HyTime* gibt. Es wurden lediglich Teile in unterschiedlichen Programmen umgesetzt (vergl. Goldfarb, Newcomb et al. 1997), und nur wenige Forschungsprojekte

berücksichtigen den Standard. Dabei hätte *HyTime* durchaus auch aus praktischer Sicht interessant sein können, da z. B. bis heute ein Standardformat zum Datenaustausch zwischen unterschiedlichen Multimedia-Autorensystemen fehlt.

Obwohl in *HyTime* Links und Link-Anker ausgesprochen ausführlich definiert werden, findet man keine Spezifikation zum Umgang mit Links und zur Realisierung der Link-Marker. Zudem fehlen Verweise auf ergänzende Richtlinien oder eine Referenz-Implementation.

C.3 Die XML Linking Language

Im Februar 1998 verabschiedete das World Wide Web Consortium die *Extensible Markup Language XML 1.0* als *W3C Recommendation*. Im August 2006 folgte Version 1.1, die insbesondere Fehler und Mehrdeutigkeiten behebt (Bray, Paoli et al. 2006). XML wurde als Nachfolger von SGML und HTML konzipiert, wobei sie die Flexibilität und Universalität von SGML bei einer deutlich vereinfachten Syntax bieten soll. XML hat den entscheidenden Vorteil, ein reines Datenformat zu sein, das – anders als HTML – keine immanenten Informationen zur Darstellung und zum Umgang mit den Daten enthält. Es gibt somit auch keine vorgegebenen Ausdrucksformen für Verknüpfungen zwischen den Dokumenten; eine Verwendung von XML für Hypertext-Systeme setzte daher die Definition einer entsprechenden neuen Sprache für Hyperlinks voraus. Aus diesem Grunde wurde die *XML Linking Language* (kurz: *XML Linking* oder *XLink*) als Standard für Verknüpfungen in und zwischen XML-Dokumenten entwickelt und *2001 als W3C-Standard* verabschiedet (DeRose, Maler et al. 2001). Die letzte Definition 1.1 der *XML Linking Language* stammt vom Mai 2010 und unterscheidet sich vom Vorgänger insbesondere dadurch, dass *XML Schema* statt *DTDs* für die Spezifikation verwendet wurden (DeRose, Maler et al. 2010).

Die *XML Linking Language* sollte die Einschränkungen und Schwächen der in HTML eingebetteten Links umgehen und den Ansprüchen moderner offener Hypertext-Systeme gerecht werden. Neben der Tauglichkeit für Hypertext sollte *XML Linking* zugleich eine Ausdrucksform bieten, um auch andere Arten von Beziehungen zwischen XML-Dokumenten darzustellen. Dieser allgemeine Ansatz führte dazu, dass die XML-Dialekte relativ komplex wurden.

Ausgangpunkt für die Entwicklung von *XML Linking* waren eine ganze Reihe früherer Sprachen und Konzepte, wobei *HyTime* (s. Anhang C.2) und *TEI*[4] eine besondere Rolle spielten. Des Weiteren sollte *XML Linking* auch (in gewissem Umfang) abwärtskompatibel zu den

[4] TEI sind die *Guidelines for Electronic Text Encoding and Interchange*, die versuchen, einen möglichst allgemeingültigen, zeitlosen und dauerhaften Standard zur Codierung von Textdokumenten zu schaffen (Sperberg-McQueen & Burnard 2004). TEI geht auf eine 1987 entstandene internationale Initiative von Philologen zurück. Basierend auf SGML und inzwischen auch XML kann man so Dokumente mit semantischen Strukturen, typografischen Informationen und Verweisen zwischen Dokumentteilen erstellen und speichern (Consortium 2004).

Hyperlinks in HTML sein; daher wurden Ausdrucksmöglichkeiten für unidirektionale, eingebettete Links vorgesehen.

XML Linking basiert auf drei Standards, die notwendig sind, um Links und Link-Anker zu beschreiben:

- *XLink* definiert Beziehungen zwischen Objekten. Ein XLink besteht aus einer beliebigen Anzahl von lokalen oder entfernten Ressourcen (*Resources*) und Verbindungsbögen (*Arcs*) zwischen ihnen. Eine Ressource ist eine beliebige, adressierbare Informationseinheit oder ein adressierbarer Dienst und stellt so die Link-Anker in *XLink* dar. Die Bögen dienen dazu, Beziehungen zwischen jeweils zwei Ressourcen auszudrücken (DeRose, Maler et al. 2001).
- *XPath* dient zur Adressierung von Knoten in XML-Dokumenten, die dann als Endpunkte von Links verwendbar sind. Die so selektierten Knoten können beliebige XML-konforme Teile des Gesamtdokuments sein (Clark & DeRose 1999).
- *XPointer* dient ebenfalls der Adressierung von XML-Dokumentteilen. Der Standard geht mit seinen Möglichkeiten über die von XPath hinaus und erlaubt es, praktisch beliebige Bereiche von XML-Dokumenten zu adressieren. Dies können einzelne Punkte, komplexere über XML-Abschnitte hinweggehende Regionen oder auch mehrere über das Dokument verteilte Bereiche sein (DeRose, Daniel et al. 2002).

Die nächsten Abschnitte werden diese drei Sprachen und ihre Potenziale genauer behandeln. Da *XML Linking* nach der Vorstellung von Teilen des W3C die Zukunft der Hyperlinks im Web darstellen soll(te), wird auf diese Standards etwas genauer eingegangen.

C.3.1 XLink

XLink ist das Herzstück von *XML Linking*. Es definiert dabei keine eigenen XML-Elemente, sondern fügt zu den Elementen anderer XML-Sprachen neue, durch den Namespace „`xlink:`“ gekennzeichnete Attribute hinzu. Dadurch kann XLink zusammen mit beliebigen anderen XML-Dialekten verwendet werden.

