Anwenderorientierte und minimalinvasive Leitstandmodernisierung

Dissertation zur Erlangung des akademischen Grades
Dr. rer. nat.
an der Fakultät für Mathematik, Informatik und Naturwissenschaften,
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von
Eugen Reiswich

Hamburg, 2013

Gutachter: Prof. Dr.-Ing. Heinz Züllighoven, Universität Hamburg
Gutachter: Prof. Dr.-Ing. Alexander Fay, Helmut-Schmidt-Universität Hamburg

Tag der mündlichen Prüfung: 09.01.2014

Korrigierte Fassung

Für meine Familie.

Abstract

Control systems allow operators to supervise and control industrial processes. The high level of automation in industrial plants reduces the current role of operators to one of passive supervisors. However, an entire automation of all industrial processes is still impossible and requires regular operator interventions. Current control systems support this process by providing operators with fine-grained sensor data, which is also used by machines for signal processing. User-centered approaches define the operator as an active part of the industrial process. These approaches require better control system software, which is able to support the human operator's needs. Modernizing control systems in a user-oriented way often leads to changes in software and proven workflows. This has a negative impact on the availability of plant machinery. To address this issue, this thesis provides methodological and software-technical modernization concepts which have a minimum negative impact on established workflows and systems.

This thesis starts with a summary of recurrent problems which operators experience during their work. According to industrial psychology, most of the problems operators are confronted with, can be traced back to a technology-centered design of control system software. Based on this observation, this thesis proposes a user-centered approach for control system modernization with appropriate solutions in automation engineering and computer science. The proposed methodology is summarized by the metaphor *Modernizing the Old Town*. Rather than changing entire existing control systems, the modernizing approach preserves existing systems and complements their shortcomings by new control system functionality. The proposed solution provides software-technical concepts on how to introduce user-oriented functionality in existing control systems with minimal interference and side effects on systems and workflows. In addition, this thesis provides concepts on how to loose couple new and existing control system functionality in a way where they can complement each other.

Zusammenfassung

Leitstände sind komplexe Kontrollsysteme, in denen Operateure Industrieanlagen steuern und überwachen. Der hohe Automatisierungsgrad aktueller Industrieanlagen verdrängt die Operateure zunehmend aus dem technischen Prozess und reduziert ihre Rolle zu passiven Zuschauern. Eine vollständige Automatisierung aller prozesstechnischer Aufgaben ist bislang jedoch nicht möglich und erfordert gezielte Eingriffe durch Operateure. Für diese Aufgabe bieten ihnen aktuelle Leitstandsysteme oft die gleichen feingranularen Sensordaten und Steuerungsoptionen, wie sie in der Signalverarbeitung der Maschinen verwendet werden. Anwenderorientierte Ansätze rücken die Rolle der Operateure zurück in den Mittelpunkt des Prozessgeschehens. Dies erfordert ein verbessertes und auf die Bedürfnissen der Operateure abgestimmtes Leitstandsystem. Die Modernisierung vorhandener Leitstandsysteme führt oft zu massiven Eingriffen in die Software und routinierte Arbeitsabläufe der Operateure. Diese Eingriffe beeinträchtigen häufig die Anlagenverfügbarkeit. In dieser Arbeit werden methodische und softwaretechnische Ansätze erarbeitet, die vorhandene Leitstandsysteme stärker an die Bedürfnisse der Operateure ausrichten, ohne dabei routinierte Arbeitsabläufe und vorhandene Systeme tiefgreifend zu manipulieren.

Zu Beginn der Arbeit werden aus Sicht der Operateure wiederkehrende Leitstandprobleme vorgestellt. Mit Hilfe der Arbeitspsychologie werden diese Probleme auf ein zu technologiezentriertes Design der Leitstandsysteme zurückgeführt. Auf dieser Grundlage werden Anforderungen an anwenderorientierte Leitstandlösungen formuliert und dazu passende Ansätze in der Automatisierungs- und Softwaretechnik identifiziert. Für die Einführung dieser Ansätze in vorhandene Systeme werden zunächst methodische Konzepte herausgearbeitet und in der Entwurfsmetapher *Modernisierung der Altstadt* konkretisiert. Bei diesem Ansatz bleiben bestehende Leitstandsysteme erhalten und werden an ausgewählten Stellen, dort wo sie Unzulänglichkeiten aufweisen, durch neue Leitstandfunktionalität ergänzt. Für die Realisierung dieser Entwurfsmetapher werden softwaretechnische Ansätze herausgearbeitet, die neue Leitstandfunktionalität mit wenigen Eingriffen und Seiteneffekten in vorhandene Systeme einführen. Dabei wird die neue und vorhandene Funktionalität lose gekoppelt, damit sie sich gegenseitig ergänzen können.

Danksagung

Ein Forschungsvorhaben ist eine lehrreiche und spannende, aber auch zähe und langwierige Angelegenheit. Es kann nur durch die Unterstützung besonderer Menschen gelingen, bei denen ich mich bedanken möchte.

Prof. Dr.-Ing. Heinz Züllighoven hat mich von Beginn an bei meinem Vorhaben unterstützt. Er hat mir am Arbeitsbereich Softwaretechnik und in dem von ihm und Dr. Guido Gryczan geführten Unternehmen, der C1 WPS, eine Umgebung geschaffen, in der ich an aktuellen Themen der Softwaretechnik arbeiten konnte. Unsere zahlreichen Diskussionen haben mir den Pfad zu dieser Arbeit aufgezeigt und mir geholfen, den Fokus der Arbeit einzugrenzen. Seine Sichtweise auf die Softwaretechnik hat mich über das Forschungsvorhaben hinaus geprägt.

Prof. Dr.-Ing. Alexander Fay hat mir die Welt der Automatisierungstechnik eröffnet und mir geholfen, sich in dieser Welt zurechtzufinden. Er hat mich durch den Dschungel der Industrie-Standards geleitet und mir neue Sichtweisen aufgezeigt, die einen fundamentalen Einfluss auf diese Arbeit haben.

Dr. Andreas Kornstädt hat die frühen Versionen meiner Arbeit gelesen und mir geholfen, meine Gedanken zu ordnen und den roten Faden in der Arbeit herauszuarbeiten. Unsere gemeinsamen Konferenzausflüge nach Afrika und China bleiben unvergessen.

Von all der Unterstützung durch meine Kollegen am Arbeitsbereich Softwaretechnik werden mir die Samstage und Urlaube mit Stefan Hofer in der Staatsbibliothek Hamburg wohl am meisten fehlen, wenngleich ich mich darauf freue, meine zukünftigen Samstage und Urlaube an anderen Orten zu verbringen.

Zuletzt, und doch vor allen anderen, möchte ich mich bei meiner Familie bedanken. Ihr habt mich bei all meinen Vorhaben stets unterstützt und ermuntert. Euch ist diese Arbeit gewidmet.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Entstehung der Arbeit 4

1.2 Die untersuchten Fragestellungen 7

1.3 Einordnung der Arbeit 10

1.3.1 Einordnung in der Automatisierungstechnik 10

1.3.2 Einordnung in der Arbeits- und Organisationspsychologie 12

1.3.3 Einordnung in der Softwaretechnik 13

1.4 Methodik 16

1.5 Aufbau der Arbeit 18

2 Prozesstechnische Anlagen **23**

2.1 Lebenszyklus technischer Anlagen 23

2.2 Automatisierungspyramide 25

2.3 Prozessleitsystem 27

2.3.1 Datenzugriff und Visualisierung 31

2.3.2 Alarmierung 33

2.3.3 Historischer Datenzugriff 35

2.4 Zusammenfassung 36

3 Anwendungsorientierung in der Automatisierungstechnik **39**

3.1 Datenzugriff und Visualisierung 43

3.2 Alarmierung 45

3.2.1 Alarmkonfiguration 46

3.2.2 Kooperative Alarmbearbeitung 49

3.3 Historischer Datenzugriff 52

3.4 Zusammenfassung 54

4 Anlagenmodellierung in der Prozessautomatisierung 57
4.1 Anforderungen an Anlagenmodelle auf der Leitebene 58
4.2 Metamodellierung 62
4.3 Anlagenmetamodelle 64
4.4 Standardisierte Anlagenmetamodelle 65
4.4.1 IEC 61346 65
4.4.2 IEC 61850 67
4.4.3 IEC 62424 68
4.4.4 OPC Unified Architecture 69
4.4.5 Diskussion 71
4.5 Anlagenmodellierung mit OPC UA 74
4.6 Zusammenfassung 78

5 Kurzvorstellung Lösungsansatz am Beispiel Alarmierung 81
5.1 Modellbasiertes Alarm-Werkzeug 82
5.2 Zusammenfassung 85

6 Umgang mit Legacy-Leitstandsystemen 87
6.1 Softwareevolution 90
6.1.1 Eigenschaften evolutionsfähiger Systeme 92
6.1.2 Maßnahmen zur Verbesserung der Analysierbarkeit . . 94
6.1.3 Maßnahmen zur Verbesserung der Änderbarkeit und Erweiterbarkeit 95
6.1.4 Zusammenfassung 97
6.2 OPC-UA-Modernisierungsansätze 98
6.2.1 Wrapper 99
6.2.2 Proxies 99
6.2.3 Gateways und Adapter 100
6.2.4 Zusammenfassung 100
6.3 Minimalanforderung an eine Leitstandmodernisierung 101
6.4 Minimalinvasiver Ansatz 105
6.5 Zusammenfassung 108

7 Minimalinvasive Leitstandmodernisierung aus anwenderorientierter Sicht 111

7.1 Evolutionäre Softwareentwicklung 112
7.2 Minimalinvasive Softwareerweiterung 115
7.3 Entwurfsmetapher *Modernisierung der Altstadt* 119
7.4 Zusammenfassung 122

8 Minimalinvasive Leitstandmodernisierung aus technischer Sicht 125
8.1 Modellbasierte und modulare Leitstandarchitektur 126
8.2 Einführung neuer Leitstandfunktionalität 131
8.2.1 Erweiterung des Anlagenmodell-Werkzeugs auf der Präsentationsschicht 135
8.2.2 Erweiterung des Anlagenmodells auf der funktionalen Schicht 137
8.2.3 Zusammenfassung 145
8.3 Lose Kopplung neuer und vorhandener Leitstandfunktionalität 146
8.3.1 Interaktion auf der Präsentationsschicht 147
8.3.2 Kommunikation heterogener Systeme auf der funktionalen Schicht 150
8.3.3 Kapselung der Komplexität in Messaging-Systemen . . 153
8.4 Zusammenfassung 155

9 Kurzvorstellung Realisierung 159

10 Realisierung modellbasierter Leitstandfunktionalität 163
10.1 Einführung des Kernsystems Anlagenmodell 165
10.1.1 Fachliche Architektursicht 165
10.1.2 Technische Realisierung auf der Präsentationsschicht . 166
10.1.3 Technische Realisierung auf der funktionalen Schicht . 168
10.1.4 Zusammenfassung 170
10.2 Einführung des Spezialsystems Alarmierung 171
10.2.1 Fachliche Architektursicht 171
10.2.2 Technische Realisierung auf der Präsentationsschicht . 172
10.2.3 Technische Realisierung auf der funktionalen Schicht . 177
10.3 Zusammenfassung 180

11 Evaluation 183
11.1 Evaluation der Konzepte, Modelle, Methoden und Exemplare 184

11.1.1 Konzepte 185
11.1.2 Modelle 186
11.1.3 Methoden 187
11.1.4 Exemplare 189
11.2 Evaluation am Beispiel Datenzugriff 190
11.2.1 Benutzungsmodell 190
11.2.2 Technische Realisierung 192
11.3 Evaluation am Beispiel historischer Datenzugriff 193
11.3.1 Benutzungsmodell 193
11.3.2 Technische Realisierung 195
11.4 Zusammenfassung 196

12 Zusammenfassung und Ausblick 197

Literaturverzeichnis 207

Abbildungsverzeichnis 226

Quellcodeverzeichnis 229

A Anhang: Verwendete Technologien 233
A.1 Rich Clients 233
A.1.1 Eclipse Rich Client Platform 234
A.1.2 Eclipse und OSGi 236
A.2 Control System Studio 236

B Anhang: Abbildungstabelle der OPC-UA-Metamodell-Elemente 239

Kapitel 1

Einleitung

Zu Beginn der Industrialisierung beeinflussten Arbeiter technische Prozesse noch durch manuelle Eingriffe und kontrollierten Messwerte direkt an den Sensoren einzelner Geräte. Die Weiterentwicklung technischer Anlagen führte zunehmend zu einer räumlichen Trennung der Arbeiter vom Ort des Prozessgeschehens. Ingenieure begannen, Daten mehrerer Sensoren in zentralen Kontrollräumen zusammenzufassen, wobei zunächst nur der Ort, nicht jedoch die Art der Anzeigen verändert wurde. Solche Kontrollräume bestanden aus einer Fülle von Einzelanzeigen und mechanischen Eingriffselementen und wurden daher auch als *Rollschuhwarten* bezeichnet [vgl. Urbas et al., 2012].

Mit der Einführung der Informationstechnologie Anfang der 1980er Jahre veränderten sich die ursprünglich mit realen Anlagenteilen festverdrahteten Kontrollräume hin zu vollständig elektronisch gesteuerten Softwareleitständen. Dabei übernahmen komplexe Regelungs- und Optimierungsverfahren zunehmend menschliche Routineaufgaben. Solche Veränderungen vollzogen sich über mehrere Jahre. Sie setzten meist auf bestehenden Leitstandsystemen auf, da nur in wenigen Fällen Anlagen als Komplettlösung von der Hardware bis zur Software von einem einzigen Hersteller geliefert wurden. Die inkrementellen Veränderungen führten in vielen Industrieunternehmen zu heterogenen Leitständen mit Anlagenteilen und Softwarebausteinen verschiedener Hersteller und einer Mischung aus klassischer, analoger Leittechnik und digitalen Systemen.

In modernen Anlagen laufen technische Prozesse überwiegend autonom, ohne Eingriffe durch Menschen ab. Die analogen Anzeigen und Regler verschwinden aus den Leitständen und werden durch digitale Anzeigen auf Monitoren abgelöst. Operateure überwachen technische Prozesse mit Hilfe einer Leitstandsoftware und greifen nur noch bei Abweichungen vom Regelbetrieb steuernd ein. Diese Veränderungen wurden bislang überwiegend aus einer ingenieurwissenschaftlichen Sichtweise vorangetrieben, die Mensch und Maschine als gegenseitig austauschbar betrachtet [vgl. Gryczan 1996, S. 12f; Herczeg 2008, S. 2]. Daraus resultierte der Trend, menschliche Arbeit zu minimieren und an Maschinen zu übertragen, die als zuverlässiger und weniger fehleranfällig galten [Lunze 2003, S. 14f; Nachreiner et al. 2006, S. 7]. Aus diesen Entwicklungen sind Leitstandsysteme hervorgegangen, die jedoch nur unzureichend die Arbeitssituationen der Operateure unterstützen:

- Zu Zeiten der analogen Leittechnik war es sehr zeit- und kostenintensiv, neue Anlageninformationen zu generieren, da dies häufig mit mechanischen und baulichen Veränderungen einherging [vgl. Hollifield und Habibi, 2007, S. 8]. Im digitalen Zeitalter können Anlageninformationen deutlich günstiger produziert werden. Operateure können auf umfangreiche Messwerte und detaillierte Informationen zu Anlagenteilen zugreifen. Diese datenorientierte Sichtweise verursacht eine Informationsflut, durch die sich Operateure im digitalen Zeitalter häufig weniger informiert fühlen, als im analogen [vgl. Endsley und Connors 2008, S. 2ff; Nachreiner et al. 2006, S. 8]. Aus der datenorientierten Sichtweise entstanden ferner Leitstandanwendungen, mit denen Operateure technische Prozesse sehr kleinschrittig steuern und überwachen. Häufig fehlen übergreifende Leitstandfunktionen, mit denen Operateure ganze Teilprozesse beeinflussen können.

- Der hohe Automatisierungsgrad aktueller Industrieanlagen reduziert die Operateure zu passiven Zuschauern und verdrängt sie zunehmend aus dem technischen Prozess. Durch diese andauernde Passivität verlernen Operateure ihre Aufgaben. Infolgedessen sind sie in kritischen Situationen, wenn die Automatisierung versagt, mit der Situation häufig überfordert [vgl. Wickens und Hollands, 2000].

- Heterogene Leitstandsysteme zeigen lediglich einen Ausschnitt der Anlage und somit ein fragmentiertes Abbild der Realität. Diese Fragmente müssen Operateure aus vielen Teilzuständen diverser Systeme zu einem mentalen Gesamtbild zusammenführen. Das ist gerade in kritischen Situationen schwierig, wenn viele Informationen in kurzer Zeit eintreffen. Zudem sind technische Anlagen dynamische Systeme, deren Zustand sich auch ohne Eingriffe durch Operateure verändert. Diese müssen daher ihr mentales Modell vom Anlagenzustand bei jeder Veränderung in der Anlage anpassen, was aufwändig und fehleranfällig ist. Das betrifft insbesondere die Schichtübergaben oder wenn Operateure nach langer Abwesenheit in den Anlagenbetrieb zurückkehren und das mentale Modell umfassend aktualisieren müssen [vgl. z. B. Hollnagel et al. 1995; Hoc et al. 1995; Sheridan 2012].

Die unzureichende Unterstützung der Operateure führt in der Praxis wiederkehrend zu kritischen Situationen, in denen technische Prozesse außer Kontrolle geraten. Dies resultiert in einer erhöhten Gefahr für Mensch und Umwelt [Czaja und Nair, 2012, S. 38f]. Hinzu kommen die ökonomischen Schäden durch Reparaturarbeiten, Produktionsausfälle und ggf. Image-Schäden. Das amerikanische Konsortium *Abnormal Situation Management (ASM)* schätzt den Schaden durch Anlagenausfälle allein in den USA auf über 20 Mrd. Dollar pro Jahr [Hollifield und Habibi, 2007, S. 13]. Der Großteil aller auftretenden Industrieunfälle (60-85%) wird nach Endsley und Connors durch Menschen verursacht. Die meisten Fehler sind jedoch weniger auf den menschlichen Faktor, sondern vielmehr auf ein für Operateure ungeeignetes und technologiezentriertes Design der Leitstandsoftware zurückzuführen [Endsley und Connors, 2008, S. 3f]. Anlagenbetreiber suchen daher nach Möglichkeiten, eine Industrieanlage möglichst störungsfrei zu betreiben, von der Norm abweichende Prozesszustände früh zu erkennen und diese zu beheben, noch bevor der kritische Punkt überschritten ist. Eine moderne, auf die Bedürfnisse der Operateure ausgerichtete Leitstandsoftware kann einen wesentlichen Beitrag dazu leisten. Operateure sind jedoch nicht die einzigen Akteure in Leitständen. Ingenieure, Mechatroniker, Elektrotechniker, Maschinenbauer und weitere tragen alle dazu bei, den Anlagenbetrieb aufrechtzuerhalten und kontinuierlich zu verbessern. Hierfür interagieren sie

ebenfalls an ausgewählten Stellen mit der Leitstandsoftware und gehören daher zur Zielgruppe dieser Arbeit[1].

In vielen Unternehmen existieren bereits etablierte und über Jahre erprobte Leitstandlösungen. Bei der Einführung moderner Leitstandkonzepte gab es für Unternehmen mit Altsystemen in der Vergangenheit häufig nur die Möglichkeit einer Neuentwicklung oder den Wechsel zu einer Standardsoftware großer Leitstandanbieter. Beide Alternativen sind für Anlagenbetreiber mit erheblichen Risiken und Kosten verbunden, da sie bestehende Systeme, Unternehmensdaten sowie die Ausbildung der Mitarbeiter nur unzureichend berücksichtigen. Gerade in der Einführungsphase erhöhen diese Ansätze das Unfallrisiko durch Fehlbedienung und Fehlfunktion. Zudem laufen viele Industrieanlagen in einem 24/7 Betrieb. Die Einführung neuer Lösungen muss daher in den betriebsarmen Zeiten, meist am Wochenende, erfolgen. Umfangreiche Renovierungsarbeiten sind dadurch nahezu ausgeschlossen.