Es gibt zwei unterschiedliche Arten von *XLinks*: *einfache* und *erweiterte XLinks*. *Einfache XLinks* (*Simple Links*) wurden als Zugeständnis an die Möglichkeiten von HTML eingeführt. Sie haben eine ähnliche Syntax und sollen so einen prolemlosen Übergang zu *XLink* erlauben. Da sie ebenfalls in die Dokumente eingebettet sein müssen, bieten sie kaum mehr Möglichkeiten als HTML-Links.

Die neuen Möglichkeiten von *XLink* erschließen sich erst mit den *erweiterten XLinks* (*Extended Links*). Sie können ebenfalls Teil des XML-Dokuments sein, auf das sie sich beziehen, können aber auch außerhalb des Dokuments in einer sogenannten *Linkbase* (s. Abschnitt B.13) ge-

speichert werden.[5] Dies ermöglicht die Trennung von Inhalt und Struktur, wie es von Hypertext-Experten schon lange gefordert wurde (s. Anderson 1997; Bieber, Vitali et al. 1997a; Nelson 1999; Vitali & Bieber 1999). Bereits diese Änderung erschließt eine Vielzahl neuer Möglichkeiten für das Web, wie das Hinzufügen von Links zu Dokumenten fremder Autoren, das Annotieren beliebiger Seiten oder eine einfachere Sicherstellung der Konsistenz von Links.

XLink kann Verknüpfungen zwischen beliebig vielen *Ressourcen* (Endpunkten) beschreiben, die als *Link-Anker* des Standards anzusehen sind. Eine derartige Ressource kann in *XLink* jedes per URI adressierbare Objekt sein, also neben XML-Dokumenten auch Bilder, Animationen und Audio-Dateien. Für die genaue Adressierung innerhalb der Objekte ist der Fragment-Teil[6] des URI vorgesehen. Hier kann beispielsweise ein *XPath*- oder ein *XPointer-Ausdruck* verwendet werden.

Im Gegensatz zu den in den beiden vorherig vorgestellten Standards ist bei *XLink* nicht jede Ressource eines Links automatisch mit jeder anderen Ressource des Links verknüpft. Das Verknüpfen der Endpunkte erfolgt *explizit* mithilfe von gerichteten *Verbindungsbögen* (*Arcs*), die Teil des *XLinks* sind. Dazu werden alle Endpunkte jeweils einer Ressourcen-Klasse zugeordnet; ein Verbindungsbogen drückt dann mithilfe der Attribute „`xlink:from`“ und „`xlink:to`“ eine gerichtete Verbindung zwischen genau zwei solcher Klassen aus.

XLink bietet außerdem umfangreiche *Typisierungsmöglichkeiten*. Diese werden zum einen durch die maschinenlesbaren Attribute „`xlink:role`“ und „`xlink:arcrole`“ definiert, mit denen 1. dem gesamten XLink, 2. allen Ressourcen und 3. den Verknüpfungsbögen maschinell auszuwertende Typen in Form von URIs[7] zugewiesen werden. Genauer wird aber nicht ausgeführt, welche Möglichkeiten hier konkret vorgesehen sind, und es wurden (bisher) keine Basistypen definiert. Zusätzlich können für alle Elemente eines XLinks mittels des „`xlink:title`“-Attributs für Menschen lesbare Titel definiert werden.

Das Verhalten eines XLinks lässt sich innerhalb des Links nur grob spezifizieren. Es stehen hierfür die beiden Attribute „`xlink:show`“ und „`xlink:acutate`“ zur Verfügung, die angeben, wann ein Link aktiviert wird und wie das Link-Ziel dargestellt werden soll. So lassen sich beispielsweise Transklusionen (s. Anhang B.5) realisieren oder ein Zielobjekt in einem neuen Fenster darstellen. Diese beiden Attribute stellen aber primär ein weiteres Zugeständnis an die Möglichkeiten von HTML-Links dar. Aussehen und Verhalten eines XLinks

---

[5] Die *XLink*-Definition lässt zahlreiche Aspekte bezüglich der Verknüpfung zwischen Web-Dokumenten und externen *Xlink*-Datenbanken und der Gültigkeit der Verknüpfungsinformationen unspezifiziert (Bry & Eckert 2005).

[6] Der Fragment-Teil einer URI ist nach RFC 3986 der Code nach dem #-Zeichen. Bei HTML-Links wird dies als Sprungadresse zu einer Position innerhalb einer Webseite verwendet (Berners-Lee, Fielding et al. 2005), beispielsweise: http://www.scone.de/example.xml#**section1.**

[7] Konkret verwendet *XLink 1.1* statt dessen „IRIs“, eine Version von URIs, bei der statt des ASCII-Zeichensatzes sämtliche Unicode-Zeichen verwendet werden können (Suignard & Dürst 2005).

sollen stattdessen mittels *Style Sheets* definiert werden, ohne dass in XLink spezifiziert wurde, von welcher Art diese Style Sheets sein könnten (s. Abschnitt C.3.4).

C.3.2 XPath

XPath dient zur Adressierung von Knoten innerhalb von XML-Dokumenten (Clark & DeRose 1999). Die Sprache gestattet es, Teile eines XML-Dokuments zu spezifizieren, wobei das Ergebnis einer solchen *XPath*-Auswertung immer auch ein wohlgeformtes XML-Dokument ist. *XPath*-Ausdrücke verwenden eine kompakte, nicht XML-konforme Syntax, damit die Ausdrücke als Attributwerte von XML-Elementen verwendbar sind. Dies macht *XPath* zudem auch mit anderen XML-Dialekten nutzbar, in denen die Teile von XML-Dokumenten zu adressieren sind, beispielsweise *XSLT* (Clark 1999) und *XQuery* (Boag, Chamberlin et al. 2011).