Operateure und Ingenieure haben in der Vergangenheit bereits viele Technologieveränderungen erlebt, dabei jedoch nur selten spürbare Verbesserungen für ihre Arbeitssituationen erfahren. Sie stehen daher neuen Konzepten und der Einführung neuer Technologien oft skeptisch gegenüber. Für viele Unternehmen stellt sich daher die Frage, wie sie ihre Leitstände modernisieren können, ohne diese abzulösen oder massiv zu verändern. Dabei müssen sie stets Nutzen, Aufwand und Risiken gegeneinander abwägen und die bereits getätigten Investitionen in Maschinen, Software, Unternehmensdaten und Ausbildung ihrer Mitarbeiter schützen. Die Konzeption und konstruktive Umsetzung von Lösungen für die aufgezeigten Probleme ist Gegenstand dieser Dissertation.

1.1 Entstehung der Arbeit

Diese Arbeit entstand im Rahmen des GeneAL-Forschungsprojektes der C1 WPS GmbH in Kooperation mit Industrieunternehmen, dem Fachbereich Softwa-

[1] Statt diese und weitere Berufsgruppen wiederkehrend aufzuführen, werden in dieser Arbeit stellvertretend Operateure als Zielgruppe genannt. Wenn eine Unterscheidung im bestimmten Kontext hilfreich ist, werden einzelne Berufsgruppen differenziert.

retechnik der Universität Hamburg und dem Technologietransferzentrum HITeC. GeneAL (**Gene**rische **A**rchitektur für **L**eitstandsoftware) erforscht, wie der Entwicklungsaufwand für individuelle Leitstandsoftware durch vorgefertigte, generische Komponenten reduziert werden kann. Die Ergebnisse des Forschungsprojektes sollen in einer generischen Entwicklungsplattform resultieren, die wiederkehrende Leitstandprobleme durch vorgefertigte Bausteine adressiert. Unternehmen können auf dieser Plattform aufsetzen und sie durch unternehmensspezifische Bausteine konkretisieren. Dadurch soll der Entwicklungsaufwand für Leitstände um 20-30% reduziert werden. Zu den Industriepartnern im GeneAL-Projekt zählen die Hamburg Port Authority (HPA), Hamburger Hafen und Logistik AG (HHLA) sowie das Deutsche Elektronen-Synchrotron (DESY). Diese Arbeit entstand vorwiegend in Zusammenarbeit mit dem DESY, wobei auch die Erfahrungen aus den Projekten mit der HPA und der HHLA in diese Arbeit eingeflossen sind:

- **HPA:** Zu den Aufgaben der HPA gehört u.a. die Überwachung und Regelung des Schiffsverkehrs auf der Elbe. Beim Ein- und Auslauf von Containerschiffen in und aus dem Hamburger Hafen müssen Nautiker in einer Verkehrszentrale im Radarsystem ständig die aktuelle Schiffsposition beobachten und diese mit den Pegelständen anhand von Peildaten abgleichen. Sie müssen in elektronischen Wachtagebüchern auf relevante Hindernisse wie z. B. Taucherarbeiten oder Baggerarbeiten achten, die den Schiffseinlauf beeinträchtigen können. Zudem müssen sie in Absprache mit den Containerterminals sicherstellen, dass ein Schiff im Hafen einen für seine Größe geeigneten Liegeplatz zugewiesen bekommt. All diese Informationen liegen in verschiedenen Systemen und teils auch in Papierform vor. Die Nautiker müssen die verteilten Informationen zu einem mentalen Gesamtbild des Hamburger Hafens zusammenführen und mit jeder Änderung im Hafen aktualisieren.

- **HHLA:** Die HHLA betreibt mehrere Containerterminals und bietet ihren Kunden eine große Bandbreite an Dienstleistungen entlang der logistischen Kette. Bei der Beladung von Containern auf ein Schiff müssen Operateure zunächst im Lagerverwaltungssystem den gesuchten Container lokalisieren und anschließend im Fahrauftragssystem den Transport

des Containers vom Lager zum Schiff veranlassen. Dieser Transport geschieht in modernen Containerterminals vollautomatisch mit fahrerlosen Transportsystemen. Stürzt jedoch während des Transports ein Container vom fahrerlosen Transportsystem, erkennen Operateure diese Situation oft erst durch einen Blick aus dem Fenster. Um das Problem zu lösen, müssen sie anschließend in diversen Systemen Aufträge stornieren, Wartungspersonal informieren und teilweise Systemzustände durch direkte Eingriffe in der Datenbank manipulieren.

- **DESY:** Das DESY ist eines der weltweit führenden Beschleunigerzentren und betreibt in Hamburg mehrere Ring- und Linearbeschleuniger zur Erforschung der Materie. Für den Betrieb der Beschleuniger überwachen Operateure in Prozessbildern die Zustände einzelner Temperaturfühler, Ventile, Motoren, Pumpen etc., die sie bei Abweichungen vom gewünschten Zustand justieren. Bei dieser Aufgabe haben die Operateure häufig nur einen Teil der Anlage im Blick, da auf den begrenzten Monitoren im Leitstand nur ein Ausschnitt der Anlage visualisiert werden kann. Daher überwachen Operateure die Anlage zusätzlich mit Hilfe von Alarmmanagementsystemen. Diese senden meist atomare Grenzwertüberschreitungen einzelner Sensoren, oft in ganzen Alarmfluten, und erschweren Operateuren die Situationsanalyse. Manche Alarme entstehen lediglich in der Anfahrphase, die bei großen Anlagen Tage dauern kann, oder wenn Anlagenteile gewartet werden. In diesen Situationen müssen Operateure wichtige von unwichtigen Alarmen unterscheiden.

Die aufgeführten Beispielen zeigen, dass trotz Modernisierung und Automatisierung der intellektuelle Anteil an Arbeitsanforderungen der Operateure in heutigen Leitständen stetig steigt [vgl. Wickens und Hollands, 2000, S. 433]. Die heterogenen Systeme und die vielen, oft unstrukturierten Datenmengen erschweren auch auf der Managementebene, benötigte Informationen zu extrahieren. Entscheider in Unternehmen benötigen diese Informationen, um beispielsweise den operativen Betrieb zu verbessern und daraus geeignete mittel- und langfristige Planungs- und Managementaufgaben abzuleiten [vgl. Sauter, 2007, S. 16]. Für diese Aufgaben werden moderne Leitstandkonzepte

und darauf aufbauende Werkzeuge benötigt, die auf allen Leitstandebenen Menschen bei ihren vielfältigen Aufgaben besser unterstützen.

1.2 Die untersuchten Fragestellungen

Durch den zunehmenden Automatisierungsgrad in technischen Anlagen gehen die aktiven Steuerungsaufgaben der Operateure zurück, während die passiven Überwachungstätigkeiten stetig zunehmen [Kindsmüller, 2006, S. 30]. Versagt jedoch die Automatisierung, müssen Operateure die Aufgaben der Maschinen übernehmen und innerhalb kürzester Zeit den aktuellen Zustand technischer Prozesse erfassen, potentielle Auswirkungen auf die Anlage antizipieren, Handlungsoptionen abwägen und schließlich in Aktionen überführen. Aktuelle Leitstandsysteme bieten Operateuren für diese Aufgabe oft die gleichen feingranularen Sensordaten und Steuerungsoptionen, wie sie in der Signalverarbeitung der Maschinen verwendet werden [vgl. z. B. Vicente et al. 2001; Mumaw et al. 2000; Sheridan 2012].

Die Entwicklung von Leitstandsystemen ist nach wie vor durch ein rationalistisches und ingenieurwissenschaftliches Weltbild geprägt, in dem die Gebrauchstauglichkeit der Software bislang ein untergeordnetes Kriterium darstellt [vgl. Parasuraman und Riley 1997, S. 231; Nachreiner et al. 2006, S. 23; Herczeg 2008, S. 2; Endsley und Connors 2008]. Die zunehmende Automatisierung verändert die Arbeit der Operateure auf eine Weise, wie sie die Konstrukteure von Automatisierungslösungen nicht vorhergesehen haben [vgl. Emigholz 1996, S. 154; Parasuraman und Riley 1997, S. 231; Nachreiner et al. 2006, S. 8]. Daraus erwachsen neue Anforderungen an moderne Leitstandsysteme. Diese bedürfen neuer Ansätze und Methoden zur Erhöhung der Gebrauchstauglichkeit in Leitständen, die über die klassischen Interaktionskonstrukte hinausgehen [Herczeg, 2008].

Diese Arbeit untersucht die Frage, wie Leitstandsysteme nach den Prinzipien der Softwaretechnik systematisch modernisiert werden können, um die veränderten Anforderungen an ihre Gebrauchstauglichkeit besser zu unterstützen. Sie nähert sich dieser Frage aus zwei Richtungen, der Anwendungsorientierung und der technischen Sicht:

- **Anwendungsorientierung:** Unter dem Begriff *Anwendungsorientierung* verstehen Züllighoven et al. [1998, S. 261]:

  BEGRIFF 1.1: ANWENDUNGSORIENTIERUNG
  Anwendungsorientierung bei der Softwareentwicklung zielt auf folgende Merkmale der Software ab:

  - *Die Funktionalität des Softwaresystems orientiert sich an den Aufgaben im Anwendungsbereich.*
  - *Die Handhabung des Softwaresystems ist benutzergerecht.*
  - *Die im System festgelegten Abläufe und Schritte lassen sich je nach Anwendungssituation problemlos den tatsächlichen Erfordernissen anpassen.*

  In dieser Arbeit wird die Anwendungsorientierung herangezogen, um zunächst aus Sicht der Operateure aktuelle Leitstandprobleme aufzuzeigen und dazu passende Lösungsansätze zu erarbeiten. Dadurch soll die aktuell immer noch vorherrschende ingenieurwissenschaftliche Sichtweise auf Leitstände durch eine anwenderorientierte Sichtweise ergänzt werden.

- **Technische Sicht:** Die Modernisierung der Leitstandsysteme bedarf einer softwaretechnischen Basis, die Veränderungen erlaubt oder gar begünstigt. Langlebige, über mehrere Jahre erprobte und bewährte Leitstandsysteme müssen häufig zunächst in den Zustand versetzt werden, in dem sie strukturelle Änderungen zulassen und neue Funktionalität aufnehmen können. In dieser Arbeit werden softwaretechnische Prinzipien herangezogen, um bestehende Systeme weitgehend zu erhalten und durch minimale Eingriffe mit neuer Funktionalität zu ergänzen. Dadurch sollen die Leitstandsysteme im Hinblick auf die Aufgaben der Operateure verbessert und gleichzeitig die Risiken der Modernisierung reduziert werden.

Aus dieser Zielsetzung sowie der in der Einleitung formulierten Problemstellung lassen sich folgende Fragen ableiten, die in dieser Arbeit behandelt werden:

1. Welchen Beitrag kann eine anwenderorientierte Sichtweise in der Automatisierungstechnik leisten, um systematisch relevante Leitstandprobleme zu erkennen und dafür geeignete Lösungsansätze zu identifizieren?
2. Welche softwaretechnischen, ökonomischen und organisatorischen Anforderungen bilden die Voraussetzungen für eine anwenderorientierte Leitstandmodernisierung?
3. Welche methodischen Ansätze unterstützen die Einführung neuer Leitstandfunktionalität, ohne dabei bewährte Arbeitsabläufe der Operateure massiv zu verändern?
4. Welche architektonischen und softwaretechnischen Konzepte erlauben, neue Leitstandkomponenten mit möglichst wenigen Eingriffen und Seiteneffekten in vorhandene Systeme einzuführen, um dadurch Aufwände und Risiken einer Modernisierung zu reduzieren?
5. Wie können neue und vorhandene Leitstandkomponenten mit wenigen Eingriffen miteinander verknüpft und aufeinander abgestimmt werden, so dass diese nicht isoliert nebeneinander bestehen, sondern sich gegenseitig ergänzen?

Durch die Beantwortung dieser Fragen leistet diese Arbeit einen wissenschaftlichen Beitrag zur anwenderorientierten Leitstandmodernisierung, indem sie methodische und architektonische Konzepte der Softwaretechnik herausarbeitet und Maßnahmen für eine kleinschrittige, risikoreduzierte und ökonomische Leitstandmodernisierung aufzeigt. Diese Arbeit verfolgt damit das übergeordnete Ziel, die Gebrauchstauglichkeit vorhandener Leitstandsysteme zu erhöhen. Die Innovation dieser Arbeit besteht darin, die vielfach vorhandenen, aber oft technologiezentrierten Lösungsansätze der Automatisierungstechnik und Informatik systematisch nach anwenderorientierten Aspekten zu untersuchen und softwaretechnische Konstruktionsprinzipien aufzuzeigen, mit denen diese Aspekte auf vorhandene Leitstandsysteme übertragen werden können. Dabei sollen bestehende Systeme weitgehend erhalten werden, um die bereits getätigten Investitionen der Unternehmen in Software, Unternehmensdaten sowie die Ausbildung ihrer Mitarbeiter zu schützen. Gleichzeitig sollen bestehende Systeme durch minimale Eingriffe auf eine

Modernisierung vorbereitet und durch neue Technologien und Leitstandfunktionalität soweit erweitert werden, dass vorhandene und neue Konzepte sich symbiotisch ergänzen und Operateure besser bei ihren elementaren Leitstandaufgaben unterstützen.

1.3 Einordnung der Arbeit

Zur Diskussion anwenderorientierter und technischer Aspekte einer Leitstandmodernisierung werden in dieser Arbeit ausgewählte Methoden und Erkenntnisse verschiedener Disziplinen wie Automatisierungstechnik, Arbeits- und Organisationspsychologie sowie Softwaretechnik herangezogen und miteinander kombiniert. Dies ermöglicht, den Ansatz dieser Arbeit aus verschiedenen Perspektiven zu beleuchten, bislang fragmentierte Beiträge unterschiedlicher Disziplinen zu verbinden und in die Leitstandmodernisierung einfließen zu lassen. Im Folgenden werden die Forschungsinhalte der einzelnen Disziplinen vorgestellt und diese Arbeit darin eingeordnet.

1.3.1 Einordnung in der Automatisierungstechnik

Die Automatisierungstechnik beschäftigt sich mit der Planung, Realisierung und dem Betrieb automatisierter und industrieller Anlagen. Sie zeichnet sich durch eine große Methodenvielfalt, ein breites Anwendungsfeld und eine hohe Innovationsgeschwindigkeit aus [Lunze 2003, S. Vff; Früh et al. 2009, S. XXIII; Lange et al. 2010, S. XIII]. Unternehmen mit Engineering-Abteilungen und -Projekten erwarten von der Automatisierungstechnik Lösungen, die dazu beitragen, Engineering-Aufgaben effizienter zu gestalten, d.h. bei reduzierten Kosten gleichbleibend qualitativ hochwertige Engineering-Leistungen zu erbringen [Fay et al., 2009]. Zur Unterstützung dieser Anforderungen werden in der Automatisierungstechnik vermehrt Softwaresysteme eingesetzt, die inzwischen zum bestimmenden Faktor bei Produkten und Anlagen geworden sind [Lange et al., 2010, S. XIII]. Diese Softwaresysteme wurden in der Vergangenheit oft als Insellösungen entwickelt, die einzelne betriebliche Funktionen unterstützen [Früh et al., 2009, S. 18]. Mit ihrer Weiterentwick-

lung und Ausbreitung begannen die Systeme, sich stärker zu berühren, was schließlich eine Integration der vielfältigen Systeme erforderte [Früh et al., 2009, S. 18].

In der Literatur wird die *horizontale Integration* verinselter Automatisierungslösungen und die *vertikale Integration* mehrerer Automatisierungsebenen diskutiert, um den Datenaustausch zwischen Feldgeräten, dazugehöriger Steuerungs- und Überwachungssysteme sowie betriebswirtschaftlicher und strategischer Managementsysteme zu verbessern [vgl. Polke 1994, S. 21; Sauter 2007; Früh et al. 2009, S. 16f]. Die horizontale und vertikale Integration soll u.a. die Interoperabilität zwischen Subsystemen erleichtern und ihre Portabilität sowie die Durchgängigkeit des Informationsflusses erhöhen [vgl. Sauter 2007, S. 13; Früh et al. 2009, S. 19]. Diese Vernetzung ist ein aktueller Trend der sog. *Industrie 4.0*, bei der die zunehmende Durchdringung der Informationstechnologie in der Automatisierungstechnik zu einer Neuausrichtung der bis dato geltenden Referenzmodelle für industrielle Informationsarchitekturen führt [vgl. z. B. Sauer, 2013].

Die industrielle Vernetzung kann nach Polke über eine projektübergreifende Informationskonsolidierung erreicht werden, die zu einem verbindlichen Unternehmensinformationsmodell führt [Polke, 1994, S. 22]. In dieser Arbeit werden Aspekte der Informationsmodellierung aufgegriffen, mit dem Ziel, ein Modell der technischen Anlage in der Leitstandsoftware zu etablieren. In der Literatur werden dazu verschiedene Industriestandards für Anlagenmodelle sowie ihre Einführung in Leitstandsysteme diskutiert [vgl. z. B. García et al. 2004; Mackiewicz 2006; Hannelius et al. 2008; Mahnke et al. 2009; Lange et al. 2010]. Diese Diskussionen werden überwiegend aus einer technologiezentrierten und datenorientierten Sichtweise geführt. Der Fokus liegt dabei auf der Plattformunabhängigkeit, Skalierbarkeit, Sicherheit, Datenübertragung sowie auf der Modellierung komplexer Datenstrukturen mit semantisch reichhaltigen Anlageninformationen [vgl. z. B. Lange et al. 2010, S. 100; Mahnke et al. 2009, S. 19; Van Tan et al. 2009]. Die vorliegende Arbeit ergänzt die technologiezentrierte Diskussion der Anlagenmodellierung um eine anwenderorientierte Sicht. Sie zeigt, wie die Anwendungsorientierung

dabei hilft, neue Leitstandkonzepte systematisch bezüglich ihres Nutzens für Anwender zu bewerten.

Bestehende Leitstandsysteme bieten meist nur eingeschränkte Schnittstellen für neue Anlagenmodelle [vgl. z. B. Hannelius et al. 2008; Reiswich und Fay 2012]. Die Systeme müssen daher entweder massiv verändert werden, um ihre Schnittstellen zu erweitern. Alternativ können vorhandene Systeme gekapselt und mit wenigen Eingriffen erweitert werden, wodurch jedoch unabhängige und nicht interoperable Systeme entstehen [vgl. Ceaparu et al. 2004, S. 334; Hannelius et al. 2008]. In dieser Arbeit werden Modernisierungsansätze vorgestellt, die vorhandene Systeme wenig beeinträchtigen und gleichzeitig neue Funktionalität komplementär einführen und dadurch Risiken und Aufwände einer Leitstandmodernisierung reduzieren.

1.3.2 Einordnung in der Arbeits- und Organisationspsychologie

Die Arbeits- und Organisationspsychologie beschäftigt sich mit der Rolle des Menschen in komplexen Systemen [Salvendy, 2012, S. xv]. Das Anwendungsfeld in diesem Bereich umfasst Untersuchungen zur Arbeitsbelastung, zur Mensch-Maschine-Interaktion, zum Design der von Menschen verwendeten Betriebsmittel und Apparaturen, ihres Arbeitsplatzes und der Arbeitsumgebung im Hinblick auf Sicherheit, Gesundheit und Komfort [Salvendy, 2012, S. xv]. Karwowski differenziert in diesen Anwendungsfeldern weitere Spezialisierungen mit Fokus auf die physische, kognitive und organisatorische Arbeitswissenschaft [Karwowski, 2012]. Erstere beschäftigt sich mit der menschlichen Anatomie und ihren biomechanischen Charakteristika. Die kognitive Arbeitswissenschaft betrachtet mentale Prozesse wie die Wahrnehmung, Erinnerung, Informationsverarbeitung und Schlussfolgerung. Die organisatorische Arbeitswissenschaft beschäftigt sich mit der Optimierung soziotechnischer Systeme inklusive ihrer organisatorischen Strukturen, Strategien und Prozesse [Karwowski, 2012, S. 4].