Der Standard leitet seinen Namen von der Methode ab, mit der innerhalb der Struktur von XML-Dokumenten navigiert wird: Ein *XPath*-Ausdruck betrachtet ein XML-Dokument als Baum und beschreibt mithilfe diverser vordefinierter Funktionen einen Pfad (*Location Path*) in der Hierarchie der XML-Elemente. Dabei verwenden *XPath*-Ausdrücke entweder eine absolute oder eine relative Adressierung: Die absolute Adressierung beginnt immer bei dem Wurzel-Element des Dokuments, wogegen sich die relative Adressierung zuvor bereits spezifizierte XML-Elemente bezieht.

Die Teilausdrücke eines *XPaths* sind aus drei Teilen aufgebaut: Einer *Achsenangabe (axis)*, einem *Knotentest (node test)* und einem *Prädikatausdruck (predicates)* mit der folgenden Symtax:

```
axis::node-test[predicates]
```

- Die *Achsenangabe* definiert, in welcher Richtung der *XPath*-Ausdruck den XML-Baum durchlaufen soll. Dabei kann diese Richtung nicht nur Eltern- („*parent*“, „*ancestor*“) und Kind-Elemente („*child*“, „*descendant*“) beschreiben, sondern auch Vorgänger („*preceding*“) und Nachfolger („*following*“) eines Elementes selektieren sowie einige komplexere „Richtungen“ spezifizieren.
- Der *Knotentest* dient zur Auswahl der durch die Achsenangabe charakterisierten Elemente. Es lassen sich sowohl Elementnamen als auch Knoten eines bestimmten Typs angeben, um beispielsweise alle Textknoten auszuwählen.
- Die *Prädikate* können mittels verschiedener Funktionen, Rechenoperationen und logischer Ausdrücke die Auswahl weiter eingrenzen. Es stehen Funktionen zur Auswertung und zum Vergleich von Elementen und Attributen zur Verfügung, Methoden zum Zählen von Elementen sowie Boole'sche Ausdrücke. So können unter anderem Knoten anhand bestimmter Attributwerte oder ihrer numerischen Position bestimmt werden.

Das folgende Beispiel wählt ausgehend vom gegenwärtigen Knoten (also beispielsweise der Dokumentwurzel) das vierte Kind-Element Namens „`absatz`“ aus:

```
child::absatz[position()=4]
```

Da die Teile eines XML-Dokuments, die durch einen *XPath*-Ausdruck beschrieben werden, selbst *immer* gültige XML-Ausdrücke sind, ist *XPath* alleine unzureichend für die Spezifikation von Link-Ankern, die nur einige Worte aus einem Fließtext umfassen. Hierfür wurde *XPointer* entworfen.

C.3.3 XPointer

XPointer baut auf *XPath* auf und durchbricht dessen Beschränkung, nur Teildokumente spezifizieren zu können, die selbst wieder wohlgeformte XML-Ausdrücke sind. Es erlaubt eine wesentlich feinkörnigere Adressierung von Link-Ankern, die auch kurze Textabschnitte umfassen oder über XML-Elemente hinweggehen können. Dennoch ist die Syntax von *XPointer* ebenfalls so kompakt gehalten, dass entsprechende Ausdrücke im Fragment-Teil des URI eines Links codierbar sind.

Die Adressierung in *XPointer* beruht auf der Definition von *Punkten* (*Points*) und *Bereichen* (*Ranges*). Punkte können z. B. über Stichworte oder darauf bezogene relative Positionen adressiert werden, Bereiche werden durch zwei Punkte definiert (DeRose, Daniel et al. 2002).

XPointer wurde als eine *beschreibende* Sprache konzipiert, mit der sich Link-Anker auch aufgrund des Inhalts eines Dokuments definieren lassen. Dies gestattet *robuste* Link-Anker, die selbst bei Änderungen am Dokument noch bestimmbar sind. Zudem erlaubt es die Adressierung von Ankern, ohne dass am Dokument Änderungen erforderlich wären, beispielsweise setzt es keine Anker-IDs als Sprungmarken voraus.

Die wichtigsten Funktionen von *XPointer* sind „`range-to()`“, das mithilfe von *XPath* zwei Punkte als Start- und Endpunkt eines *XPointers* angibt und „`string-range()`“, das die Suche nach Textmustern ermöglicht.

Zwei Beispiele sollen die Potenziale von *XPointer* verdeutlichen:

```
xpointer(descending::start/range-to(following::ende[1]))
```

wählt alle Knoten eines XML-Dokuments aus, die von allen Elementen „start“ bis zum jeweils darauf folgenden „ende“-Element gehen. Auf diese Weise können sich auch längere Link-Anker ergeben, die sogar mehrfach im Dokument auftreten.

Der Ausdruck:

```
xpointer(string-range(//fruechte , "Mango",-5,15))
```

findet alle Vorkommen des Textes „Mango” innerhalb des Knotens „fruechte“. Der Anker beginnt 5 Zeichen vor dem Suchtext und umfasst 15 Zeichen. Ein solcher Ausdruck wäre z. B. für generische Links (s. Abschnitt B.13 und 5.2.7) einsetzbar, wobei auch dieser Anker mehrfach in einem Dokument auftreten kann.