Für diese Arbeit sind insbesondere Erkenntnisse über die Gestaltung der Mensch-Maschine-Schnittstelle und aus der kognitiven Arbeitswissenschaft

relevant. Der Schwerpunkt liegt dabei auf den Aufgaben der Operateure und Ingenieure in technischen Anlagen. Zu diesen Aufgaben gehören nach Sheridan die Planung aller Automatisierungsaufgaben, ihre Programmierung, so dass sie von einer Maschine automatisch ausgeführt werden können, die Überwachung automatisierter Abläufe, manuelle Steuerung, wenn die Automatisierung versagt, und aus solchen Erfahrungen zu lernen, um die Anlagenverfügbarkeit zu erhöhen [Sheridan, 2012, S. 993]. In dieser Arbeit werden vorwiegend die überwachenden und steuernden Aufgaben der Operateure untersucht, um relevante Leitstandprobleme zu identifizieren und dafür passende Lösungskonzepte zu erarbeiten.

1.3.3 Einordnung in der Softwaretechnik

Die Softwaretechnik ist ein Teilgebiet der Informatik, „[...] die sich mit der Erstellung von Programmsystemen befasst“ [Goll, 2011, S. 68]. Traditionell hat die Softwaretechnik menschliche Routinetätigkeiten automatisiert und die menschliche Arbeit auf wenige Dateneingaben reduziert [Züllighoven et al., 1998, S. 260]. „Mit der Verbreitung grafischer Anwendungen auf Arbeitsplatzrechnern und PCs verschob sich auch der Fokus von der Automatisierung menschlicher Routine zur Unterstützung alltäglicher Arbeitsaufgaben“ [Züllighoven et al., 1998, S. 260]. Dies erforderte ein Umdenken bei Softwareentwicklern, die sich nun intensiver mit Anwendern und ihren Aufgaben beschäftigen müssen. Aus diesen Herausforderungen „[...] kristallisierte sich die Anwendungsorientierung als eine zentrale Anforderung und Ausrichtung für die Softwareentwicklung heraus“ [Züllighoven et al., 1998, S. 261]. Ein Vertreter der anwendungsorientierten Softwareentwicklung ist der am Fachbereich Softwaretechnik der Universität Hamburg entwickelte Werkzeug & Material Ansatz (WAM) [Züllighoven et al., 1998], der einen fundamentalen Einfluss auf diese Arbeit hat.

WAM unterstützt den Softwareentwicklungsprozess durch eine kundenorientierte Sichtweise, bei der die Anforderungen der Anwender die softwaretechnische Realisierung bestimmen. Dazu bietet der WAM-Ansatz u.a. Unterstützung bei der Anforderungsermittlung durch für Anwender und Entwickler gleichermaßen verständliche Analysedokumente, Konstruktionstechniken,

sowie eine evolutionäre Vorgehensweise, die eine kontinuierliche Rückkopplung zwischen Anwendern und Entwicklern ermöglicht. Diese Auffassung der objektorientierten Softwareentwicklung ist in Abbildung 1.1 verdeutlicht.

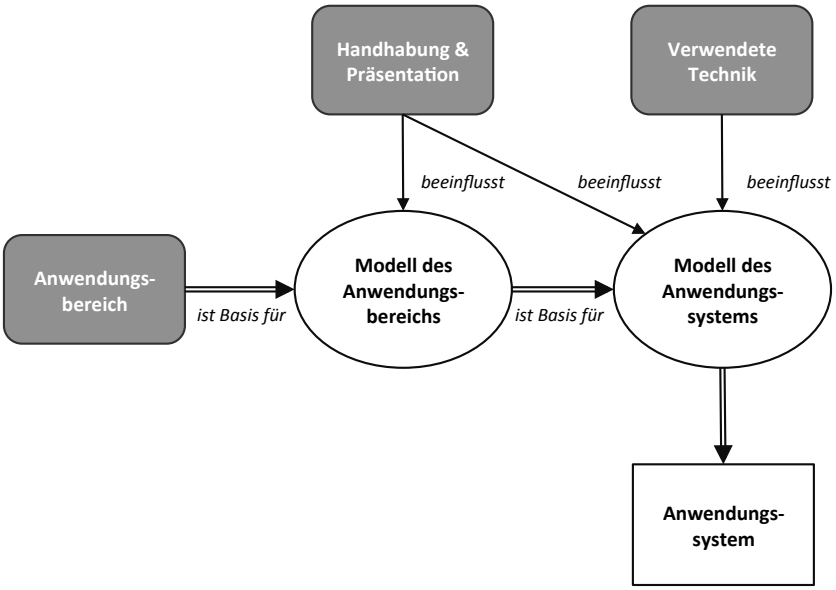


Abbildung 1.1: Vereinfachtes Modell der objektorientierten Softwareentwicklung nach WAM (Quelle: Züllighoven et al., 1998, S. 308)

Im WAM-Ansatz bildet der *Anwendungsbereich* den Ausgangspunkt der Softwareentwicklung und gleichzeitig den Kontext, in dem das zukünftige System eingesetzt wird. Entwickler müssen zunächst die Arbeitsaufgaben und Situationen im Anwendungsbereich verstehen, bevor sie ein *fachliches Modell* des Anwendungsbereichs konstruieren. Dieses Modell beschreibt Gegenstände, Aufgaben und Bezüge des Anwendungsbereichs, die durch eine Software unterstützt werden sollen. Das fachliche Modell orientiert sich dabei an den Gegenständen und an den Umgangsformen der Anwender mit diesen während der Erledigung ihrer Aufgaben. Der Umgang der Anwender mit einer Software beeinflusst somit das fachliche Modell. Aus dem Modell des Anwendungsbereichs entsteht ein *Modell des Anwendungssystems*. Dieses realisiert die fachlichen Konzepte durch softwaretechnische Konstruktionskomponenten wie z. B. Klassen, Muster oder Rahmenwerke, die in einer gewählten Programmiersprache verfasst werden. Die eingesetzte Technologie beeinflusst dabei die Realisierungsmöglichkeiten und grenzt das Modell des Anwendungssystems auf das Machbare ein. Das Ergebnis des Entwicklungsprozesses ist

schließlich das *Anwendungssystem*. Es vergegenständlicht die fachlichen und softwaretechnischen Modelle in einer ablauffähigen Anwendungssoftware [vgl. Züllighoven et al., 1998, S. 307ff].

Im WAM-Ansatz bilden die Elementtypen *Werkzeug*, *Material*, *Fachwert*, *Service* und *Automat* die Basis eines interaktiven Systems [Züllighoven et al., 1998, S. 11ff]:

- **Werkzeuge:** modellieren die interaktiven Teile eines Systems und haben eine grafische Benutzungsschnittstelle. Sie visualisieren Materialien und Fachwerte und erlauben Anwendern, diese zu bearbeiten, zu erstellen und zu entfernen. Hierfür greifen sie auch auf Services zu.
- **Materialien:** realisieren veränderliche anwendungsfachliche Gegenstände wie z. B. Konto und Kunde.
- **Fachwerte:** verkörpern anwendungsfachliche, unveränderliche Werte wie z. B. Kontonummer oder Geldbetrag.
- **Services:** bieten fachliche Dienstleistungen an, die systemweit zur Verfügung stehen. Services verwalten Materialien und bieten materialübergreifende Operationen.
- **Automaten:** erledigen autonom eine vorab festgelegte Aufgabe und produzieren ein definiertes Ergebnis. Automaten laufen meist im Hintergrund ab und werden nach ihrer Initialisierung für gewöhnlich nur im Notfall gestoppt.

In dieser Arbeit wird diese Sichtweise der objektorientierten Softwareentwicklung auf die Entwicklung von Leitstandsoftware übertragen. Dabei nehmen Operateure die Rolle des Kunden ein, dessen Arbeitsplatz im Leitstand den Anwendungsbereich für diese Arbeit bildet. Der WAM-Ansatz beeinflusst somit maßgeblich die Methodik dieser Arbeit.

1.4 Methodik

Im Fokus dieser Arbeit steht der Arbeitsplatz der Operateure, der in ein technisches System bestehend aus Software, dazugehöriger Hardware und der realen Anlage eingebettet ist. Die Operateure kooperieren bei ihrer Arbeit mit unterschiedlichen Akteuren wie Ingenieuren, Elektrotechnikern oder Mechatronikern. Die angemessene Unterstützung dieser Akteure setzt zunächst ein Verständnis ihrer Aufgaben sowie der kooperativen Arbeitsweise voraus. Zur systematischen Analyse der Aufgaben und der Kooperation in Leitständen orientiert sich diese Arbeit an dem Paradigma des *Design Research*[2] [vgl. Hevner et al. 2004; Winter 2008].

> *„The design-science paradigm [. . .] seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished“ [Hevner et al., 2004].*

Hevner et al. unterscheiden bei dieser Forschungsmethodik zwischen *Routine Design* und *Design Research* [Hevner et al., 2004]. Während *Routine Design* darauf abzielt, vorhandenes (best practice) Wissen auf unternehmensspezifische Probleme anzuwenden, hat *Design Research* zum Ziel, für wichtige und noch ungelöste Probleme einzigartige und innovative Lösungsansätze zu erforschen. Im Design Research entstehen Forschungsergebnisse durch einen zyklischen Prozess aus Entwurf und Evaluation von Artefakten. March und Smith sowie Hevner et al. unterteilen diese Artefakte in *Konzepte*, *Modelle*, *Methoden* und *Exemplare/Ausprägungen* [March und Smith 1995; Hevner et al. 2004]:

- **Konzepte** bieten ein Vokabular, um Aufgaben, Probleme und dazu passende Lösungen zu beschreiben. Sie beeinflussen signifikant die Wahr-

[2] Der Begriff *Design Research* folgt hier der Terminologie Winters [Winter, 2008], der die Anwendung des Paradigmas als Design Research bezeichnet und die Verbesserung der Paradigmas als Design Science.

nehmung des Problemfeldes und bestimmen maßgeblich die Konzeption der Modelle.

- **Modelle** nutzen Konzepte, um eine reale Situation inklusive des Design-Problems und Lösungsraums darzustellen. Modelle helfen, Probleme und Lösungen zu verstehen sowie Auswirkungen von Design-Entscheidungen auf die reale Welt nachzuvollziehen.

- **Methoden** bieten eine Anleitung, wie Probleme gelöst werden können. Methoden basieren auf Konzepten und Modellen des zugrundeliegenden Lösungsraums. Sie können formal, informell oder eine Kombination aus beidem sein.

- **Exemplare/Ausprägungen** sind ablauffähige Realisierungen der angewendeten Konzepte, Modelle und Methoden. Exemplare bzw. Ausprägungen demonstrieren die Einsatzfähigkeit der angewendeten Konzepte, Modelle und Methoden in einem realen System sowie ihre Angemessenheit für den Verwendungszweck.

Zu Beginn dieser Arbeit entstanden zunächst Konzepte, die aus einer anwenderorientierten Sicht die grundlegenden Probleme in Leitständen beschreiben. Dazu haben Interviews und Workshops mit Operateuren und Ingenieuren in den zuvor erwähnten Unternehmen DESY, HPA und HHLA, aber auch Beobachtungen und Interviews in anderen Unternehmen wie ABB (Ladenburg), BTC (Berlin), der Hamburger Hochbahn AG, Tunnelbetrieb Hamburg (Elbtunnel), Deutsche Post DHL (Hamburg), Kühne + Nagel (Hamburg), Müllverbrennungsanlage Bielefeld sowie Siemens Hamburg beigetragen. Eine intensive Literaturarbeit sowie der Austausch mit den Instituten für Automatisierungstechnik an der Helmut Schmidt Universität Hamburg, an der Otto von Guericke Universität Magdeburg sowie dem Institut für Wirtschaftsinformatik an der Johannes Kepler Universität Linz und dem Institut für Software Systems Engineering an der Universität Duisburg-Essen, halfen bei der Eingrenzung des Lösungsraumes sowie der Konzeption konkreter Lösungen.

Nach March und Smith sowie Hevner et al. müssen die konzipierten Lösungsansätze in einer technischen Infrastruktur eingesetzt und gegen die

Anforderung aus dem Anwendungskontext geprüft werden, um ihre Einsatzfähigkeit und Tragfähigkeit zu demonstrieren [March und Smith 1995; Hevner et al. 2004]. Dafür wurden in dieser Arbeit ablauffähige Softwareartefakte inkrementell und iterativ in der Leitstandsoftware beim DESY eingesetzt und mit Operateuren wiederkehrend rückgekoppelt. Dieses Feedback hat dazu beigetragen, diese Arbeit an den Anforderungen der Operateure auszurichten und damit die Angemessenheit der entwickelten Lösungen für ihren Verwendungszweck zu erhöhen. Zusätzlich wurden Auszüge dieser Arbeit in Vorträgen und Artikeln der Fachwelt vorgestellt und diskutiert [vgl. Kornstädt und Reiswich 2010a; Kornstädt und Reiswich 2010b; Reiswich und Fay 2012; Reiswich 2012a; Reiswich 2012b].

1.5 Aufbau der Arbeit

Zu Beginn dieser Arbeit werden in Kapitel 2 prozesstechnische Anlagen sowie darin eingesetzte Softwaresysteme vorgestellt. Dieses Kapitel ordnet die Arbeit zunächst entlang des Lebenszyklus technischer Anlagen ein. Der Fokus dieser Arbeit liegt auf den Lebenszyklusphasen *Errichtung*, *Inbetriebsetzung* und vor allem *Betrieb und Wartung*. In diesen Phasen nehmen Leitstandmitarbeiter unterschiedliche Aufgaben wahr, wie die Planung aller Automatisierungsaufgaben, ihre Programmierung, die Überwachung und Steuerung prozesstechnischer Abläufe sowie betriebswirtschaftliche Managementaufgaben. Zur besseren Strukturierung dieser vielfältigen Aufgaben werden sie in der Automatisierungstechnik üblicherweise entlang der Ebenen der Automatisierungspyramide eingeordnet, die ein gängiges Referenzmodell für Automatisierungsaufgaben darstellt. Der Fokus dieser Arbeit liegt auf der Prozessleitebene (Abschnitt 2.2), auf der Operateure die technische Anlage mit Hilfe einer Leitstandsoftware steuern und überwachen. In der Vergangenheit wurde Leitstandsoftware häufig aus einer ingenieurwissenschaftlichen und technologiezentrierten Sichtweise entwickelt [vgl. z. B. Hoc et al. 1995; Sheridan 1996; Parasuraman und Riley 1997; Mumaw et al. 2000; Nachreiner et al. 2006]. Daraus sind Lösungen hervorgegangen, die aktuelle Anforderungen an die Aufgaben der Operateure nur unzureichend unterstützen.

In Kapitel 3 werden aktuelle Leitstandprobleme der Operateure auf den hohen Automatisierungsgrad technischer Anlagen und eine dafür zu technologie- und datenorientierte Leitstandsoftware zurückgeführt. Mit Hilfe der Arbeits- und Organisationspsychologie werden die damit verbundenen Auswirkungen auf die Arbeit der Operateure demonstriert und der Paradigmenwechsel von einer technologiezentrierten Sichtweise hin zu einer anwenderorientierten Sichtweise vorgestellt. Mit dieser rückt der Arbeitsplatz der Operateure stärker in den Mittelpunkt des Prozessgeschehens. Die Neuausrichtung der Rolle der Operateure erfordert eine verbesserte Leitstandsoftware, die Operateuren gerade in kritischen Situationen hilft, die komplexen Anlagenzustände schneller zu erfassen und daraus Aktionen abzuleiten [vgl. Hoc et al., 1995, S. xi]. In Kapitel 3 wird die erste Forschungsfrage dieser Arbeit aufgegriffen und am Beispiel der elementaren Leitstandaufgaben demonstriert, wie eine anwenderorientierte Sichtweise dabei hilft, relevante Leitstandprobleme zu erkennen und dafür passende Lösungsansätze zu finden. Diese Lösungsansätze weisen im Kern eine Gemeinsamkeit auf: Sie benötigen ein explizites Modell der technischen Anlage mit einer hohen Strukturähnlichkeit zwischen den realen Anlagenteilen und den Modellelementen [vgl. z. B. Bratthall et al. 2002; EEMUA 2007; Hollifield und Habibi 2007; Reiswich und Fay 2012; Schlick et al. 2013].

In der Automatisierungstechnik gab es in der Vergangenheit bereits Bemühungen, ganzheitliche und möglichst mehrere Industriezweige umfassende Anlagenmodelle zu entwickeln [vlg. Sauter 2005; Sauer und Ebel 2007a; Epple 2011]. Diese Modelle haben sich bislang nicht durchgesetzt, da sie eher umfassende Visionen statt konkrete Lösungen präsentieren [vgl. Sauter 2005, S. 256ff; Sauter 2007, S. 11ff]. Folglich entstanden viele domänenspezifische und für Unternehmen nur schwer durchschaubare Standards [Mahnke et al., 2011]. Statt ein weiteres, eigenentwickeltes Anlagenmodell einzuführen, werden in Kapitel 4 die für diese Arbeit relevanten Anforderungen an Anlagenmodelle zusammengefasst, mit den aktuellen Standards verglichen, um für diese Arbeit einen geeigneten Modell-Kandidaten zu finden. Damit werden in diesem Kapitel erste softwaretechnische Aspekte der zweiten Forschungsfrage beantwortet und mit dem expliziten Anlagenmodell eine

architektonische Anforderung an eine anwenderorientierte Leitstandmodernisierung eingeführt.

Im fünften Kapitel wird die grundlegende Idee dieser Arbeit am Beispiel der elementaren Leitstandaufgabe *Alarmierung* skizziert. Dazu werden Ansätze aus der Arbeits- und Organisationspsychologie aufgegriffen, Operateure stärker in den Mittelpunkt des Prozessgeschehens zu stellen. Mit Hilfe der Anwendungsorientierung werden dazu passende Lösungsansätze systematisch identifiziert und auf Anlagenmodelle übertragen. Um diese Lösungsansätze in bestehende Leitstandsysteme einzuführen, müssen neue IT-Lösungen nach Sauter, trotz ihrer technologischen Überlegenheit, in der Lage sein, bestehende Altsysteme einzubeziehen [Sauter, 2005, S. 262]. Diese enthalten ein über mehrere Jahre angehäuftes Wissen und sind somit für Unternehmen wertvoll [Bennett, 1995].

In Kapitel 6 wird die Langlebigkeit von Leitstandsystemen hervorgehoben und die Gefahr aufgezeigt, dass Leitstandsysteme zu Legacy-Systemen zu verfallen drohen, wenn Änderungen und Erweiterungen dieser Systeme ohne entsprechende Anpassungen der Architektur erfolgen. In einem zum Legacy-System erodierten Leitstandsystem sind jegliche Modernisierungsmaßnahmen aufwändig und im Extremfall gänzlich ausgeschlossen. Statt Legacy-Leitstandsysteme abzulösen, wird diese Arbeit der Softwareevolution zugeordnet. Diese verfolgt nach Sneed et al. das Ziel, bestehende Softwaresysteme zu erhalten und ihre langfristige Wartbarkeit und Erweiterbarkeit zu verbessern [Sneed et al., 2010]. Dazu werden Eigenschaften evolutionsfähiger Systeme benannt und ausgewählte Reengineering-Maßnahmen herausgearbeitet, wie Legacy-Leitstandsysteme ihre Evolutionsfähigkeit wiedererlangen können. Auf dieser Basis werden anschließend aktuelle Leitstand-Modernisierungsansätze vorgestellt, ihre Vorgehensweise der Softwaremigration zugeordnet und die damit verbundenen Schwierigkeiten aufgezeigt. Als Alternative zu den aktuellen Modernisierungsansätzen werden im weiteren Verlauf des Kapitels softwaretechnische, ökonomische und organisatorische Einflussfaktoren auf Softwaresysteme untersucht und Minimalanforderungen an eine Leitstandmodernisierung formuliert. Als fundamentalen Einflussfaktor auf eine Modernisierung wird in diesem Kapitel die Bereitschaft der Anlagenbetreiber und

Operateure für Veränderungen identifiziert, die in einer konservativen Leitstandwelt nicht immer gegeben ist. Als Lösungsansatz für dieses Problem wird die *minimalinvasive Vorgehensweise* eingeführt und die damit verbundenen Chancen für Anlagenbetreiber aus softwaretechnischer, ökonomischer und organisatorischer Sicht aufgezeigt. Diese Aspekte beantworten im Rahmen dieser Arbeit abschließend die zweite Forschungsfrage, welche softwaretechnischen, ökonomischen und organisatorischen Anforderungen Anlagenbetreiber erfüllen müssen, um eine anwenderorientierte Leitstandmodernisierung zu ermöglichen. Das minimalinvasive Vorgehen dieser Arbeit wird in den folgenden Kapiteln detailliert und dazu anwenderorientierte und softwaretechnische Konzepte erarbeitet, die dieses Vorgehen ermöglichen.

In Kapitel 7 werden Ansätze zur minimalinvasiven Leitstandmodernisierung zunächst aus einer anwenderorientierten Sicht untersucht und die Metapher *Modernisierung der Altstadt* eingeführt. Diese Metapher demonstriert methodische und softwaretechnische Konzepte, wie vorhandene Leitstandsysteme mit neuer und verbesserter Funktionalität ergänzt werden können. Dazu tragen evolutionäre Vorgehensmodelle bei, die neue Funktionalität schrittweise einführen. Die *Softwareerweiterung* ergänzt dieses Vorgehen und führt neue Leitstandfunktionalität zusätzlich zur vorhandenen ein. Operateure können dadurch Neuerungen schrittweise, in z. B. weniger kritischen Situationen erproben und erlernen. Diese Aspekte beantworten die dritte Forschungsfrage und präsentieren eine Methodik, mit der neue Funktionalität in vorhandene Leitstandsysteme einführt werden kann, ohne dabei bewährte Arbeitsabläufe der Operateure massiv zu verändern.

Im achten Kapitel werden die softwaretechnischen Randbedingungen einer minimalinvasiven Leitstandmodernisierung untersucht. Dazu wird zunächst eine (anlagen-)modellbasierte und modulare Leitstandarchitektur konzipiert. Diese erweitert die klassische Drei-Schichten-Architektur und ermöglicht eine strukturierte und organisierte Leitstandmodernisierung. Auf Basis der Plug-in-Technologie werden minimalinvasive Konzepte zur Einführung neuer Leitstandfunktionalität vorgestellt und mit Architekturmustern ergänzt, die eine langfristige Evolution der neuen und vorhandenen Leitstandfunktionalität unterstützen. Diese Konzepte beantworten die Forschungsfrage, welche

architektonischen und softwaretechnischen Konzepte die Einführung neuer Leitstandfunktionalität mit möglichst wenigen Eingriffen und Seiteneffekten in vorhandene Systeme erlauben. Damit die neu eingeführte Leitstandfunktionalität die vorhandene Funktionalität komplementär ergänzen kann, werden im achten Kapitel Ansätze zur losen Kopplung von Werkzeugen auf der Präsentationsschicht und verteilter heterogener Systeme auf der funktionalen Schicht vorgestellt. Mit dem Nachrichtenaustausch wird ein für beide Schichten geeigneter Ansatz zur Interaktion identifiziert. Die mit dem Nachrichtenaustausch oft einhergehende Komplexität wird durch eine Datenzugriffsschicht eingedämmt. Damit wird in diesem Kapitel die letzte Forschungsfrage beantwortet und Konzepte präsentiert, die neue und vorhandene Leitstandkomponenten mit wenigen Eingriffen miteinander verknüpfen und aufeinander abstimmen, so dass diese sich gegenseitig ergänzen können.

Kapitel 9 fasst die Ergebnisse vorangegangener Kapitel zur anwenderorientierten und minimalinvasiven Leitstandmodernisierung zusammen und stellt den Lösungsansatz dieser Arbeit vor. In Kapitel 10 wird der Ansatz dieser Arbeit am Beispiel der Fallstudie DESY konstruktiv umgesetzt und damit seine Einsatzfähigkeit demonstriert. In Kapitel 11 wird der Ansatz dieser Arbeit auf weitere elementare Leitstandaufgaben wie *Datenzugriff* und *Historischer Datenzugriff* übertragen, konstruktiv umgesetzt und evaluiert. Damit soll neben der prinzipiellen Einsatzfähigkeit des Ansatzes auch seine Tragfähigkeit nachgewiesen werden.

Abschließend werden die Ergebnisse dieser Arbeit zusammengefasst, Vor- und Nachteile diskutiert und Anknüpfungspunkte für weitere Forschungsarbeiten aufgezeigt.

Kapitel 2

Prozesstechnische Anlagen

Mit der Erfindung der Dampfmaschine im 18. Jahrhundert und etwa gleichzeitig der mechanischen Webstühle beschritt die Menschheit den Weg der Mechanisierung, bei der menschliche Handlungen durch Maschinen nachgebildet werden [vgl. Polke 1994, S. 11; Lunze 2003, S. 3]. Die Maschinen ersetzen dabei die körperliche Tätigkeit durch eine maschinelle, wobei sie nach einem von Menschen vorgegebenen Kommando agieren. Die manuelle Steuerung der Maschinen und die dazu notwendige geistige Tätigkeit wurden mit der Verbreitung der Informationstechnologie zunehmend an Programme übertragen und somit automatisiert. 1980 wurde in der Bayer AG der Begriff *Prozessautomatisierung* eingeführt, der zunächst „. . . auf die ganzheitliche Ausrichtung der mit der Führung bzw. Leitung von verfahrenstechnischen Produktionsprogrammen Befaßten“ hinweisen sollte [Polke, 1994, S. 13]. Dieser Begriff wurde später als Synonym für die IT-Orientierung verwendet. Im Lebenszyklus technischer Anlagen dringt die IT immer weiter in neue Bereiche vor. Daraus resultieren zahlreiche Softwaresysteme, die einer groben Einordnung bedürfen, um den Schwerpunkt dieser Arbeit zu setzen.

2.1 Lebenszyklus technischer Anlagen

Technische Anlagen sind langlebige Systeme, die über mehrere Jahrzehnte, meist 20-50 Jahre, betrieben werden [Früh et al., 2009, S. 199]. In dieser

Zeit durchlaufen sie einen Lebenszyklus, der nach Freund und Haußner et al. aus den in der Abbildung 2.1 dargestellten Phasen besteht [Freund 2010; Haußner et al. 2010]. Diese Phasen werden dabei nicht strikt sequenziell durchlaufen, sondern teils parallel und im Wechsel.

Abbildung 2.1: Lebenszyklus technischer Anlagen (Quelle: eigene Erstellung)

- **Planung:** In der Planungsphase werden zwischen Anbieter und Kunden Anforderungen an das zukünftige System definiert. Dies umfasst bei größeren Projekten auch die Entwicklung digitaler Modelle der Hallenplanung, Zellen- Layouts, Produktionsabläufe etc., die mit Hilfe von Computer Aided Engineering (CAE)-Tools erstellt werden.

- **Errichtung:** In der anschließenden Errichtungsphase erfolgt die mechanische und elektrische Konstruktion der Anlage. Dabei werden Steuerungs- und Überwachungsprogramme für technische Geräte und Roboterprogramme entwickelt und getestet. Zusätzlich entstehen in dieser Phase Prozessbilder, die technische Prozesse grafisch visualisieren.

- **Inbetriebsetzung:** In der Inbetriebsetzungsphase wird das Zusammenspiel zwischen Mechanik und Software überprüft und angepasst. Diese Test- und Abnahmevorgänge sollen die letzten Fehler im Anlagenaufbau und der Steuerungsprogrammierung aufzeigen. Dazu wird die Anlage häufig an ihre Leistungsgrenze gebracht. Dabei werden die in der Ausschreibung definierten Durchsätze und Antwortzeiten überprüft. Gleichzeitig werden die späteren Bediener der Anlage geschult.

- **Betrieb und Wartung:** In der Betriebsphase übernimmt der Anlagenbetreiber die Steuerung und Überwachung der Betriebszustände seiner Anlage. Darüber hinaus werden in regelmäßigen Wartungsintervallen mechanische und steuerungstechnische Komponenten überprüft und ggf. getauscht.

- **Instandhaltung:** Die Phase Instandhaltung dient der Erhöhung der Anlagenverfügbarkeit, Verbesserung der Betriebssicherheit sowie der

Optimierung der Betriebsabläufe. Häufig werden dabei auch erhöhte Anforderungen an Durchsatz, Lagerkapazität oder ein verändertes Produktspektrum umgesetzt.

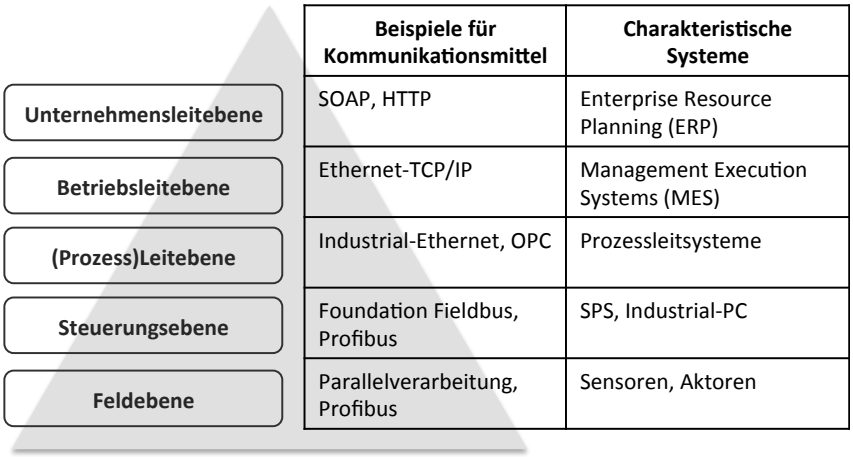
- **Recycling:** Schließlich wird die Anlage in der Recyclingphase am Ende ihres Lebenszyklus möglichst ökologisch rückgebaut und recycelt.

Für diese Arbeit sind insbesondere die Phasen von der Errichtung bis zum Betrieb und Wartung relevant. In diesen Phasen setzen Leitstandmitarbeiter vermehrt Softwaresysteme verschiedener Hersteller sowie Eigenentwicklungen ein. Aufgrund der zahlreichen Aufgaben der Leitstandmitarbeiter sowie der Vielfalt der dafür eingesetzten Softwaresysteme ist es in der Automatisierungstechnik üblich, diese entlang der Automatisierungspyramide einzuordnen.

2.2 Automatisierungspyramide

Die Automatisierungspyramide dient als Referenzmodell, um Begriffe zu vereinheitlichen, Zusammenhänge zwischen Begriffen und Automatisierungsfunktionen zu verdeutlichen und die Kommunikation zwischen den involvierten Disziplinen zu verbessern [vgl. Früh et al. 2009, S. 68; Polke 1994, S. 62; Lunze 2003, S. 32]. Je nach Genauigkeit werden dabei drei bis sechs Automatisierungsebenen unterschieden. In dieser Arbeit wird die Automatisierungspyramide mit fünf Ebenen nach Strube et al. vorgestellt, da sich die meisten Autoren in der Literatur auf diese Ebenen beziehen [Strube et al., 2012]:

- **Feldebene:** Zu den wichtigsten Aufgaben auf dieser Ebene gehören Messen und Stellen relevanter Größen sowie die Datenübertragung über Feldbussysteme. Messen erfasst im engeren Sinne analoge physikalische Größen wie Temperatur, Druck, Durchfluss und Füllstand. Das Stellen beeinflusst den Prozess, indem zumeist Stoff- oder Energieströme mittels Stellventilen oder Antrieben verändert werden.



Beispiele für Kommunikationsmittel	Charakteristische Systeme
SOAP, HTTP	Enterprise Resource Planning (ERP)
Ethernet-TCP/IP	Management Execution Systems (MES)
Industrial-Ethernet, OPC	Prozessleitsysteme
Foundation Fieldbus, Profibus	SPS, Industrial-PC
Parallelverarbeitung, Profibus	Sensoren, Aktoren

Abbildung 2.2: Automatisierungspyramide (Quelle: Strube et al., 2012)

- **Steuerungsebene:** Auf dieser Ebene sind Programme für eine automatische Regelung und Steuerung des Produktionsprozesses und Roboter angesiedelt. Diese Programme werden meist auf dedizierter Hardware ausgeführt (z. B. Speicherprogrammierbare Steuerung, SPS). Die im Produktionsprozess entstehenden Daten werden an Bedien- und Beobachtungsstationen auf der Leitebene gesendet und gleichzeitig protokolliert, archiviert und für Analysezwecke aufbereitet.

- **Leitebene:** Auf der Leitebene überwachen und steuern Operateure in Leitständen (auch Leitwarten) technische Prozesse mit Hilfe eines Prozessleitsystems. „In den zentralen Leitwarten der Prozessindustrie fließt alle Information über Energie- und Materialflüsse, Zustände und Zustandsänderungen der technischen Prozesse sowie Zielvorgaben für die Prozessführung zusammen“ [Urbas et al., 2012].

- **Betriebsleitebene:** Typische Aufgaben auf dieser Ebene betreffen die Produktionsplanung einschließlich der Einsatzplanung für Maschinen- und Anlagenbelegung mit Hilfe eines Manufacturing Execution Systems (MES). Auf dieser Ebene sind auch betriebswirtschaftliche Aufgaben wie Bilanzierung, Auftragsverfolgung, Lagerverwaltung und Personaleinsatzplanung angesiedelt.

- **Unternehmensleitebene:** Auf dieser Ebene sind unternehmensstrategische Aufgaben wie Unternehmensführung, Produkt- und Vertriebs-

planung sowie Kostenüberwachung mit Hilfe von Enterprise Ressource Planning (ERP) Systemen angesiedelt.

In dieser Arbeit liegt der Schwerpunkt auf der Leitebene. Auf dieser überwachen und steuern Operateure technische Abläufe mit Hilfe eines Prozessleitsystems. Die historische Entwicklung der Prozessleitsysteme hat zu vielen Verbesserungen, aber auch neuen Problemen geführt, die in dieser Arbeit adressiert werden. Im Folgenden werden daher Prozessleitsysteme und ihre Entstehung kurz vorgestellt.

2.3 Prozessleitsystem

Während zu Beginn der Leittechnik Arbeiter Prozessgrößen noch direkt an den technischen Geräten ablasen, führte die technologische Weiterentwicklung zunehmend zu einer räumlichen Trennung der Operateure vom Ort des Prozessgeschehens. Dabei wurden in einem ersten Schritt mehrere Messstellen in Leitständen zusammengefasst, wobei zunächst nur der Ort, nicht jedoch die Art der Anzeigen geändert wurde. Für solche Leitstände haben sich daher Begriffe wie „[. . .] Rollschuhwarten, mit scheinbar endlosen Reihen von Einzelanzeigen“ [Urbas et al., 2012] sowie *Ein-Sensor-Eine-Anzeige-Prinzip* (SSSI - single sensor single indicator) durchgesetzt [Kindsmüller, 2006, S. 12ff]. Die steigende Anzahl an Sensoren verursachte schließlich Raum- und Übersichtsprobleme, da zahlreiche Anzeigeinstrumente gleichzeitig um die Aufmerksamkeit der Operateure konkurrierten (Abbildung 2.3).

Mit der Einführung der Informationstechnologie wurde begonnen, die Daten mehrerer Sensoren in einem Anzeigeinstrument zusammenzufassen. Diese Veränderung verlief in mehreren Iterationen und hat zu Übergangslösungen geführt, in denen sich analoge und digitale Anzeigesysteme zunächst vermischten[1] (Abbildung 2.4).

[1] Diese Vermischung war nicht nur auf technische Anlagen begrenzt. In Flugzeugcockpits vermischen sich beispielsweise elektronische Bildschirme, klassische Anzeigeinstrumente und analoge haptische Steuerungsinstrumente. In Automobilcockpits vermischen sich digitale Anzeigen wie z. B. Navigationssysteme mit klassischen analogen Instrumenten wie Drehzahl- und Geschwindigkeitsanzeige [Herczeg, 2008].

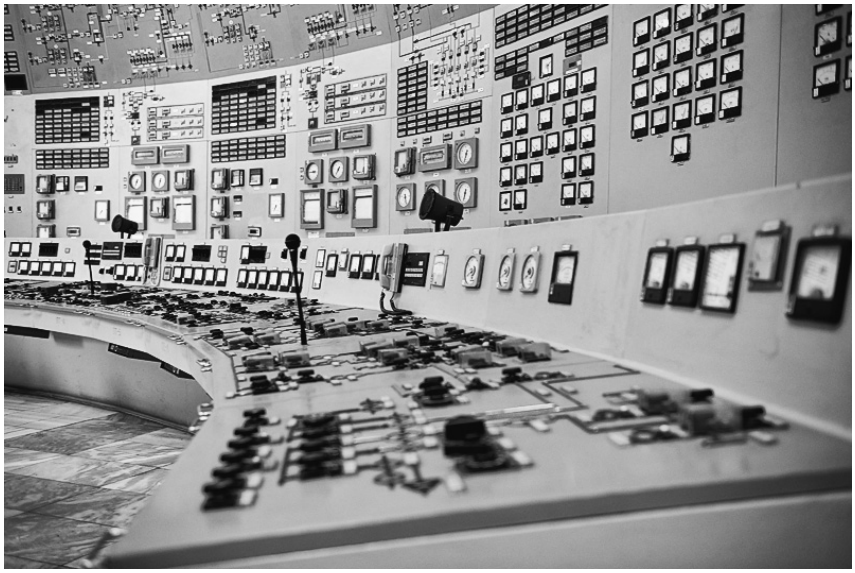
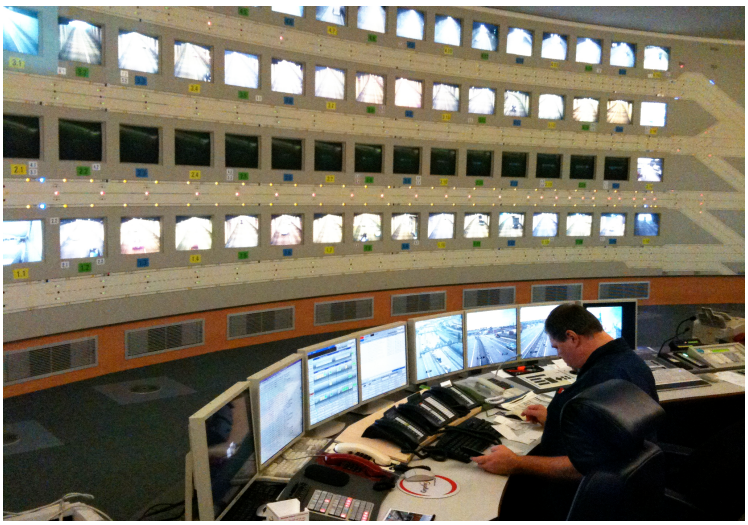


Abbildung 2.3: Ein-Sensor-eine-Anzeige-Prinzip (Foto: Yovko Lambrev)

In modernen Leitständen verschwinden analoge Instrumente. Sensordaten werden elektronisch übertragen und auf Monitoren oder Projektionswänden dargestellt (siehe Abbildung 2.5). Die Verbreitung der Informationstechnologie in technischen Anlagen führte zu sog. *Prozessleitsystemen (PLS)*, die alle Prozessleitfunktionen und Daten an einer Stelle konzentrieren. „Prozessleitsysteme [. . .] stellen den Kern moderner Automatisierungslösungen für die verfahrenstechnische Industrie dar“ [Früh et al., 2009, S.191]. Sie sind auf der Leitebene der Automatisierungspyramide angesiedelt. Mit der Popularität der Prozessleitsysteme wuchs auch die Anzahl der verwendeten domänen- und herstellerspezifischen Kommunikationsprotokolle, Bussysteme und Gerätetreiber. Ihre Wartung und Entwicklung forderte einen Großteil der Ressourcen in Unternehmen ein [Lange et al., 2010, S. 5]. Im Jahr 1995 begann die OPC-Foundation[2], ein Unternehmenskonsortium aus Hard- und Softwareherstellern, die Datenübertragung und Datenverarbeitung zu standardisieren. Dies resultierte in der Veröffentlichung der OPC-Spezifikation mit folgenden Standards [Lange et al., 2010, S. 5ff]:

[2]OPC = OLE for Process Control. OLE (Object Linking and Embedding) ist eine Microsoft-Technologie, mit der Echtzeitdaten unter Windows-Betriebssystemen übertragen werden.