C.3.4 Die Benutzungsschnittstelle von XML Linking

XML Linking geht in keinem seiner drei Standards auf geeignete Benutzungsschnittstellen für die erweiterten Link-Möglichkeiten ein (Obendorf 2001; Weinreich, Obendorf et al. 2001).

Stattdessen wird auf den Einsatz von Style-Sheets verwiesen, die aber bis heute nur fragmentarisch realisiert wurden. CSS1 und CSS2.1 bieten kaum nützliche Parameter; lediglich das seit 1998 in der Entwicklung befindliche CSS3 führt einige Erweiterungen ein, die sich auf Links und Link-Anker beziehen (Bos 2011): Das *Hyperlink Presentation Module* verfügt über Möglichkeiten zur Visualisierung von Link-Ankern in Abhängigkeit von Link-Attributen (vergl. Abschnitt 5.3) und kann beschreiben, wie und wo das Ziel eines Links erscheinen soll (beispielsweise in einem neuen Fenster oder Tab, s. auch Kapitel 4.4.4; Çelik, 2004 #469}). Die Darstellung komplexerer Verknüpfungen, wie *XLink* sie bietet – z. B. multipler Links, im Dokument verteilter oder sich überlappender Link-Anker – wird aber weiterhin nicht berücksichtigt.

Die Standards von *XML Linking* gehen ebenso wenig auf die mögliche Semantik von Links ein. Es können zwar *Links*, *Verbindungsbögen (Arcs)* und *Anker (Resources)* typisiert werden, wozu dies aber jeweils dient, welche konkrete Umsetzung dieser Möglichkeiten sinnvoll wäre und wie eine Applikation hiermit umgehen soll, wird offen gelassen. Somit definiert *XLink* beispielsweise keine Ausdrucksformen, um multiple assoziative Links (Verknüpfungen mit mehreren Zielen), von *Pfaden* (Trails, s. Abschnitt B.1), *Verbundkomponenten* (Transklusionen, s. Abschnitt B.5) oder *hierarchischen Strukturen* (vergl. Abschnitt 3.1.2) zu unterscheiden (vergl. Christensen, Hansen et al. 2003).

In Hinblick auf eine mögliche Realisierung einer Benutzungsschnittstelle von *XLink* sind Forschungsprojekte, die sich *XLink* zunutze machen, ergiebiger als der Standard selbst. Im folgenden Abschnitt wird daher das wohl bis heute fortschrittlichste derartige System kurz vorgestellt.

C.3.5 XLinks in der Praxis: Das Xspect-System

Das *Xspect-System* der Universität Århus ist ein prototyptisches Hypertext-System, das *OHIF*[8] und *XLink* verwendet, um existierende Webseiten mit zusätzlicher Hypertext-Funktionalität zu versehen. Die Strukturinformationen werden in *OHIF* gespeichert und mittels XSL-Transformationen in *XLink* umgewandelt. Diese *XLink*-Verknüpfungsdaten können dann sowohl seitens des Clients als auch des Servers mit existierenden Webseiten zusammengefügt und dem Benutzer als angereichertes Hypermedia-Dokument angeboten werden. XSL-Transformationen erzeugen aus den Ausgangsdaten Ajax-Code, der entweder als eigenes Dokument angezeigt (beispielsweise für eine Guided Tour) oder zu existierenden Webseiten hinzugefügt wird (z. B. um Annotationen zu ermöglichen).

[8] Das „*Open Hypermedia Interchange Format (OHIF)*“ ist ein XML-Dialekt, der auf ähnliche Weise wie *XLink* Verknüpfungen zwischen XML-Dokumenten ausdrücken kann. *OHIF* stellt ein noch reicheres Datenmodell zur Verfügung. Es lassen sich beispielsweise auch Verbundkomponenten, Annotationen und komplexe Rundgänge repräsentieren. Zudem benutzt *OHIF* einen Adressierungsmechanismus namens *LocSpecs*, der mehr Möglichkeiten als *XPointer* im Umgang mit anderen Dokumentformaten außer XML bietet (Grønbæk, Sloth et al. 2000).