(a) Kanadisches Atomkraftwerk mit überwiegend analogen Anzeigen (Quelle: Mumaw et al., 2000)



(b) Elbtunnel Hamburg mit zunehmend digitalen Anzeigen

Abbildung 2.4: Vermischung analoger und digitaler Anzeigen in Leitständen

- **Datenzugriff (engl. Data Access, DA)** definiert die grundsätzlichen Mechanismen und Funktionalitäten zum Lesen und Schreiben von Prozessdaten.
- **Alarmierung (engl. Alarms and Events, A&E)** definiert, wie Clients über das spontane Auftreten kritischer Ereignisse innerhalb eines Prozesses durch einen Server informiert werden.
- **Historischer Datenzugriff (engl. Historical Data Access, HDA)** definiert, wie historische Daten für Analysezwecke und Dokumentation bereitgestellt und abgelegt werden.

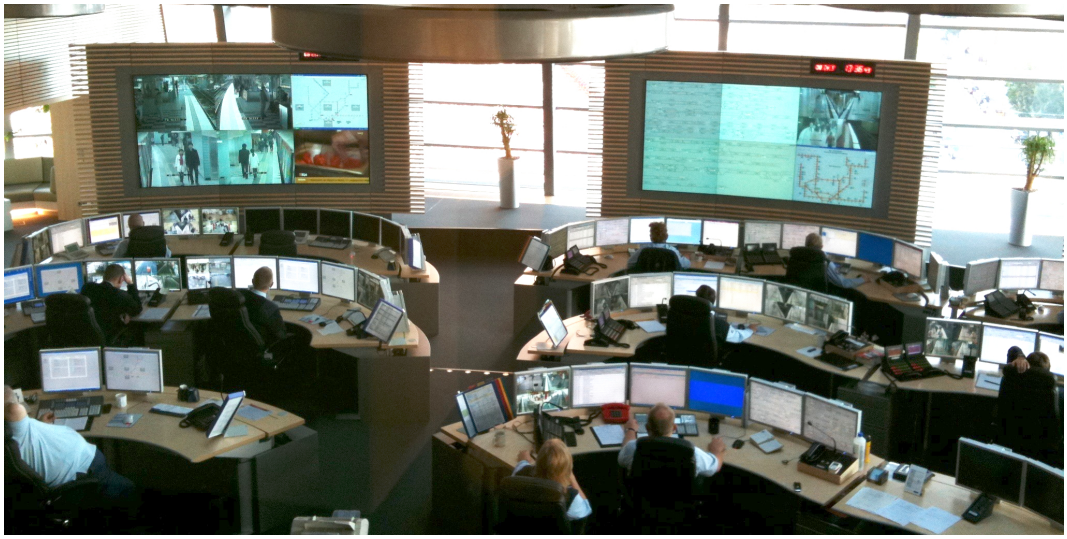


Abbildung 2.5: Leitstand Hamburger Hochbahn

Die OPC-Spezifikation ist aktuell ein De-facto-Standard, der in zahlreichen Industrieanwendungen umgesetzt wurde. Neben der OPC-Spezifikation existieren noch andere, überwiegend domänenspezifische Alternativen wie z. B. das Experimental Physics and Industrial Control System (EPICS) [Dalesio et al., 1994].

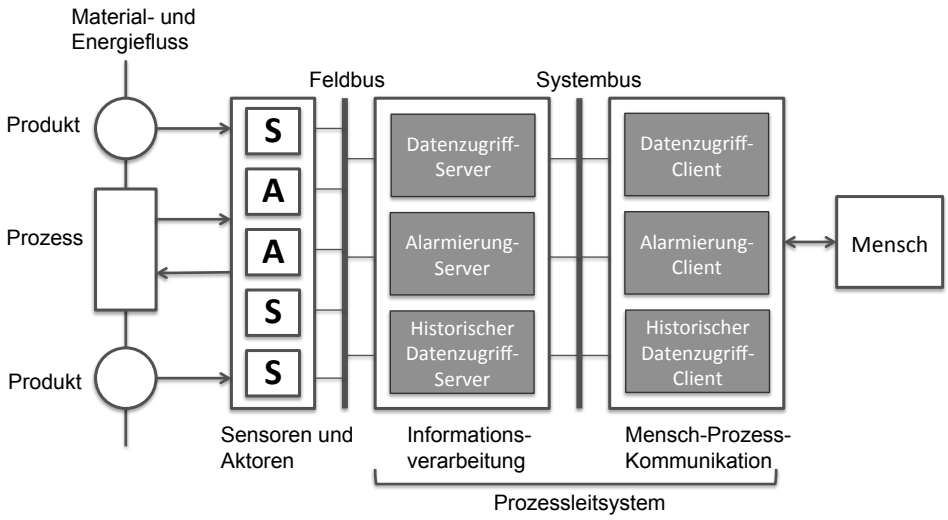


Abbildung 2.6: Prozessleitsystem (Quelle: In Anlehnung an Polke, 1994, S. 339, ergänzt um die Client/Server-Komponenten)

Prozessleitsysteme bestehen zum einen aus prozessnahen Rechnern, die an einen Feldbus angeschlossen sind und entsprechend der OPC-Spezifikation in

dedizierten Servern Sensor-Daten verarbeiten. Zum anderen bestehen Prozessleitsysteme aus Bedien- und Beobachtungsstationen, in denen Operateure über eine Leitstandsoftware mit der Anlage interagieren (siehe Abbildung 2.6) [vgl. Früh et al., 2009, S. 192ff]. Zu den elementaren Leitstandaufgaben der Operateure gehören u.a. Zugriff auf Echtzeitdaten, Überwachung der Alarme und Auswertung historischer Daten. In dieser Arbeit werden Verbesserungsvorschläge für diese Aufgaben präsentiert, daher werden die elementaren Leitstandaufgaben in den folgenden Abschnitten kurz vorgestellt.

2.3.1 Datenzugriff und Visualisierung

Operateure interagieren mit der Anlage mit Hilfe sog. *Prozessbilder* (auch Fließbilder). Prozessbilder verbinden den aktuellen Prozesszustand sowie die Prozesstopologie in einer Darstellung und ermöglichen Operateuren, Messwerte abzulesen und über Bedienelemente in den Prozess steuernd einzugreifen [Polke, 1994, S. 683]. Sie sind grafische Schnittstellen zum realen Prozess und bestehen grundsätzlich aus zwei Teilen [Früh et al., 2009, S. 172]:

- einem statischen Teil, der aus einer graphischen Darstellung mit Apparaten und Rohrleitungen besteht;
- einem dynamischen Teil mit Echtzeitdaten für z. B. numerische Anzeigen von Messwerten, Säulen und Balken sowie Status von Maschinen.

In der Abbildung 2.7 ist exemplarisch ein Prozessbild des Kryogenik-Moduls abgebildet, das beim DESY zur Kühlung der Teilchenbeschleuniger eingesetzt wird. Bei der Prozessbild-Erstellung, auch *Projektierung* genannt, verknüpfen sog. Projekteure[3] grafische Symbole mit dazugehörigen Anlagenteilen über eindeutige Bezeichner. Diese Bezeichner werden entsprechend einem Kennzeichnungssystem benannt, wie z. B. dem Kraftwerkskennzeichnungssystem [KKS, 1978]. Für diese eindeutigen Bezeichner existieren diverse Begriffe wie z. B. Datenpunkt, Messpunkt oder Prozessvariable. Im englischen Sprachraum

[3] Projekteure und Operateure sind in diesem Kontext, als Rollen aufzufassen, die in der Praxis durch verschiedene oder die gleiche Person ausgefüllt werden können.

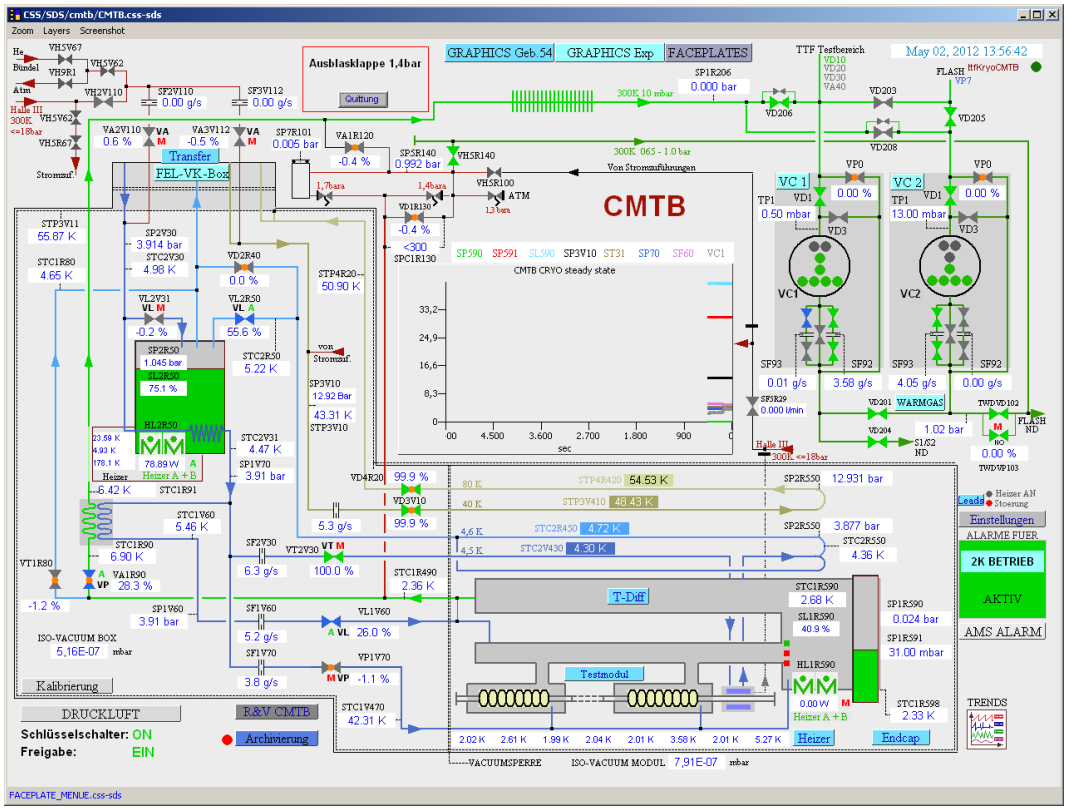


Abbildung 2.7: CMTB Prozessbild (Quelle: DESY)

sind auch die Begriffe Tag, Channel und Record verbreitet. Trotz dieser Begriffsvielfalt sind die dahinter stehenden Konzepte vergleichbar, daher wird in dieser Arbeit stellvertretend der Begriff *Prozessvariable* verwendet.

BEGRIFF 2.1: PROZESSVARIABLE

Eine Prozessvariable repräsentiert auf einer abstrakten Ebene Messwerte realer Anlagenteile aber auch Berechnungen, die sich aus Messwerten mehrerer Prozessvariablen zusammensetzen [vgl. Cupek et al., 2009]. Sie besteht mindestens aus den Elementen Bezeichner, Datentyp und Variablenwert und kann zusätzliche Metadaten wie Messeinheit (z. B. °C, kg, oder U/Min) sowie den Betriebsbereich mit zulässigen Grenzwerten (z. B. 100-200 C°, 800-6000 U/Min) enthalten [vgl. Epple 2003; Mahnke et al. 2009, S. 19]. Der Bezeichner einer Prozessvariable wird in jedem Unternehmen individuell, entsprechend einem Kennzeichnungssystem wie z. B. KKS [1978] oder IEC 61346 [García et al., 2004], benannt [vgl. Polke, 1994, S. 618f].

Ingenieure organisieren technische Systeme üblicherweise in Subsysteme und Komponenten [García et al., 2004]. Dies spiegelt sich auch in der Benennung der Prozessvariablen wider z. B. `WK:K:42b:1:BI` oder `35L216_ao`. Die ersten Buchstaben bzw. Ziffern der Prozessvariable repräsentieren meist Gebäudenummern oder Teilanlagen wie z. B. Elektro- oder Kryotechnik, gefolgt von Anlagenteilen bis hin zu einzelnen Sensoren und Aktoren, die durch Kürzel wie ai/ao (analog input/output) oder bi/bo (binary input/output) typisiert werden. Prozessvariablen werden im Prozessleitsystem konfiguriert und auf die Hardware-Adressen realer Anlagenteile abgebildet.

2.3.2 Alarmierung

In den meisten technischen Anlagen treten wiederkehrend Störungen auf, wie z. B. Temperaturüberschreitungen, Änderungen in der Produktqualität oder Störungen in der Stromversorgung [Vicente et al., 2001, S. 833]. Diese Störungen werden von Sensoren gemessen und an das Prozessleitsystem übertragen. In diesem werden sie als Überschreitungen von Schwellwerten erkannt und überwiegend automatisch aufgefangen und behoben. Es treten jedoch wiederkehrend außergewöhnliche Betriebszustände oder Anlagenausfälle auf, die das Prozessleitsystem allein nicht bewältigen kann. Die Überwachung und Bearbeitung all dieser Störungen gehört zu der wichtigsten Aufgabe der Operateure im Leitstand [Liu et al., 2003]. Alarme sind Signale an Operateure, die auf außergewöhnliche Betriebszustände hinweisen und ein Eingreifen der Operateure erfordern [NAMUR, 2003]. Die Norm ANSI/ISA-18.2 definiert den Alarmbegriff wie folgt [ANSI/ISA-18.2, 2009, S. 21]:

BEGRIFF 2.2: ALARM
. . . an audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a response. The essential element of this definition is the response to the alarm.

Ingenieure konfigurieren Alarme, indem sie Schwellwerte für Situationen definieren, in denen eine Anlage nicht im ökonomischen und mechanischen

Optimum betrieben wird oder gar Anlagenausfälle drohen, die Mensch und Umwelt gefährden [EEMUA, 2007, S. 10]. Überschreitungen der Schwellwerte lösen Alarme aus, die Operateuren häufig in Alarmtabellen präsentiert werden. In der Abbildung 2.8 ist exemplarisch eine DESY-Alarmtabelle dargestellt. Diese enthält neben Alarmen auch Warnungen und Hinweise, die kein Eingreifen der Operateure erfordern und somit nach der ANSI/ISA-18.2-Norm keine Alarme sind. Diese Aspekte werden ausführlicher im nächsten Kapitel diskutiert.

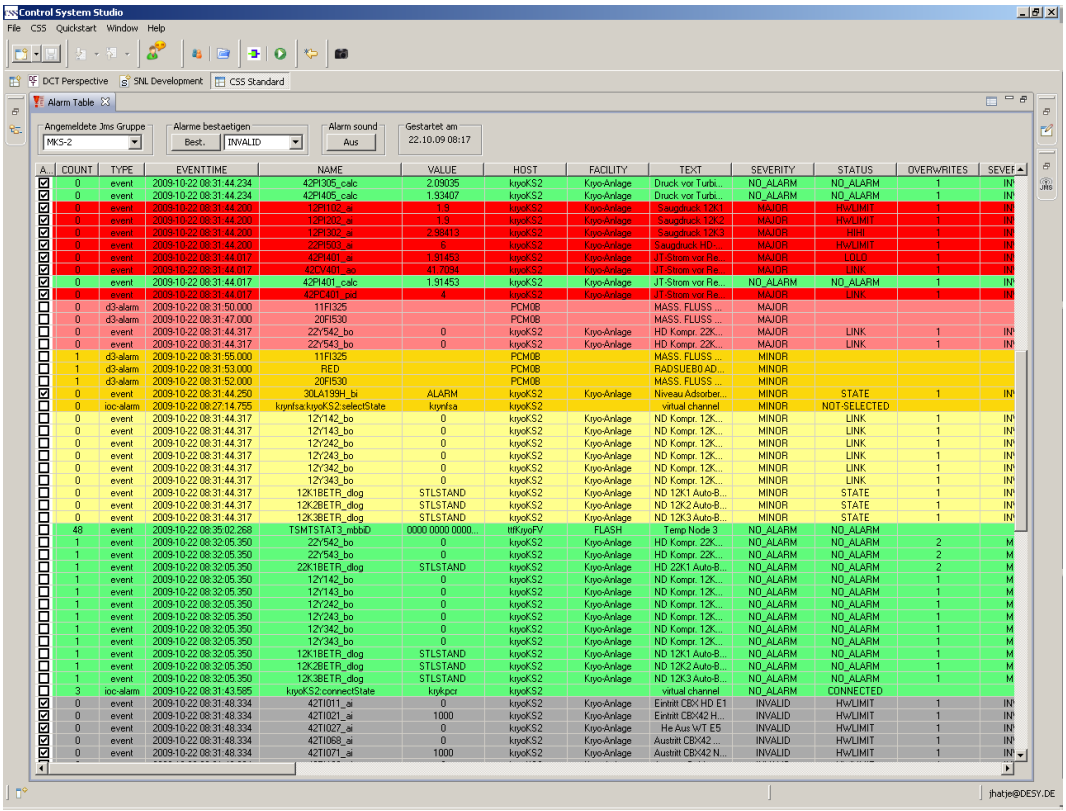


Abbildung 2.8: Alarmtabelle (Quelle: DESY)

Alarme und Alarm-Werkzeuge unterstützen Operateure, die folgenden Ziele zu erreichen [EEMUA, 2007]:

- die Anlagenverfügbarkeit zu erhöhen, in dem sie auf potentielle Gefahren hinweisen, bevor Notabschaltungsmechanismen greifen;

- die Wirtschaftlichkeit der Produktion sicher zu stellen, in dem sie auf abnormale Situationen hinweisen, die zu Produktschäden und damit zu finanziellen Ausfällen führen können;
- komplexe Prozessabläufe besser zu verstehen, um in Notfällen geeignet zu reagieren.

Um diese Aufgaben angemessen zu erfüllen, sollen Alarme nach der EEMUA visuell und auditiv *alarmieren*, über die Alarmursache *informieren* und Operateure dabei *anleiten*, die Ursache zu beheben [EEMUA, 2007].

2.3.3 Historischer Datenzugriff

Für die Auswertung historischer Daten verwenden Operateure Kurvenbilder oder auch Trenddarstellungen, die beispielsweise Temperatur- oder Druckverläufe in einem gewünschten Zeitraum darstellen (siehe Abbildung 2.9). Während zu Beginn der Prozessautomatisierung mechanische Kurvenschrei-

Abbildung 2.9: Kurvenbild (Quelle: DESY)

ber verbreitet waren, verwenden moderne Prozessleitsystemen überwiegend digitale Anzeigen. Diese bieten verbesserte Anzeige- und Interaktionsmöglichkeiten. Dazu gehören die Bildvergrößerung (zoomen), die flexible Anpassung

der Zeiträume sowie die Überlagerung mehrerer Zeitreihen. Dadurch erkennen Operateure, wie die Änderung einer Prozessgröße sich auf andere Größen auswirkt. Darüber hinaus zeigen Kurvenbilder Extremwerte sowie die Schwankungsbreite zwischen Ist- und Soll-Werten [Früh et al., 2009, S. 175]. Kurvenbilder werden für die Auswertung historischer Daten eingesetzt, da sie viele atomare Daten verdichtet visualisieren und dadurch eine schnelle Erfassung von Abweichungen, Tendenzen oder Zyklen sowie Vergleich zwischen Größen unterstützen [Kindsmüller, 2006, S. 51].

2.4 Zusammenfassung

In diesem Kapitel wurde der Fokus dieser Arbeit zunächst entlang des Lebenszyklus technischer Anlagen und anschließend entlang den Ebenen der Automatisierungspyramide eingeordnet. Im Mittelpunkt dieser Arbeit steht die Betriebsphase, in der Operateure technische Prozesse über ein Prozessleitsystem steuern und überwachen. Prozessleitsysteme bestehen aus prozessnahen Komponenten auf der Steuerungsebene und den Bedien- und Beobachtungsstationen auf der Leitebene der Automatisierungspyramide. Prozessleitsysteme ermöglichen Anlagenbetreibern, komplexe Regelungs- und Optimierungsverfahren einzuführen, die zunehmend menschliche Aufgaben übernehmen. Dabei streben die Ingenieure danach, automatisierte Systeme zu entwickeln, die „. . . die aktuell zu erfüllenden Aufgaben selbstständig finden und daraus die durchzuführenden Lösungsschritte ableiten, also autonom (ohne Eingriff eines Menschen) arbeiten“ [Lunze, 2003, S. 1]. Automatisierte Systeme sind gerade bei Routineaufgaben weniger fehleranfällig als Operateure. Gleichzeitig sind sie nicht in der Lage, Lösungen für außergewöhnliche und unvorhersehbare Betriebszustände zu finden. Diese Fähigkeit gehört zu den menschlichen Stärken [Nachreiner et al. 2006, S. 8; Kurz 2008].

Parasuraman und Riley sowie Sheridan stellen fest, dass in der Praxis tendenziell all das automatisiert wird, was technologisch und ökonomisch einfach zu automatisieren ist. Der Rest bleibt Aufgabe der Operateure [Sheridan 2012, S. 1011; Parasuraman und Riley 1997, S. 232]. Dieser Rest ist nicht selten unstimmig und komplex, was die Arbeit der Operateure erheblich erschwert.

Mit der zunehmenden Automatisierung wächst in heutigen Leitständen die Anforderung, eine ausgewogene Arbeitsteilung zwischen Mensch und Maschine zu finden, bei der sich beide entsprechend ihren jeweiligen Stärken ergänzen [vgl. Sheridan 1996; Parasuraman und Riley 1997]. Bei der sog. *human-centered automation* rücken Operateure wieder stärker in den Mittelpunkt des Prozessgeschehens. Sie benötigen dafür Softwaresysteme, die menschliche Arbeit systematisch adressieren und besser unterstützen [vgl. Parasuraman und Riley 1997, S. 248; Schwarz et al. 2012]. Die *human-centered automation* bildet in dieser Arbeit die Grundlage für eine anwenderorientierte Leitstandmodernisierung und wird im nächsten Kapitel vorgestellt.

Kapitel 3

Anwendungsorientierung in der Automatisierungstechnik

Mit der Einführung der Prozessautomatisierung verfolgten Ingenieure das Ziel, menschliche Arbeit zu minimieren, um technische Anlagen ökonomischer und zuverlässiger zu betreiben [Lunze, 2003, S. 14]. Mit dem höheren Automatisierungsgrad wurden Menschen zunehmend aus dem Prozessgeschehen verdrängt und zu passiven Zuschauern degradiert. Sie sollen nur noch in Ausnahmesituationen eingreifen, wenn die Automatisierung versagt [vgl. z. B. Emigholz, 1996, S. 8]. Hoc et al. stellen jedoch fest, dass mit der wachsenden Komplexität der Systeme die menschliche Arbeit praktisch unverzichtbar wird [Hoc et al., 1995, S. xi]. Dafür führen sie zwei wichtige Gründe an:

1. Es ist technisch und ökonomisch oft nicht möglich, jede Funktion eines Systems zu automatisieren.

2. Selbst mit zunehmendem technischen Fortschritt ist es für Menschen nach wie vor unmöglich, in einem automatisierten System alle potentiellen Zustände und Bedingungen zu antizipieren, die im Lebenszyklus technischer Systeme auftreten können.

Eine vollständige Automatisierung aller prozesstechnischen Aufgaben ist bislang nicht möglich. Die Prozessführung erfordert somit gezielte Eingriffe

durch Operateure, die auch als Anlagenfahrer bezeichnet werden. „Der Anlagenfahrer kann kritische Anlagenzustände im Vorfeld abwenden. Er hat die Fähigkeit, komplexe und unklare Informationen zu verstehen und in Aktionsentscheidungen zu integrieren“ [Kurz, 2008, S. 86]. Dieser kognitiven menschlichen Fähigkeit wirkt nach Sheridan und Parasuraman et al. die andauernde Passivität der Operateure entgegen, die eine Folge des hohen Automatisierungsgrades ist [vgl. Sheridan 1996, S. 8; Parasuraman et al. 2000, S. 290f]. In der Praxis werden wiederholt folgende, mit der Automatisierung einhergehende Phänomene beobachtet [vgl. Parasuraman et al. 2000, S. 290f; Sheridan 1996, S. 8]:

1. **Reduziertes Situationsbewusstsein:** Menschen nehmen Veränderungen im System weniger wahr, wenn die Reaktion auf diese Änderungen prinzipiell im Aufgabenbereich einer Maschine liegt.

2. **Nachlässigkeit:** Operateure vertrauen zu stark der Automatisierung und vernachlässigen ihre Überwachung und Kontrolle.

3. **Degeneration des Expertenwissens:** Operateure verlernen die Fähigkeit, auf kritische Situationen zu reagieren, wenn die überwiegende Entscheidungskompetenz an Maschinen übertragen wird.

Nach Endsley und Connors sind Operateure, quer durch alle relevanten Industrien und Anwendungsklassen, für 60-85% aller auftretenden Unfälle verantwortlich [Endsley und Connors, 2008, S. 3]. Ein Großteil der von Operateuren verursachten Unfälle ist jedoch weniger auf den menschlichen Faktor, sondern vielmehr auf ein zu technologiezentriertes Design der Systeme zurückzuführen [Endsley und Connors, 2008, S. 4]. Parasuraman und Riley argumentieren ferner, dass die Fehlerquelle Mensch nicht durch einen höheren Automatisierungsgrad aus einem System verdrängt werden kann. Dadurch wird lediglich die Fehlerquelle von den Operateuren auf die Konstrukteure des Automatisierungssystems verlagert [Parasuraman und Riley, 1997, S. 247]. Die Automatisierung verdrängt somit nicht die menschliche Arbeit, sondern verändert sie auf eine Weise, die von den Designern von Automatisierungslösungen bislang nur unzureichend berücksichtigt wurde [vgl. Nachreiner et al. 2006, S. 8; Emigholz 1996, S. 154; Parasuraman und

Riley 1997, S. 231]. Leitstandsysteme erfordern nach wie vor gezielte korrigierende Eingriffe durch Menschen. Gleichzeitig sind viele Leitstandsysteme stark technologiegeprägt und unterstützen nur unzureichend die menschliche Arbeitsweise [vgl. Parasuraman und Riley 1997; Sheridan 2012; Hoc et al. 1995].

Diese Erkenntnis hat in der Automatisierungstechnik einen Umdenkprozess eingeleitet, bei dem die Aufgabenverteilung zwischen Mensch und Maschine neu diskutiert wird [vgl. z. B. Parasuraman et al., 2000]. „Underlying this recognition is the understanding that only human operators can be granted fiduciary responsibility for system safety, so the human opertor should be at the heart of the system, with full authority over all it's functions“ [Parasuraman und Riley, 1997, S. 248]. Bei der sog. *human-centered automation* rücken Operateure deutlich mehr in den Mittelpunkt des Prozessgeschehens und übernehmen eine aktive Rolle [vgl. Hoc et al. 1995; Parasuraman et al. 2000; Endsley und Connors 2008; Sheridan 2012; Emigholz 1996]. Dabei ist bislang nicht eindeutig geklärt, wie Menschen und Maschinen einander ergänzen und zusammenarbeiten sollen. Früh et al. plädieren grob dafür, soviel Automatisierungstechnik einzusetzen wie nötig [Früh et al., 2009, S. 8]. Sheridan macht hingegen zehn konkrete Vorschläge, wie die Arbeitsteilung zwischen Mensch und Maschine aussehen könnte [Sheridan, 2012, S. 1012]. Im Kern der Argumentation schlägt er vor:

> *„Allocate to the human the tasks best suited to the human, allocate to the automation the tasks best suited to it“ [Sheridan, 2012, S. 1012].*

Eine vergleichbare Diskussion wurde Ende der 80er Jahre mit der Entwicklung des *Personal Computers* geführt. Damals argumentierte Kay hinsichtlich der Aufgabenverteilung zwischen Mensch und Computer: „Do not automate the work you are engaged in, only the materials“ [Kay, 1977, S. 13]. Wenn Menschen beispielsweise gerne malen, soll man nicht das Malen automatisieren, sondern eher ein Programm mit neuen Malwerkzeugen und Farben entwickeln.

Parasuraman et al. empfehlen bei der Aufgabenverteilung zwischen Mensch und Maschine, die Informationserfassung und Aufbereitung weitgehend zu automatisieren, aus den Informationen Handlungsalternativen abzuleiten und in Aktionen umzusetzen, jedoch einem Menschen zu überlassen [Parasuraman et al., 2000]. Sheridan führt an, dass eine optimale Aufgabenverteilung zwischen Mensch und Maschine sehr schwierig und stark situationsabhängig ist [Sheridan, 1996]. Diese Unsicherheit spiegelt sich auch in den unterschiedlichen Herangehensweisen wider, wie die *human-centered automation* im Hinblick auf die elementaren Leitstandaufgaben erzielt werden kann. Jede Disziplin im Bereich der Automatisierungstechnik wie z. B. die Ingenieurswissenschaft, Arbeitspsychologie oder Informatik, hat eine eigene Sichtweise auf die *human-centered automation*, wobei auch innerhalb der jeweiligen Disziplin mannigfaltige Ansichten existieren. In jeder Disziplin werden zahlreiche Vorschläge präsentiert, die eine Skala von *technologie- und datenzentriert* bis *anwenderorientiert* abdecken. Die technologie- und datenzentrierten Ansätze sind bestrebt, wiederkehrende Leitstandprobleme durch z. B. umfassende Daten, ausgefeilte Konfigurationen sowie leistungsfähige Technik zu lösen. Hingegen betrachten anwenderorientierte Ansätze Technologien aus Sicht der Benutzer und stellen die Gebrauchstauglichkeit der Software in den Vordergrund. Anwenderorientierte Ansätze sind mit einer Wertvorstellung verbunden, die qualifizierte menschliche Arbeit als unersetzlich betrachtet. Charakteristisch für diese Wertvorstellung ist eine unterstützende Sichtweise, bei der Software Menschen hilft, anstehende Aufgaben durch ihre Erfahrung, Kreativität und ihr Können zu erledigen [vgl. Züllighoven et al. 1998, S. 258; Endsley und Connors 2008].

Technologiezentrierte und anwenderorientierte Ansätze verfolgen im Kern das gemeinsame Ziel, einen ökonomischen, zuverlässigen und sicheren Anlagenbetrieb zu ermöglichen und setzen dafür technische Hilfsmittel ein. Sie nähern sich diesem Ziel jedoch aus unterschiedlichen Richtungen mit abweichenden Herangehens- und Sichtweisen. Gleichzeitig existieren keine allgemein akzeptierten Kriterien, die eine eindeutige Unterscheidung zwischen diesen Sichtweisen ermöglichen, wodurch ihre Übergänge oft weich sind. Betrachtet man jedoch die Extreme technologiezentrierter und anwenderorientierter Sichtweisen, resultieren daraus gravierende Unterschiede für

eine Leitstandmodernisierung. In den folgenden Abschnitten werden für die elementaren Leitstandaufgaben aktuelle und in der Literatur viel diskutierten Lösungsansätze herausgearbeitet und der Kern dieser Ansätze einer anwenderorientierten oder technologiezentrierten Sichtweise zugeordnet, um den Standpunkt dieser Arbeit zu verdeutlichen.

3.1 Datenzugriff und Visualisierung

Wie im Abschnitt 2.3.1 beschrieben, überwachen und steuern Operateure technische Anlagen mittels Prozessbildern und Kurvenbildern. Diese Werkzeuge visualisieren Prozesszustände und müssen für jede Anlage individuell entwickelt und gepflegt werden. Die Entwicklung der Prozessbilder wird auch als *Projektierung* und Operateure in diesem Kontext als *Projekteure* bezeichnet. Für diese Aufgabe existieren in der Literatur datenzentrierte Ansätze, die primär auf die Darstellung der Prozesszustände fokussieren. Polke und Früh et al. plädieren beispielsweise für die Verwendung standardisierter Symbole für Anlagenteile in Prozessbildern sowie eine einheitliche Darstellung der Messwerte mit ihrer physikalischen Einheit und mit einheitlichen Farben [Polke 1994, S. 705; Früh et al. 2009, S. 173]. Balken- und Säulendarstellungen helfen nach Polke zusätzlich, komplexe Prozesszustände einfacher zu erfassen. Früh et al. adressieren die Interaktion der Operateure mit Prozessbildern und schlagen Maßnahmen vor, bedienbare und nicht bedienbare Anlagenteile in Prozessbildern unterschiedlich zu gestalten, um Operateuren im Betrieb ihre Eingriffsmöglichkeiten explizit aufzuzeigen und sie dadurch bei der Bedienung anzuleiten [Früh et al., 2009, S. 173].

Nähert man sich der Projektierung aus einer anwenderorientierten Sicht, bergen nach Sauer und Ebel, Schleipen, Früh et al. und Urbas et al. die manuelle Projektierung und kontinuierliche Pflege der Prozessbilder, die weitaus größeren Herausforderungen für Projekteure [vgl. Sauer und Ebel 2007b; Schleipen 2008; Früh et al. 2009, S. 171; Urbas et al. 2011]. Diese erstellen Prozessbilder überwiegend von Hand, durch Copy&Paste-Operationen vorgefertigter grafischer Symbole. Dazu platzieren Projekteure die Symbole auf Prozessbilder und ergänzen sie mit Verbindungsinformationen zur Hardware,

Interaktionsmechanismen und dynamischer Visualisierung in Alarmsituationen [vgl. Schmitz und Epple 2007; Sauer und Ebel 2007b, S. 335]. Die manuelle Projektierung ist sehr zeitaufwändig und fehleranfällig. Zudem müssen Prozessbilder im Lebenszyklus der Anlage wiederkehrend angepasst und mit der realen Welt synchronisiert werden [Sauer und Ebel 2007b, S. 331; Urbas und Doherr 2011, S. 627]. In der Literatur werden daher Ansätze diskutiert, die Projektierung zu vereinfachen. Diese Ansätze können grob in „vollautomatische“ und „halbautomatische“ Projektierung unterteilt werden:

- **Vollautomatische Projektierung:** Vollautomatische Ansätze nutzen für die Projektierung Anlagendaten aus früheren Planungsphasen, die beispielsweise in Computer Aided Engineering (CAE)-Werkzeugen vorliegen [Schmitz und Epple 2007; Schleipen 2008; Urbas et al. 2011; Obst et al. 2012]. Export- und Transformationsfunktionen überführen die CAE-Daten automatisch in Prozessbilder, die Projekteure anschließend manuell nachbereiten können.

- **Halbautomatische Projektierung:** Halbautomatische Ansätze führen ein Objekt-Typ-Konzept ein, bei dem Projekteure ähnliche Anlagenteile einem Typ zuordnen können. Statt Hunderte oder gar Tausende individuelle Geräte zu konfigurieren, nehmen Projekteure die gewünschten Einstellungen auf der Typ-Ebene vor und übertragen diese anschließend auf konkrete Anlagenteile [Bratthall et al., 2002]. Dadurch können Projekteure beispielsweise mehrere, herstellerspezifische Temperaturfühler dem Standard-Typ PT100 zuordnen und zentral konfigurieren. Die Typbasierte Konfiguration ermöglicht eine nachträgliche Rekonfiguration der Anlagenteile an einer zentralen Stelle.

Die vollautomatische Projektierung verwendet CAE-Daten früherer Planungsphasen. Diese Daten wurden ursprünglich ohne die Absicht erzeugt, für eine automatische Projektierung wiederverwendet zu werden. Daher müssen viele Informationen in CAE-Werkzeugen zunächst ergänzt und überarbeitet werden, bevor eine vollautomatische Projektierung akzeptable Ergebnisse liefert [Urbas und Doherr, 2011]. Da sich technische Anlagen über die Zeit verändern, müssen Prozessbilder wiederkehrend angepasst werden. Bei der vollautomatischen Projektierung können Projekteure diese Änderungen zunächst in den

CAE-Werkzeugen nachpflegen und daraus erneut vollautomatisch Prozessbilder generieren. Diese müssen die Projekteure anschließend manuell nachbereiten. Alternativ können Projekteure die Änderungen auch direkt in den Prozessbildern nachpflegen, wodurch jedoch die Daten in CAE-Werkzeugen entweder über die Zeit erodieren oder mit viel Aufwand parallel gepflegt werden müssen. Diese, bislang nur ansatzweise gelösten Probleme erschweren aktuell den Einsatz vollautomatischer Projektierungsansätze. Die halbautomatische Projektierung ist im direkten Vergleich deutlich ausgereifter und wird bereits kommerziell eingesetzt [Bratthall et al., 2002]. Sie weist zwar einen höheren Anteil manueller Tätigkeiten auf, liefert dafür adäquate Ergebnisse und birgt deutlich weniger Risiken. Dieser Ansatz setzt ein Modell der realen Anlage voraus, das Anlagenteile mit dazugehörigen Typinformationen bereitstellt.

3.2 Alarmierung

Wie im Abschnitt 2.3 beschrieben, hat sich in der Vergangenheit die Rolle und Arbeitsumgebung der Operateure in technischen Anlagen signifikant verändert. Während Operateure früher Abweichungen von normalen Betriebszuständen direkt in der Anlage überwacht und korrigiert haben, werden in modernen Anlagen Alarme elektronisch erfasst und über Bus-Technologien an zentrale Leitstände übertragen. In großen Anlagen kooperieren Operateure in räumlich verteilten Leitständen, da ein Operateur alleine nicht die gesamte Anlage steuern und überwachen kann. Die veränderte Rolle und Arbeitsumgebung der Operateure wartet mit neuen Verbesserungen, aber auch Problemen auf. Diese können grob auf die Konfiguration der Alarme und ihre Entstehung in abnormalen Betriebszuständen sowie auf die kooperative Alarmbehandlung und Abstimmung zwischen Operateuren zurückgeführt werden. Aufgrund der hohen Relevanz der Alarmierung für den sicheren Anlagenbetrieb werden die skizzierten Probleme differenziert und im Folgenden eingehender untersucht.

3.2.1 Alarmkonfiguration

Zu Zeiten der analogen Leittechnik war es sehr zeit- und kostenintensiv, Alarme zu generieren, da dies häufig mit mechanischen und baulichen Veränderungen in Leitständen einherging [Hollifield und Habibi, 2007, S. 8]. In der modernen Leittechnik können Alarme deutlich günstiger produziert werden, was in der Praxis zu diversen Problemen führt, die in folgende Problemklassen eingeteilt werden können:

- **Unnötige Alarme (engl. nuisance alarms):** Zu dieser Kategorie gehören Alarme, die wiederkehrend erscheinen, jedoch für Operateure nutzlos sind. Dabei handelt es sich häufig um bedeutungslose Informationen, die nach der ANSI/ISA-18.2-Definition keine Alarme sind. Weitere unnötige Alarme sind die sog. *stehenden Alarme (engl. stale alarms)*. Diese bleiben über einen längeren Zeitraum bestehen (oft >24 Stunden), ohne dass die Ursache von Operateuren behoben werden kann (z. B. defekte Geräte, für die zunächst ein Servicetechniker bestellt werden muss) [Hollifield und Habibi, 2007, S. 136]. Aber auch bei regelmäßigen Wartungsarbeiten und Tests der Anlagenteile entstehen unnötige Alarme, die Operateure nur mit viel Aufwand als solche erkennen können [Vicente et al., 2001, S. 843]. Mumaw et al. stellen bei ihren Untersuchungen fest, dass über 50% aller versendeten Alarme unnötig sind [Mumaw et al., 2000].

- **Alarmfluten (engl. alarm floods):** Dabei treffen in kurzer Zeit mehr Alarme ein, als Operateure verarbeiten können. So empfiehlt beispielsweise die EEMUA eine Alarmobergrenze von 300 Alarmen pro Tag [EEMUA, 2007]. In praktischen Studien wurden dagegen bis zu 6000 Alarme pro Tag und in kritischen Situationen gar bis zu 300 Alarme pro Minute aufgezeichnet [vgl. Hollifield und Habibi 2007, S. 41; Endsley und Connors 2008, S. 149]. Die meisten Alarmfluten entstehen, während Anlagenteile gewartet werden oder komplett ausfallen und dadurch Folgealarme auslösen [EEMUA 2007, S. 158; Hollifield und Habibi 2007, S. 96-97].