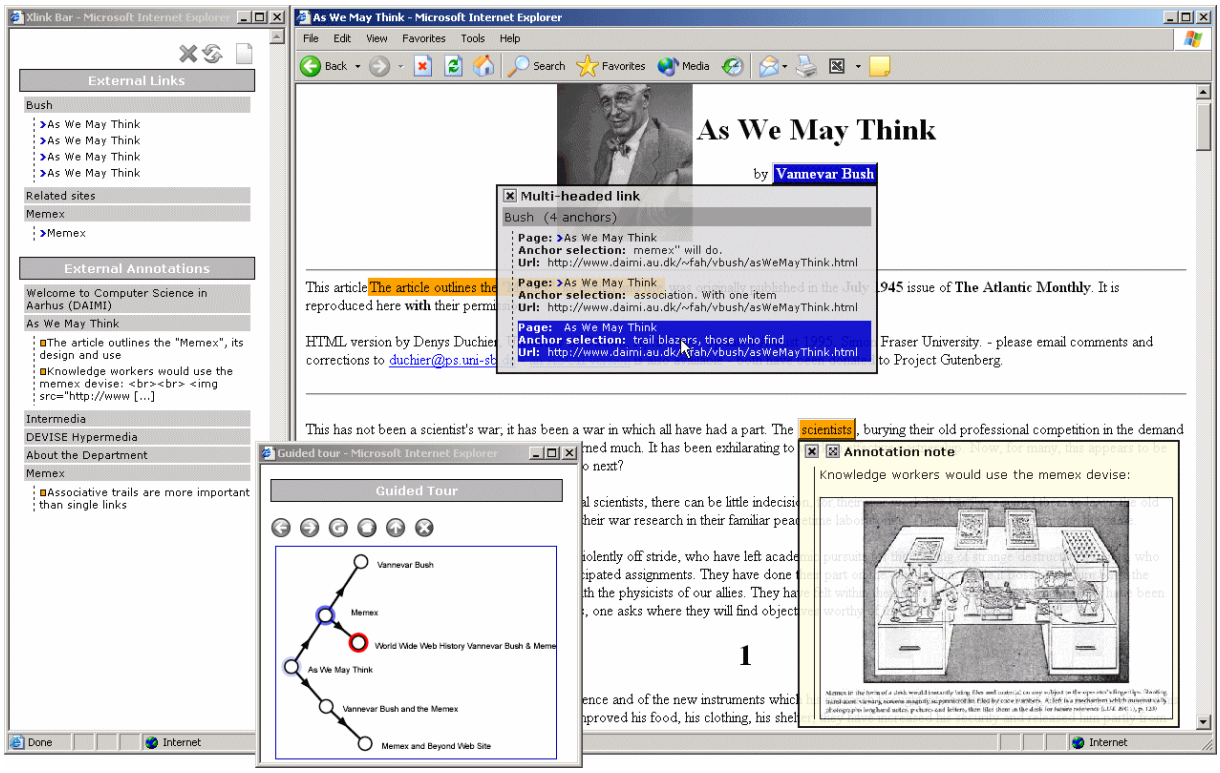


Abb. 204: Die Benutzungsschnittstelle des Xspect-Systems im Überblick. Für die Hervorhebung zusätzlicher XLinks in Webseiten werden Hintergrundfarben verwendet (nach Christensen, Hansen et al. 2003).

Abb. 204 gibt einen Eindruck von der Benutzungsschnittstelle von *Xspect* (Christensen & Hansen 2002; Christensen, Hansen et al. 2003). Rechts oben im Webbrowser ist ein multipler Link zu sehen, der mittels eines *blauen Hintergrundes* hervorgehoben wird. Der Benutzer hat ihn ausgewählt, woraufhin ein Auswahlmenü in einer Art Tooltip erschienen ist, das jeweils die Titel der Zieldokumente, ihre URIs und die Texte der Zielanker darstellt (s. auch *XlinkProxy* in: Ciancarini, Folli et al. 2002). Etwas darunter ist ein in Orange hervorgehobener Annotations-Link zu sehen. Der linke Bereich des Bildschirms zeigt einen Überblick aller zusätzlichen Links und Annotationen der aktuellen Seite. Das Fenster im Vordergrund des Bildschirmfotos bietet eine Guided Tour (s. Abschnitt 3.1.5), dessen Benutzungsschnittstelle sich an einem Metro-Streckenplan orientiert (s. auch Sandvad, Grønbæk et al. 2001).

Eine Übersicht weiterer prototypischer *XLink*-Implementationen findet man in (Macheras 2003: 37ff), von denen aber (bis heute) keine so ausgereift ist wie das *Xspect-System*.

C.3.6 Zusammenfassende Betrachtung von XLink

Anders als der Autor dieser Arbeit noch 2001 mutmaßte (Weinreich, Obendorf et al. 2001), sind die Chancen für den baldigen Erfolg von *XLink* gegenwärtig nur gering. Die Unterstützung von XML in den Browsern wurde zwar in den letzten Jahren sukzessive ausgebaut, *XLink* fand dabei allerdings keine Berücksichtigung. Es gibt bis heute keinen Parser, der *XLink* vollständig implementiert, lediglich einige wissenschaftliche Projekte machen es sich zunutze (DuCharme 2002b) und weder XHTML 1.1, XHTML 2.0 noch (X)HTML5 berücksichtigen es (vergl. Kapitel 7.5.2 und 9.2 sowie Axelsson, Epperson et al. 2010; McCarron &

Ishikawa 2010; Hickson 2011). Als Folge setzen Autoren im Web auf Techniken wie JavaScript, Ajax und Flash, um erweiterte Link- und Navigationsmöglichkeiten zu realisieren. Aktuelle Web-Autorensysteme und Content Management Systeme unterstützen solche Techniken auch umfangreich, wogegen für *XLink* ein einsetzbares Autorensystem fehlt; ein entsprechendes Werkzeug wäre aber aufgrund der Komplexität von *XLink* unverzichtbar (Wilde & Lowe 2000).

Angesichts des Entwicklungsverlaufes und der Erfahrung mit HTML lässt sich das Vorgehen bei *XLink* nicht nachvollziehen. Die unterschiedlichen Darstellungsweisen einzelner HTML-Tags in den verschiedenen Browsern und die ursprünglich fehlenden Möglichkeiten zur Definition von Style Sheets führen bis heute zu Problemen bei der Gestaltung von Webseiten. Ganze Kapitel in Büchern über Web-Design befassen sich mit Darstellungs- und Benutzbarkeitsproblemen, hervorgerufen durch die unterschiedliche Interpretation von HTML in den Browsern (z. B. Siegel 1999; Lynch & Horton 2002). Da *XLinks* jedoch weitaus komplexer sind als die Links in HTML und erheblich mehr Interaktionsmöglichkeiten bieten, erscheinen Richtlinien für die Realisierung von Hyperlinks in *XML Linking* und entsprechende Standards für Style-Sheets notwendig. Diese Kritik unterstützen beispielsweise auch die Entwickler des *Xspect*-Systems: Sie stellen aufgrund ihrer Erfahrungen im Umgang mit *XLink* fest, dass ergänzende Guidelines für *XLink* notwendig seien, da man die aktuelle Spezifikation sehr unterschiedlich interpretieren und implementieren kann. Zudem sehen sie eine direkte Integration von *XLink* in die Webbrowser als notwendig an, weil sonst viele der Möglichkeiten nur unzureichend nutzbar sind (Christensen & Hansen 2002).