Für ein effizientes Alarmmanagement existieren zahlreiche Lösungsansätze, bei denen Themen wie Risikobewertung, Alarmkonfiguration sowie Tests der Alarmsysteme und Alarm-Werkzeuge im Fokus stehen. Von diesen Ansätzen wird im Folgenden nur eine kleine Auswahl vorgestellt, bei der Operateure und ihr Umgang mit Alarmen im Vordergrund stehen.

Im Bereich der Alarmkonfiguration präsentieren die Norm ANSI/ISA-18.2, der NAMUR-Arbeitskreis und die EEMUA Maßnahmen für ein effektives Alarmmanagement, die eine Tendenz zur technologie- und datenzentrierten Sichtweise erkennen lassen [ANSI/ISA-18.2 2009; NAMUR 2003; EEMUA 2007]. Diese Maßnahmen verfolgen im Kern das Ziel, eine Alarmphilosophie zu etablieren und Alarme mit dazugehörigen Schwellwerten möglichst exakt zu konfigurieren. Dadurch sollen Alarme bewusster, nach einem festgelegten Leitfaden konfiguriert und viele Alarme bereits an der Quelle ihrer Entstehung adressiert und reduziert werden. Sobald Alarme auftreten, werden Operateure in entsprechenden Alarm-Werkzeugen akustisch und visuell darauf hingewiesen. Dabei empfehlen die Normen DIN 19235 und NAMUR bestimmte Alarmtöne mit unterschiedlichen Lautstärken und Pulsfrequenzen sowie Farben und Blinkfrequenzen, die Operateure bei der auditiven und visuellen Wahrnehmung der Alarme unterstützen [DIN 19235; NAMUR 2003]. In komplexen Alarmsituationen nutzen Operateure neben den Alarm-Werkzeugen zusätzlich Prozessbilder, um Alarme im größeren Kontext der Anlage nachzuvollziehen. Dazu konfigurieren sie für besonders kritische Anlagenteile Abbildungstabellen, in denen Alarme mit passenden Prozessbildern über sog. *Sprungmarken* statisch verknüpft sind [vlg. NAMUR, 2003, S. 17].

Trotz aller Bemühungen, Alarme durch ein effektives Alarmmanagement mit ausgefeilter Alarmkonfiguration zu reduzieren, sind sich viele Experten darin einig, dass auch das beste Alarmmanagement nicht alle Alarmprobleme gänzlich vermeiden kann [EEMUA, 2007, S. 22]. Die Ursachen hierfür sind vielseitig und liegen vorwiegend in der Prozesskomplexität und der aufwändigen Alarmkonfiguration begründet [vgl. EEMUA 2007, S. 22; Emigholz 1996]. Anwenderorientierte Analysen der Alarmsituationen haben gezeigt, dass nicht die Wahrnehmung eingehender Alarme, sondern vielmehr die schiere Anzahl und Häufigkeit aller auftretenden Alarme und eine dafür unzureichende

Softwareunterstützung der Operateure die weitaus größeren Probleme verursachen [vgl. EEMUA 2007; Nachreiner et al. 2006; Emigholz 1996; Mumaw et al. 2000]. Die in aktuellen Leitständen häufig eingesetzten tabellarischen Alarm-Werkzeuge werden gerade bei Alarmfluten schnell unübersichtlich. Sie wachsen rasant an und unkritische Warnungen verdecken oft schwerwiegende Alarme. Zudem präsentieren Alarmtabellen Alarme relativ unstrukturiert, z. B. basierend auf dem Zeitstempel ihrer Entstehung, statt einer logischen Struktur, die auf die Ursache der Alarme hinweist und dadurch eine schnelle Diagnose fördert [vgl. Nachreiner et al. 2006; Schleburg et al. 2012]. Die Analyse der Alarmsituationen wird zusätzlich erschwert, da die benötigten Prozessbilder zu Alarmen in den statisch konfigurierten Abbildungstabellen oft nicht auffindbar sind. Dies liegt primär daran, dass Anlagen sich kontinuierlich verändern und dadurch die Sprungmarken in den Abbildungstabellen über die Zeit erodieren. Die aufgezeigten Probleme erschweren Operateuren, die kritische Situation schnell zu erfassen, daraus Konsequenzen für den Anlagenbetrieb sowie Handlungsoptionen abzuleiten und erhöht dadurch das Risiko für Anlagenausfälle.

Während bei technologiezentrierten Ansätzen häufig die Integrität der Hardware und der dazugehörigen Konfiguration im Fokus stehen, lassen sich nach der EEMUA Verbesserungen im Bereich der Alarmierung deutlich einfacher und mit weniger Aufwand durch höhere Gebrauchsqualität der Alarm-Werkzeuge erzielen:

> *„Alarm hardware integrity is important, but in practice, risk reduction benefits are generally more easily derived from improving usability than from improving hardware integrity“ [EEMUA, 2007, S. 13]*.

Anwenderorientierte Ansätze akzeptieren die Prozesskomplexität in technischen Anlagen. Sie adressieren die daraus resultierenden Alarmprobleme, wie Alarmfluten und unnötige Alarme, durch verbesserte, auf diese Probleme angepasste Alarm-Werkzeuge. Diese Werkzeuge helfen Operateuren, Alarmsituationen besser zu verstehen und geeignete Maßnahmen zur Alarmbehebung einzuleiten. Hierfür präsentiert die EEMUA Konzepte, die bei Alarmfluten zusammengehörige Alarme vorab zu Alarmgruppen aggregieren und Alarme

den betroffenen Anlagenteilen zuordnen [EEMUA, 2007, S. 114]. Schleburg et al. präsentieren einen Ansatz, bei dem sie die kausalen Abhängigkeiten der Alarme entlang der Prozesstopologie untersuchen, potenziell zusammengehörige Alarme gruppieren und dadurch die Anzahl der Alarme reduzieren [Schleburg et al., 2012]. Die Realisierung dieses Ansatzes erfordert die Konfiguration zusätzlicher Filterregeln und die Modellierung kausaler Beziehungen zwischen Anlagenteilen entlang der Prozesstopologie.

Zur Unterdrückung unwichtiger Warnhinweise sowie Alarme, die von Anlagenteilen in Wartung oder Testbetrieb verschickt werden, eignen sich Konzepte, bei denen Operateure die betroffenen Anlagenteile oder ganze Teilanlagen unmittelbar stumm schalten und damit aus der Alarmierung ausschließen können. Die stummgeschalteten Anlagenteile werden Operateuren explizit visualisiert und ggf. mit einer Erinnerungsfunktion versehen, damit Operateure Anlagenteile nach abgeschlossener Wartung oder Testbetrieb wieder in die Alarmierung aufnehmen können [EEMUA 2007, S. 27; NAMUR 2003, S. 13; Hollifield und Habibi 2007, S. 96]. Dadurch entfällt die manuelle und fehleranfällige Konfiguration von Filterregeln. Zudem vergessen Operateure häufig Filter wieder zu entfernen, wenn die Wartung oder der Testbetrieb abgeschlossen ist. Dies liegt darin begründet, dass die Filter meist im Alarmmanagementsystem abgelegt und für Operateure nur implizit sichtbar sind. Für die Aktualisierung der Abbildungstabellen mit Sprungmarken von Alarmen zu Prozessbildern existieren Ansätze, die diese Verknüpfungen automatisiert herstellen und pflegen. Dazu analysieren Algorithmen Prozessbilder und alle darin enthaltenen Prozessvariablen und verknüpfen diese mit dazu passenden Alarmen [NAMUR, 2003, S. 14]. Diese Algorithmen berücksichtigen Änderungen in der realen Anlage sowie in Prozessbildern und liefern dadurch auch langfristig stets aktuelle Ergebnisse.

3.2.2 Kooperative Alarmbearbeitung

In großen Anlagen überwachen Operateure lediglich einen Teil der Anlage und müssen an den Berührungspunkten ihrer Zuständigkeiten kooperieren und ihre Arbeit koordinieren. In einer Alarmsituation müssen sich Operateure beispielsweise über die Bearbeitung eingehender Alarme abstimmen. Dazu

quittieren sie eingehende Alarme mit Hilfe sog. *Quittiernachrichten* und dokumentieren damit im System, dass sie den Alarm zur Kenntnis genommen haben. Gleichzeitig verpflichten sie sich gegenüber anderen Operateuren, die Alarmursache zu untersuchen und zu beheben [vgl. NAMUR, 2003, S. 15]. Züllighoven et al. bezeichnen diese Kooperationsform als *implizite Kooperation* und verstehen darunter [Züllighoven et al., 1998, S. 876]:

BEGRIFF 3.1: IMPLIZITE KOOPERATION
Bei der impliziten Kooperation wird der konkurrierende Zugriff mehrerer Benutzer auf gemeinsame Ressourcen im Benutzungsmodell ermöglicht und verdeutlicht. Kooperation oder Koordination selbst sind aber nicht vergegenständlicht.

Als zentrales Problem der impliziten Kooperation sehen Züllighoven et al. die fehlende Transparenz bei der konkurrierenden Verwendung gemeinsam genutzter Materialien [Züllighoven et al., 1998, S. 877]. Im Kontext der Alarmierung entsteht die Intransparenz vor allem bei der gemeinsam genutzten Ressource *Alarm*. Für diese fehlt oft ein explizites Modell, in dem alle zu einem Zeitpunkt existierenden Alarme vorgehalten und von Operateuren koordiniert bearbeitet werden können. Dieses Modell entsteht aktuell meist implizit aus der Summe der versendeten und quittierten Alarme. Dies führt in der Praxis zu folgenden Problemen:

- **Abweichende Vorstellung von der gegenwärtigen Alarmsituation:** Starten Operateure eine Leitstandanwendung beispielsweise nach einer Softwareaktualisierung oder einem Systemabsturz neu, ist diese nicht in der Lage, den gegenwärtigen Alarmzustand zu replizieren. Ohne ein explizites Modell der Alarmsituation empfängt eine Leitstandanwendung lediglich die ab ihrem Neustart aufgetretenen Alarme. Alarme, die vor Stunden oder Tagen aufgetreten und noch nicht behoben wurden, bleiben den Operateuren verborgen. Dadurch entstehen große Zeitspannen, in denen Operateure eine unterschiedliche Vorstellung vom gegenwärtigen Alarmzustand haben und ggf. Entscheidungen zur Regelung und Steuerung der Anlage auf Basis unterschiedlicher Vorstellungen treffen.

- **Erschwerte Alarmanalyse:** Eingehende Alarme sind oft Folgealarme von vorangegangenen Alarmsituationen. Fehlt Operateuren der historische Kontext, können sie neue Alarme nur schwer nachvollziehen und behandeln.

- **Aufwändige Pflege eines mentalen Modells:** Operateure kompensieren die unvollständige Abbildung der gegenwärtigen Alarmsituation in der Software, in dem sie ein mentales Modell des Alarmzustandes aufbauen und ihr mentales Modell bei jeder Veränderung in der Anlage aktualisieren. Diese Aufgabe ist sehr anspruchsvoll und fehleranfällig. Zusätzlich müssen Operateure ihr mentales Modell bei der Schichtübergabe ihren Kollegen vermitteln und dadurch deren mentales Modell aktualisieren. Das ist gerade dann schwierig, wenn die Kollegen nach einer längeren Unterbrechung, z. B. urlaubs- oder krankheitsbedingt, in den Anlagenbetrieb zurückkehren [vgl. Vicente et al., 2001, S. 846].

Diese Beispiele motivieren die explizite Modellierung des Alarmzustandes in der Leitstandsoftware. Züllighoven et al. bezeichnen den Zustand von Anwendungssystemen als *Anwendungszustand* und verstehen darunter [vgl. Züllighoven et al., 1998, S. 1018]:

BEGRIFF 3.2: ANWENDUNGSZUSTAND
Der Anwendungszustand eines nach WAM entworfenen, interaktiven Anwendungssystems umfaßt den in Software repräsentierten Zustand der Werkzeuge, Materialien, Automaten und Behälter zur Laufzeit.

Der Anwendungszustand legt die möglichen Umgangsformen des Benutzers mit dem Anwendungssystem fest. Der Zustand des (sozialen) Kontextes des Anwendungssystems, also der Zustand des Benutzers, wird dagegen weder modelliert noch implementiert.

Wenn der Anwendungszustand der Leitstandsoftware mit der realen Alarmsituation in der Anlage nicht übereinstimmt, vermitteln die Alarm-Werkzeuge Operateuren ein falsches Abbild der Realität. Diese Situation ist nicht trivial und hat beispielsweise 2003 in den USA zu einem Stromausfall geführt. Dieser betraf weite Teile des Nordostens sowie Teile Kanadas, mit einem geschätzten

Schaden von 6 Mrd. Dollar [ELCON, 2003]. Eine der Ursachen der Katastrophe war eine fehlerhafte Leitstandsoftware, die Operateuren ein falsches Bild der Alarmsituation vermittelt hat [vgl. PSOTF, 2003, S. 18].

Als Lösungsansatz zur Unterstützung der impliziten Kooperation präsentieren Züllighoven et al. u.a. das Konzept des *Kooperationsmediums*. „Ein Kooperationsmedium ist ein fachlich motivierter Gegenstand, der zur Realisierung von Kooperation in Anwendungssystemen dient“ [Züllighoven et al., 1998, S. 874]. Zur Unterstützung der kooperativen Arbeit der Operateure im Kontext der Alarmierung wird als Kooperationsmedium ein explizites *Alarmmodell* benötigt, das in dieser Arbeit wie folgt verstanden wird:

BEGRIFF 3.3: ALARMMODELL
Das Alarmmodell spiegelt alle in einer technischen Anlage aufgetretenen Alarme sowie den Status ihrer Bearbeitung zur Laufzeit wider. Es hilft Operateuren, eine einheitliche Vorstellung vom gegenwärtigen Alarmzustand zu entwickeln. Das Alarmmodell legt die möglichen Umgangsformen der Operateure mit Werkzeugen und Materialien im Kontext der Alarmierung fest und hilft Operateuren, die Bearbeitung der Alarme zu koordinieren.

Für den Umgang mit den aufgezeigten Alarmproblemen werden explizite Modelle der Anlage und der gegenwärtigen Alarmsituation benötigt. Diese Modelle helfen dabei, zusammengehörige Alarme zu erkennen, zu gruppieren, den betroffenen Anlagenteilen zuzuordnen und ihre kooperative Bearbeitung durch Operateure zu unterstützen.

3.3 Historischer Datenzugriff

Die Menge an Daten, die in Prozessleitsystemen erfasst und gespeichert werden, nimmt rasant zu [Früh et al., 2009, S. 158]. Diese Daten liegen meist sehr feingranular vor und repräsentieren dynamische Veränderungen einzelner Messgrößen im zeitlichen Verlauf. Für die Auswertung historischer Daten existieren datenzentrierte Ansätze, die atomare Messwerte in Kurven- oder Balkendarstellungen visualisieren oder mit komplexen Datenkompressions-

und Analysefunktionen auswerten. Historische Daten können auch mit speziellen Programmen durchleuchtet werden, um z. B. Ineffizienzen in Produktionsabläufen zu erkennen [vgl. Schwarz und Börcsök 2000; Lange et al. 2010, S. 63ff]. Weitere Auswertungskonzepte analysieren sog. Meldearchive, in denen historische Alarme und Eingriffe der Anlagenfahrer in den Prozess protokolliert werden. Mit Hilfe statistischer Analysen können so Häufigkeitsverteilungen ermittelt werden, die besonders problematische Anlagenteile oder solche mit häufigen Bedienereingriffen aufdecken [Früh et al., 2009, S. 158ff].

Eine anwenderorientierte Sichtweise auf die historischen Datenauswertung offenbart, dass Kurvenbilder, Balkendarstellungen oder Datenkompressionsverfahren Operateuren zwar helfen, atomare Daten zu aggregieren und Wechselbeziehungen zwischen Messgrößen verschiedener Anlagenteile aufzuzeigen. Diese Werkzeuge können jedoch nur wenige, höchstens 10 Messgrößen gleichzeitig darstellen [vgl. Vicente et al. 2001, S. 839; Schöner 2010; Pätzold 2011]. Darüber hinaus sinkt die Übersicht rapide und erschwert die Datenanalyse. Große Anlagen enthalten jedoch Tausende Messgrößen. Dadurch fällt es Operateuren schwer, diese Daten in einem größeren Anlagenkontext zu betrachten und die Wechselbeziehungen zwischen Anlagenteilen auf einer höheren Ebene nachzuvollziehen, die von einzelnen Sensoren und ihren Messwerten abstrahiert. Bei der Fallstudie DESY werden historische Messwerte ohne zugehörige Metadaten wie z. B. Alarmgrenzen persistiert, um die Datenmenge in der Datenbank zu reduzieren. Dies erschwert Operateuren, Kurvenverläufe semantisch zu interpretieren und z. B. Ausreißer in einem Kurvenverlauf mit einer Alarmsituation in Verbindung zu bringen. Zur Lösung der aufgeführten Probleme existieren Ansätze, die historische Daten, neben den Kurvenbildern, zusätzlich in Prozessbildern visualisieren [Reiswich und Fay 2012; Reiswich 2012a]. Dies ermöglicht, die historischen Daten in den Kontext der Anlage einzuordnen und erleichtert Operateuren die semantische Interpretation der Daten. Für diesen Ansatz benötigen Prozessbilder historische Messwerte und dazugehörige Metadaten wie z. B. Alarmgrenzen, um deren Überschreitungen zu visualisieren. Diese Informationen können Anlagenmodelle mit inkludierten Metadaten liefern und sie bei Bedarf mit

zugehörigen historischen Messwerten zu semantisch reichhaltigen Zeitreihen zusammenführen [Reiswich und Fay 2012; Reiswich 2012a].

3.4 Zusammenfassung

Technologiezentrierte Ansätzen nähern sich Lösungen für relevante Leitstandprobleme vorwiegend durch die Fragen: Was kann die Technik leisten und wie kann ihr Potential erschöpfend genutzt werden? Die daraus resultierenden Lösungen sind vorrangig auf reine Datenverarbeitung ausgerichtet und nutzen dazu ausgefeilte Konfigurationen und verbesserte Automaten. Die Prozesskomplexität in großen Industrieanlagen führt jedoch dazu, dass Konfigurationen und Automaten gerade in kritischen Situationen oft versagen oder nicht den Kern der Leitstandprobleme adressieren. Dadurch wird ein Großteil der ungelösten Probleme auf Operateure verlagert. Diese sind oft mit Softwaresystemen ausgestattet, die Operateure bei der Lösung der aufgezeigten Probleme nur unzureichend unterstützen.

Bei einer anwenderorientierten Sichtweise rückt der Arbeitsplatz der Operateure in den Mittelpunkt des Prozessgeschehens. Anwenderorientierte Ansätze akzeptieren und adressieren die Komplexität in technischen Anlagen und nähern sich Lösungen für Leitstandprobleme primär durch die Fragen: Wie kann Operateuren der Umgang mit der Prozesskomplexität erleichtert werden und welche technischen Hilfsmittel eignen sich dazu? Anwenderorientierte Ansätze nutzen die kognitiven Fähigkeiten und das Expertenwissen der Operateure, komplexe und unklare Informationen zu verstehen und in Aktionsentscheidungen zu integrieren. Sie untersuchen Leitstandprobleme aus Sicht der Operateure, helfen ihre Ursache zu verstehen und bieten Operateuren dazu passende Lösungen, die zudem mit vertretbarem Aufwand realisiert werden können. In diesem Abschnitt wurden zentrale Aspekte der ersten Forschungsfrage beantwortet und der Beitrag der Anwendungsorientierung in der Automatisierungstechnik herausgearbeitet.