Die Flexibilität von *XLink* führt gleichzeitig zu einer recht großen Komplexität des Standards. Um Strukturinformationen auszudrücken, haben daher viele Entwickler in den letzten Jahren zu anderen, einfacheren Sprachen wie *RDF* (Berners-Lee, Hendler et al. 2001) oder *XML TopicMaps* (Pepper & Moore 2000) zurückgegriffen. Ein konkretes Beispiel ist der Browser Annozilla,[9] der für die Verknüpfung von Webseiten mit Annotationen nicht *XLink*, sondern *RDF* verwendet. Nach persönlicher Aussage des Projektleiters auf der World-Wide-Web-Konferenz 2003 in Budapest war diese Lösung deutlich einfacher zu implementieren.

Offenbar wurde mit *XLink* ein Standard geschaffen, der – indem er die Benutzungsschnittstelle ausgrenzt – nicht die Bedürfnisse seiner potenziellen Anwender in den Mittelpunkt stellt und ihre praktische Fragen beantwortet (Wilde & Lowe 2000), sondern der primär versucht, technisch exzellent zu sein. Es ist abzusehen, dass er als Folge auch das Schicksal seiner Vorgänger *Dexter* und *HyTime* teilt, die in der Praxis kaum eine Rolle spielten.

[9]Annozilla ist eine Erweiterung für Mozilla Firefox. Sie erlaubt das Annotieren von Webseiten. Die Homepage des Projektes lautet: http://annozilla.mozdev.org/.

C.3.7 Weitere Hypertext-Standards

Es gibt noch eine Reihe weiterer Hypertext-Standards, die jeweils versuchen, einzelne Aspekte von Hypertext-Systemen zu normieren. Sie befassen sich in der Regel aber nicht so ausführlich mit Links und Link-Ankern wie die zuvor beschriebenen Modelle. Im Folgenden werden drei Beispiele kurz aufgeführt, um einen Eindruck zu vermitteln.

Das erste Beispiel betrifft das *Open Hypertext Protocol (OHP)*, das die Kommunikation zwischen Hypertext-Client und Link-Server standardisiert. *OHP* geht auf einen Workshop der ECHT'94-Konferenz in Edinburgh zurück. Es wurde in den folgenden Jahren zu einem Standard weiterentwickelt, der ein (recht komplexes) Protokoll und ein allgemeines Datenmodell für offene, verteilte Hypertext-Systeme definiert und eine entsprechende Middleware-Architektur vorschlägt (Davis, Reich et al. 1997). *OHP* zeigt zwar einige zukunftsweisende Ideen für die Entwicklung von Link-Servern und -Protokollen auf, hatte aber insgesamt nur wenig Einfluss auf andere Projekte. Seine Entwicklung ist trotz diverser Workshops Ende der neunziger Jahre noch im „Proposal-Status" zum Erliegen gekommen.

Der zweite erwähnenswerte Standard ist das *HAM-Modell (General Purpose **H**ypertext **A**bstract **M**achine)*, welches als universelle Architektur für Hypertext-Systeme dienen sollte (Campbell & Goodman 1988). Ähnlich wie das *Dexter-Modell* unterscheidet es drei Schichten, wobei diese aber mit einer anderen Sichtweise modelliert wurden. Die unterste Schicht ist die *Datenbank-Schicht (Database Level)*. Sie ist mit traditionellen Datenverwaltungsaufgaben vertraut, wie der Speicherung und der Zugriffssteuerung auf die Daten. Die *Hypertext Abstract Machine (HAM-)Schicht* stellt das Zentrum des Modells dar und definiert die Natur von Objekten und Links im System und die auf ihnen definierten Operationen. Es werden fünf verschiedene Objektarten definiert, unter anderem Knoten, Links und Link-Anker. Als mögliche Sprache für Hyperlinks wird in *HAM* allerdings auf *HyTime* (s. Anhang C.2) verwiesen, wodurch die Modellierung von Links und Link-Ankern ausgegrenzt wurde (Nielsen 1993a: 108). Die *Präsentationsschicht (Presentation Level)* definiert als obere Schicht die Benutzungsschnittstelle; sie ist für die Darstellung von Knoten und Links zuständig und erlaubt es, die Präsentation und Interaktion an die Bedürfnisse des Benutzers und die Möglichkeiten seiner Hardware anzupassen. So können Link-Marker und Typ-Informationen auf angemessene Art und Weise dargestellt und zusätzliche Navigationshilfen wie Übersichtskarten bereitgestellt werden. Das HAM-Modell bleibt dabei aber recht abstrakt und ist insgesamt primär für die technische Planung und die systematische Analyse offener Hypertext-Systeme geeignet.

Der dritte Standard ist das *Open Hypermedia Protocol (OHP)* der *Open Hypermedia Systems Working Group (OHSWG)*, die sich im Rahmen der jährlichen Hypertext-Konferenzen formiert hatte. Diese Arbeitsgruppe hat seit Mitte 1996 versucht, Architekturen, Schnittstellen und Protokolle für komponentenbasierte offene Hypermedia-Systeme zu entwickeln und zu standardisieren. Im Laufe von mehreren Workshops entstanden Beschreibungen für entsprechende komponentenbasierte Systeme. Dabei wurde neben assoziativem auch taxo-

nomischer und räumlicher Hypertext berücksichtigt. Es gab zahlreiche Publikationen zu verschiedenen Teilen des Protokolls und diverse Prototypen wurden vorgestellt. Obwohl man bei der Konzeption von Szenarios ausging, blieben dennoch die Ergebnisse der Gruppe auf technische Aspekte wie das Datenmodell, Kommunikationsmechanismen und eine Referenzarchitektur beschränkt; die Benutzungsschnittstelle der Systeme wurde hingegen nur marginal behandelt (Nürnberg, Wiil et al. 1997; Wiil 1997; Davis, Millard et al. 1999; Wiil 2000). Die Arbeit an dem Standard scheint Anfang dieses Jahrhunderts eingestellt worden zu sein.

In den letzten Jahren wurden keine vergleichbaren Hypertext-Standards mehr entwickelt. Die Motivation für neue Initiativen in dieser Richtung scheint zu fehlen, da sich das Web als *De-facto-Standard* für Hypertext etabliert hat. Die Protokoll- und Sprach-Standards des Webs lassen ebenfalls kaum Forschritte bezüglich der Link-Möglichkeiten erkennen. Das in der Entwicklung befindliche (X)HTML5 unterscheidet sich hierin gegenüber HTML 4.01 und XHTML 1.1 nur unwesentlich (vergl. Abschnitt 7.5.2 und 9.2). Die letzte Stylesheet-Sprache des Webs CSS3 wurde bis heute nicht fertiggestellt, und die letzte Version ihres *Hyperlinks Presentation Modules* (s. Abschnitt 4.4.4) datiert aus dem Jahre 2004 (Çelik, Bos et al. 2004; Bos 2011).

C.4 Fazit: Hypertext-Standards aus Sicht der Benutzbarkeit

Die vorgestellten Standards geben einen guten Ein- und Überblick in die Ergebnisse und Konzepte der Hypertext-Forschung und zeigen die potenziellen Möglichkeiten von Hyperlinks auf. Sie sind sowohl konzeptionell als auch technisch in vielerlei Hinsicht wegweisend und erlauben die Modellierung von derart komplexen Verknüpfungen, wie sie bisher durch kein existierendes System jemals vollständig implementiert wurden (Goldfarb, Newcomb et al. 1997; Grønbæk & Trigg 1999: 42), auch durch das Web nicht.

Gleichzeitg sind die Standards aufgrund ihrer abstrakten Definitionen in mehreren Aspekten unterschiedlich interpretierbar. Obwohl die Ausdrucksformen für Links und Link-Anker zumeist sehr ausführlich modelliert wurden, fehlen *systematische Überlegungen zur konkreten Realisierung* vieler Gesichtspunkte der Hyperlinks. Die wenigen Hinweise auf mögliche Umsetzungen scheinen insgesamt kaum durchdacht und sind eher als Anwendungsbeispiele zu werten, nicht aber als Vorschläge für einen eigenen entsprechenden Standard.

Die Ausgrenzung semantischer Aspekte von Links und Ankern und die fehlenden Überlegungen zur Benutzungsschnittstelle sind auch deshalb als kritisch anzusehen, da die erweiterten Verknüpfungsmöglichkeiten viele Fragen zum Umgang mit entsprechenden Systemen und ihrer Benutzbarkeit aufwerfen. In der Hypermedia-Forschung wurde auch ansonsten kaum systematisch die Problematik der Benutzbarkeit von Hyperlinks behandelt, es gibt daher eine ganze Reihe von Fragen, die bis in die Gegenwart unbeantwortet geblieben sind (s. auch Weinreich, Obendorf et al. 2001):

- Hyperlinks können in allen aufgeführten Modellen auf unterschiedliche Art und Weise mit *Typ-Informationen* versehen werden; allerdings finden sich keine genaueren Hinweise für die konkrete Umsetzung und für die Nutzung der Typisierung. Es wird weder darauf eingegangen, welche Funktion die Link-Typen haben sollen, noch *welche Typen sinnvoll* wären oder *wie Typ-Informationen dargestellt* werden könnten. Dabei stellt gerade die Typisierung von Links ein wichtiges Mittel dar, um die Navigation zu vereinfachen und den „Cognitive Overhead" zu reduzieren (s. Kapitel 3.2ff und 4.1ff). Dies kann aber nur gelingen, wenn die Realisierung von typisierten Links in erwartungskonformer und konsistenter Weise geschieht und dem Benutzer diese Informationen zugänglich gemacht werden. Abschnitt 4.3 geht genauer auf diese Problematik und die Möglichkeiten und Grenzen typisierter Links ein.
- Links und Link-Anker können bei allen Modellen von den Dokumenten *getrennt gespeichert* werden. Dies erlaubt es, Link-Anker zu Dokumenten hinzuzufügen, ohne dass sie geändert werden müssten oder ein Schreibzugriff auf sie notwendig wäre. Autoren werden somit in die Lage versetzt, beliebige Ressourcen mit Verknüpfungen und Annotationen zu versehen. Es werden konsistente Methoden für die Darstellung von Link-Ankern und Link-Informationen für solche zusätzlichen Links benötigt.
- Links lassen sich durch die Trennung von den Dokumenten in dedizierten *Link-Datenbanken* speichern. So werden neue Dienste realisierbar: Beispielsweise könnten *Link-Services* Benutzern zusätzliche, hochwertige Verweise auf weiterführende Informationen anbieten. Diese neuen Potenziale bergen aber das Risiko, dass einige Dokumente *sehr viele zusätzliche Links* erhalten können und der Benutzer mit einer neuen Form des *Information Overload*[10] konfrontiert wird (s. Abschnitt 3.3.2): Die Menge der Links wird dabei für den Benutzer so groß, dass er sie nicht mehr als hilfreich empfindet und er unvertretbar viel Zeit benötigt, um die für ihn nützlichen Verweise zu ermitteln. Es fehlen Konzepte zum Umgang mit dieser neuen Form des Information Overload.
- *Überlappende* Link-Anker werden von allen Standards unterstützt. Hierzu kann es kommen, wenn ein Autor *bewusst* die flexiblen Möglichkeiten der Standards nutzt; sie können aber auch *ungewollt* auftreten, wenn mehrere Autoren Link-Anker zu einem Dokument hinzufügen und diese sich zufällig überschneiden. Überlappende Links sind bisher jedoch kaum erprobt und führen zu dem Problem, dass der Start und das Ende der Link-Marker ineinander übergehen und sie somit nur schwer auseinanderzuhalten sind.
- Die vorgestellten Standards erlauben wesentlich komplexere Link-Anker als die meisten früheren Hypertext-Systeme: Ein Anker kann *umfangreiche* Teile eines Dokuments umfassen und sich *auf mehrere Abschnitte gleichzeitig* beziehen. Dabei ist bisher ungelöst, wie derartige Anker auf nicht störende und dennoch eindeutige Weise visualisierbar sind.

[10]In (Weinreich, Obendorf et al. 2001) wird vorgeschlagen, diese Variante des Information Overload folglich als *Link Overload* zu bezeichnen.

- Alle Standards unterstützen *multiple Links*, also Verknüpfungen, die auf mehrere Link-Ziele gleichzeitig verweisen. In XLink werden zusätzlich mithilfe von *Verbindungsbögen (Arcs)* komplex strukturierte Links definiert. Dem Benutzer muss daher die Möglichkeit gegeben werden, das oder die gewünschten Link-Ziele auszuwählen, wobei er ausreichende Informationen zu den unterschiedlichen Rollen der Links und zu den Zielobjekten benötigt. Zudem sollten für Benutzer Link-Anker mit multiplen Zielen eindeutig erkennbar sein, da Benutzbarkeitstests Probleme mit unerwartet auftretenden Auswahlmöglichkeiten bei Links aufgezeigt haben (Ojakaar 2001). Obwohl eine ganze Reihe früherer Hypertext-Systeme multiple Links boten, wurde ihre Benutzbarkeit kaum untersucht.
- Hyperlinks sind in allen Modellen *bidirektional* und können *gerichtet* sein. Daraus folgt, dass sie ebenfalls vom Ziel- zum Startanker traversierbar sind. Dies kann bei großen verteilten Hypertext-Systemen wie dem Web problematisch sein, da hier häufig viele Links auf einzelne, populäre Dokumente verweisen, wodurch sich eine unüberschaubare Anzahl von eingehenden Links ergibt.
- Wenn zwei Hypertext-Systeme denselben Standard unterschiedlich interpretieren, kann dies nicht nur Inkompatibilitäten, sondern auch Benutzbarkeitsprobleme zur Folge haben. Sie gelten zum einen für die Leser, die mit einer zwischen zwei Systemen inkonsistenten Bedienung konfrontiert werden. Zum anderen kann es die Arbeit der Autoren beeinträchtigen, wenn sie mit den Verweisen etwas Bestimmtes ausdrücken wollen, dies aber auf einem anderen System in falscher Weise vermittelt wird.

Die Übersicht belegt die dringende Notwendigkeit einer systematischen Entwicklung von Konzepten und Standards für die Benutzungsschnittstelle von Hyperlinks, um die Gebrauchstauglichkeit solcher Systeme sicher zu stellen.

Resümierend ist nach Ansicht des Autors der im Wesentlichen *technische Fokus* bei der Entwicklung der Hypertext-Standards nur schwer nachvollziehbar, da Hypertext ein Konzept ist, das den Zugriff auf Informationen vereinfachen soll und somit eine Verbesserung der Gebrauchstauglichkeit von Informationssystemen als Ziel hat. Leitlinien für die Realisierung angemessener Benutzungsschnittstellen hätten nicht unbedingt Teile der Standards sein müssen; entsprechende parallel entwickelte und systematisch durchdachte *Gestaltungsrichtlinien* wären hilfreich, um die Potenziale der Standards zu vermitteln sowie die Entwicklung von untereinander kompatiblen, ergonomischen Systemen zu fördern. Der geringe Erfolg, den die Standards trotz ihrer offensichtlichen Qualitäten haben, deutet darauf hin, dass diese Nachlässigkeit möglicherweise ihre Verbreitung behindert hat, zumal die Implementation der Standards unnötig erschwert wurde, da viele Aspekte für Software-Entwickler zu abstrakt und unfassbar blieben.

Diese Arbeit versucht, Teile dieser Lücken zu füllen, indem zahlreiche *Benutzbarkeitsaspekte der Benutzungsschnittstelle von Hyperlinks* am Beispiel des Web untersucht werden (s. Kapitel 3 bis 6). Den Entwicklern von neuen Systemen und Werkzeugen werden dabei Möglich-

keiten zur Gestaltung von Link-Markern und zur Anzeige zusätzlicher Link-Informationen aufgezeigt. Die Ergebnisse können auch für den Entwurf zukünftiger Standards hilfreich sein und Hinweise geben, worauf geachtet werden sollte.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, die vorliegende Dissertation "*Zur Benutzbarkeit assoziativer Verknüpfungen in verteilten Informationssystemen: Entwicklung und Evaluation von Konzepten zur Optimierung der Navigation in offenen, verteilten Hypertext-Informationssystemen gezeigt am World Wide Web*" eigenständig verfasst zu haben und keine anderen als die angegebenen Hilfsmittel verwendet zu haben.

Hamburg, 12. Dezember 2011

Dplom-Informatiker Harald Walter Reinhardt Weinreich, geb.am 13.12.1968 in Hamburg