Die in diesem Kapitel vorgestellten anwenderorientierten Lösungsansätze können durch moderne, besser auf die Aufgaben der Operateure abgestimmten Softwarewerkzeuge unterstützt werden. Die Realisierung dieser Lösungsan-

sätze erfordert ein softwaretechnisches Modell der realen Anlage. In vielen aktuellen Leitständen, wie auch bei der Fallstudie DESY, fehlen explizite Anlagenmodelle, oder sie weisen nur unzureichende Eigenschaften auf [vgl. z. B. Mahnke et al. 2009, S. 19; Lange et al. 2010, S. 100; Schlick et al. 2013]. Dadurch können die vorgestellten anwenderorientierten Ansätze gar nicht, oder nur unter erheblich hohem Aufwand und Risiko auf bestehende Leitstandsystemen übertragen werden. In der Automatisierungstechnik gab es in der Vergangenheit bereits Bemühungen, Anlagenmodelle in Leitstandsystemen zu etablieren. Dabei wurde oft das Ziel verfolgt, ein generisches und domänenübergreifendes Gesamtmodell zu erschaffen. Dieses Modell sollte einen möglichst großen, mehrere Industriezweige umfassenden Anwenderkreis adressieren und dabei helfen, homogene und interoperable Systeme zu entwerfen. Dieses Gesamtmodell war jedoch sehr umfangreich und technisch und praktisch nicht umsetzbar [vlg. Sauter 2005; Sauter 2007; Epple 2011]. Die ursprüngliche Idee zersplitterte daher in zahlreiche domänenspezifische Modellierungsstandards [Mahnke et al., 2011]. Verunsichert durch diese Diversität bevorzugen Unternehmen heutzutage nicht selten Eigenentwicklungen, die wiederum die Verbreitung der vorliegenden Standards erschweren [Siltanen und Pärnänen, 2006]. Im folgenden Kapitel werden allgemeine Anforderungen an Anlagenmodelle formuliert, aktuelle Standards vorgestellt und diese schließlich mit den Anforderungen verglichen, um für diese Arbeit einen geeigneten Modell-Kandidaten zu finden.

Kapitel 4

Anlagenmodellierung in der Prozessautomatisierung

Anlagenmodelle entstehen in verschiedenen Ausprägungen während des gesamten Lebenszyklus technischer Anlagen sowie entlang aller Ebenen der Automatisierungspyramide. In den frühen Planungsphasen entstehen Anlagenmodelle vorwiegend in Computer Aided Design (CAD)-Werkzeugen. Diese enthalten zwei- und dreidimensionale Darstellungen der Anlage und unterstützen Ingenieure vornehmlich bei der Zeichenarbeit, bei Berechnungen und der technischen Projektüberwachung [Früh et al., 2009, S. 632]. In der anschließenden Errichtungsphase entwickeln Ingenieure mathematische Modelle zur Regelung von Prozessen, zur Fehlerdiagnose und Simulation. In dieser Phase designen Projekteure auch genormte grafische Modelle der Anlage in Form von Prozessbildern [vgl. Polke, 1994, S. 39]. In der späteren Betriebsphase entwickeln Ingenieure Anlagenmodelle, die in einer Leitstandsoftware ablaufen und Operateuren Anlagenteile und ihre Beziehungen strukturiert präsentieren [vgl. z. B. Liu et al. 2005; García und Gelle 2004]. Darüber hinaus existieren in der Literatur Vorschläge, Modelle einzusetzen, die mehrere Lebenszyklen der Anlage umspannen und beispielsweise zur besseren IT-Integration [vgl. z. B. ISO 15926] oder zu einfacherem Datenaustausch in Unternehmen beitragen [vgl. z. B. Drath et al., 2008]. Diese Auswahl stellt nur einen kleinen Ausschnitt der Modellwelt in der Prozessautomatisierung dar. Eine detaillierte Übersicht bieten z. B. Nagel und Westfechtel [2003,

S. 25], Freund [2010, S. 66ff] sowie Mahnke et al. [2011]. All diese Modelle werden von diversen Leitstandanwendungen für verschiedene Zwecke und zu unterschiedlichen Zeitpunkten genutzt [vgl. Fröhlich et al., 2002a, S. 169]. Für diese Arbeit sind Anlagenmodelle relevant, die während der Betriebsphase auf der Leitebene in einer Leitstandsoftware ablaufen, Operateuren komplexe technische Zusammenhänge strukturiert präsentieren und sie dadurch bei ihren Leitstandaufgaben unterstützen. In den folgenden Abschnitten werden aufbauend auf den anwenderorientierten Lösungsansätzen des vorangegangenen Kapitels Anforderungen an Anlagenmodelle formuliert und mit standardisierten Anlagenmodellen verglichen, um für diese Arbeit einen geeigneten Modell-Kandidaten zu finden.

4.1 Anforderungen an Anlagenmodelle auf der Leitebene

Große Industrieanlagen bestehen aus tausenden Anlagenteilen wie Pumpen, Ventilen oder Motoren. Anlagenbetreiber führen im Lebenszyklus einer Anlage diverse Projekte durch, die einen Bezug zu diesen Anlagenteilen haben. Diese Projekte beschäftigen sich beispielsweise mit der Rekonfiguration, Wartung oder Erweiterung einer Anlage in der Betrieb&Wartungs-Phase, die im Fokus dieser Arbeit ist (siehe Abschnitt 2.1). Bei dieser Vielzahl an Projekten und Anlagenteilen benötigen Anlagenbetreiber systematische Strukturierungskonzepte, die alle Beteiligten dabei unterstützen, komplexe Zusammenhänge in technischen Anlagen zu erkennen und die Anlagenkomplexität zu beherrschen [vgl. García und Gelle 2006; Hollender 2009, S. 56]. In der Automatisierungstechnik werden für diese Strukturierungskonzepte Anlagenmodelle verwendet. Diese übernehmen eine Brückenfunktion zwischen der materiellen und der immateriellen softwaretechnischen Welt und helfen, diese Welten zu synchronisieren [Hesse und Mayr 2008; Ludewig 2003; Fröhlich et al. 2002a, S. 166].

Technische Anlagen weisen eine Reihe identischer Strukturmerkmale und Eigenschaften auf, aus denen sich grundlegende Anforderungen an Anlagen-

modelle ableiten lassen [vgl. Hollender 2009, S. 56ff; Epple 2003]. Siltanen und Pärnänen unterteilen diese Anforderungen in drei Kategorien [Siltanen und Pärnänen, 2006, S. 348]:

- **Business Anforderungen:** Das Anlagenmodell muss den Austausch technischer Informationen unterstützen und vorzugsweise weltweit in der Industrie akzeptiert sein. Die formalisierte Beschreibung des Anlagenmodells (Spezifikation) sollte leicht verständlich und schnell umsetzbar sein.

- **Technische Anforderungen:** Die Spezifikation des Anlagenmodells sollte technologieunabhängig sein, um in diversen Unternehmen eingesetzt zu werden. Der Datenaustausch zwischen Anlagenmodellen sollte durch einen offenen Standard unterstützt werden.

- **Usability Anforderungen:** Das Anlagenmodell sollte verständlich und mit den gegebenen Technologien leicht umzusetzen sein. Es sollte flexibel sein, um sich schnell technologischen und wirtschaftlichen Veränderungen anzupassen. Das Anlagenmodell sollte auf realen Objekten im Arbeitsumfeld basieren. Züllighoven et al. bezeichnen dies als *Strukturähnlichkeit* und verstehen darunter [Züllighoven et al., 1998, S. 263]:

  BEGRIFF 4.1: STRUKTURÄHNLICHKEIT
  Strukturähnlichkeit bezieht sich im WAM-Ansatz auf das Verhältnis von Software und Anwendungsbereich:

  Die softwaretechnischen Komponenten eines Softwaresystems modellieren die relevanten Konzepte und Gegenstände des Anwendungsbereichs.

  Die Architektur des Anwendungssystems spiegelt die wesentlichen Bezüge zwischen den Konzepten und Gegenständen des Anwendungsbereichs wider.

Die Strukturähnlichkeit zwischen den realen Anlagenteilen und den Modellelementen im Anlagenmodell hat für diese Arbeit eine besonders hohe Relevanz. Sie bietet, aus einer anwenderorientierten Sicht betrachtet, folgende Vorteile:

- Operateure finden reale Anlagenteile im Modell repräsentiert und können dadurch komplexe technische Zusammenhänge, wie z. B. den Aufbau einer Anlage, strukturiert betrachten und schrittweise explorieren.
- Operateure können Anlagenteile im Modell suchen und feststellen, um welche Art von Anlagenteil es sich dabei handelt. Dies ist hilfreich, da Anlagenteile in jedem Unternehmen individuell, entsprechend einem Kennzeichnungssystem benannt werden. Dies erschwert gerade neuen Operateuren die Einarbeitung, aber auch erfahrenen Operateuren das Verständnis über die Anlage, da große Anlagen oft aus tausenden Anlagenteilen bestehen [Nachreiner et al., 2006, S. 15].
- Ein strukturähnliches Anlagenmodell bietet Leitstand-Softwareentwicklern die Möglichkeit, fachliche Konzepte und Modellelemente bei fachlichen oder softwaretechnischen Änderungen zueinander in Beziehung zu setzen und somit die wechselseitigen Abhängigkeiten zu erkennen [Züllighoven et al., 1998, S. 264]. Dadurch können Softwareentwickler z. B. Änderungen in der realen Anlage schneller in der Leitstandsoftware lokalisieren und die Software entsprechend anpassen.

Neben diesen grundlegenden Anforderungen an Anlagenmodelle lassen sich, aus den anwenderorientierten Lösungsansätzen des vorangegangenen Kapitels 3, weitere konkrete Anforderungen an die Eigenschaften der Anlagenmodelle ableiten:

- **Anlagenstruktur:** Technische Systeme werden typischerweise in Subsysteme und Komponenten organisiert, um Operateuren die Orientierung und Navigation in komplexen Anlagen zu vereinfachen [vgl. García und Gelle 2004; Mahnke et al. 2011; Drath et al. 2008]. Dazu bilden Ingenieure die Anlage in einer hierarchischen Struktur ab. Je nach Anforderung organisieren Ingenieure Anlagenteile z. B. entsprechend ihrer räumlichen Beziehung (gleicher Einbauort) oder ihrer Beteiligung an gemeinsamen Produktionsprozessen [Fröhlich et al. 2002b; Bratthall et al. 2002]. Dabei werden üblicherweise nur die relevanten Merkmale bis zu einem gewissen Detaillierungsgrad abgebildet wie z. B. Roboter und Greifer, jedoch nicht deren Achsen oder Gelenke [vgl. Drath et al., 2008].

Die Anlagenstruktur kann beispielsweise herangezogen werden, um bei Alarmfluten (entsprechend dem EEMUA-Ansatz) Alarme zu aggregieren und sie dem Ort ihrer Entstehung zuzuordnen.

- **Prozessvariablen:** Prozessvariablen repräsentieren Messwerte sowie Stellgrößen, über die Anlagenfahrer oder automatische Regelungsfunktionen den Prozess beeinflussen können [vgl. Früh et al., 2009, S. 56f]. Sie werden nach einem unternehmensspezifischen oder standardisierten Kennzeichnungssystem benannt und bei der Erstellung und Pflege der Prozessbilder verwendet, um grafische Symbole mit realen Anlagenteilen zu verbinden.

- **Metadaten zu Prozessvariablen:** Damit Operateure die übertragenen Messwerte besser interpretieren können, benötigen sie neben dem eigentlichen Wert zusätzliche Informationen wie die Messeinheit (z. B. °C, kg, oder U/Min) sowie den Betriebsbereich mit zulässigen Grenzwerten (z. B. 100-200 C°, 800-6000 U/Min) [vgl. Epple 2003; Mahnke et al. 2009, S. 19]. Diese und weitere Metadaten werden Prozessvariablen zugeordnet, um ihren Informationsgehalt semantisch anzureichern.

- **Typinformationen:** Typinformationen für Anlagenteile und Prozessvariablen ermöglichen Operateuren u.a. Rückschlüsse auf das Geräteverhalten wie z. B. Hitze- und Druckbeständigkeit oder Leistung [vgl. Epple 2003; Mahnke et al. 2009, S. 19]. Ein Temperaturfühler vom Typ PT-100 gibt z. B. Aufschluss über die maximale Temperaturbeständigkeit zwischen -200°C und +850°C. Typinformationen unterstützen zusätzlich die Umsetzung halbautomatischer Projektierungsverfahren (wie beim Objekt-Typ-Ansatz) und erleichtern somit Projekteuren die Erstellung und Pflege der Prozessbilder.

Die Auswahl dieser Eigenschaften orientiert sich an den in Kapitel 3 betrachteten Leitstandproblemen sowie dazugehöriger Lösungsansätze. Obwohl die vorgestellten Probleme in vielen Unternehmen wiederkehrend auftreten, unterscheiden sich ihre konkreten Ausprägungen, wodurch auch die Anforderungen an die Lösungsansätze sowie die Anlagenmodelle variieren. Um nicht für jedes Unternehmen Anlagenmodelle grundlegend neu zu konstruieren,

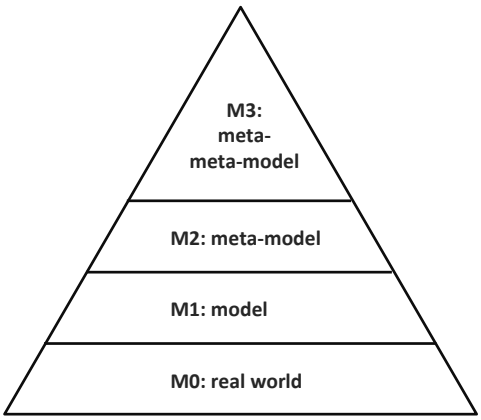
werden Modellierungskonzepte benötigt, die lediglich eine Infrastruktur zur Modellierung bieten, mit der Unternehmen ihre individuellen Anlagenmodelle ausgestalten können [vgl. Mahnke et al., 2009, S. 19]. Diese Infrastruktur bieten Metamodelle. Sie formulieren allgemeine Gesetzmäßigkeiten als sprachliche Regeln und stellen diese in einen definierten formalen Rahmen [vgl. Epple 2003; Diedrich und Mühlhause 2011].

4.2 Metamodellierung

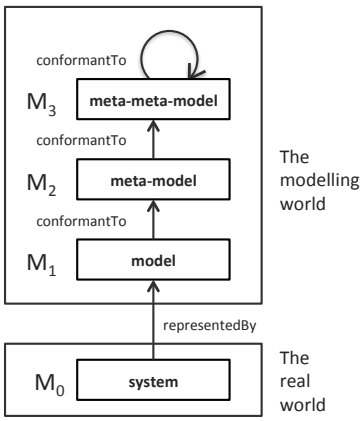
Der Begriff *Metamodell* wird in der Literatur unterschiedlich verwendet und hat zu wissenschaftlichen Auseinandersetzungen bei der Unterscheidung zwischen den Metastufen und ihren Beziehungen während der Modellbildung geführt [Hesse und Mayr, 2008]. Die Definition der Object Management Group (OMG): „Metamodel: a model of models“ soll verdeutlichen, dass Metamodelle als Basis für eine Menge von Modellen dienen [OMG, 2003, S. A-2]. Hesse und Mayr bezeichnen diese knappe Definition als unpräzise und konkretisieren den Metamodell-Begriff wie folgt [Hesse und Mayr, 2008]:

BEGRIFF 4.1: METAMODELL
„Ein Modell C wird genau dann ein Metamodell genannt, wenn sein Original selbst ein Modell – hier B genannt – mit einem zugehörigen Original A ist und wenn beide Modellierungsbeziehungen von der Art Typmodellierung sind, d.h. auf Stellvertreterprojektion beruhen“ [Hesse und Mayr, 2008, S. 389].

Bei dieser Definition unterscheiden die Autoren zwischen Original, Modell und dem Metamodell. Zur Illustration dieser Konzepte und ihren Beziehungen wird in der Literatur üblicherweise die OMG-Metapyramide herangezogen. Bézivin und Favre wandeln diese Pyramide leicht ab. Sie trennen die reale Welt von der Modellierungswelt, um sie besser unterscheiden zu können (Abbildung 4.1) [Bézivin 2004; Favre 2005]. Die Ebene M_0 der Metapyramide ist dabei die reale Welt, die durch ein Modell auf der Ebene M_1 repräsentiert wird. Dieses Modell bildet nur eine Auswahl der Eigenschaften seines Originals ab, die der Modellierer nach pragmatischen, für seinen Zweck relevanten



(a) OMG Pyramide



(b) The 3+1 MDA organisation revisited (Quelle: Bézivin, 2004)

Abbildung 4.1: Meta-Modellierung Ebenenmodelle

Aspekten auswählt [Stachowiak, 1973]. Dies geschieht mit Hilfe einer Modellierungssprache, die u.a. Entitäten, ihre Beziehungen und Attribute definiert [Karagiannis und Höfferer, 2006]. Die Syntax einer Modellierungssprache regelt dabei, wie diese Elemente miteinander verknüpft werden. Die Semantik fügt diesen Verknüpfungen eine Bedeutung hinzu [vgl. z. B. Harel und Rumpe 2004; Karagiannis und Kuhn 2002]. Ein Metamodell auf der Ebene M_2 bietet Konzepte zur Konstruktion von Modellen und wird mit Hilfe einer Meta-Modellierungssprache entwickelt. Ein Meta-Metamodell der Ebene M_3 wird wiederum mit einer Meta-Metamodellierungssprache entwickelt. Diese bietet Konzepte zur Konstruktion von Metamodellen.

In welcher Beziehung Modell, Metamodell und Meta-Metamodell zueinander stehen, ist in der Wissenschaft umstritten. Die aus der Objektorientierung oft übernommenen und auf die Metamodellierung übertragenen Typ-Exemplar-Beziehungen, wie sie in der vorgenannten Metamodell-Definition von Hesse und Mayr zu finden ist, bezeichnen Bézivin und Seidewitz als irreführend, da ein Modell kein Exemplar eines Metamodells ist [Bézivin 2004; Seidewitz 2003]. Bézivin definiert daher die Beziehung zwischen Original und Modell als *representedBy* und die Beziehungen zwischen den Modellen der Ebenen M_1 - M_3 als *conformantTo*. Diese Beziehungen grenzt Bézivin von den objektorientierten Beziehungen ab, die er jedoch nicht als gegensätzliche, sondern als komplementäre Ansätze betrachtet [Bézivin, 2004]. Gleichzeitig weist er

darauf hin, dass die Bedeutung dieser Beziehungen wohl ähnlich kontrovers und langwierig diskutiert werden wird, wie einst die grundlegenden Konzepte und Beziehungen in der Objektorientierung. In dieser Arbeit werden für die Modell-Beziehungen die Begriffe nach Bézivin verwendet.

4.3 Anlagenmetamodelle

Metamodelle existieren in verschiedenen Varianten, von konzeptionellen Vorschlägen bis zu implementierten und in Software ablauffähigen Produkten [vgl. Karagiannis und Kuhn, 2002, S. 6]. Für diese Arbeit werden in einer Leitstandsoftware ablauffähige Anlagenmodelle benötigt, die mit einer Modellierungssprache des zugrundeliegenden Metamodells entwickelt werden. Anlagenmodelle repräsentieren die reale technische Anlage und können daher der Ebene M_1 zugeordnet werden. Sie sind konform zu ihrem Metamodell auf der Ebene M_2. Zur Verdeutlichung werden Metamodelle für technische Anlagen im Folgenden als *Anlagenmetamodelle* bezeichnet. Sie ermöglichen eine strukturierte Modellierung der realen Anlage unter Berücksichtigung aller relevanter Anlagenmerkmale. Damit Anlagenmetamodelle nicht nur die aktuellen, sondern auch zukünftig relevanten Anlagenmerkmale unterstützen, müssen sie flexibel sein und Veränderungen in ihrem Anwendungsbereich adaptieren können [vgl. Karagiannis und Kuhn, 2002].

Ein generisches, in der Prozessautomatisierung akzeptiertes Anlagenmetamodell darf nach Epple nicht zu speziell formuliert sein, um Erweiterungsmöglichkeiten offen zu halten [Epple, 2003]. Je allgemeiner und unschärfer jedoch ein Anlagenmetamodell formuliert wird, desto mehr verliert es an semantischer Ausdrucksstärke und an praktischem Nutzwert [vgl. z. B. Epple 2003; García und Gelle 2004]. In der Prozessautomatisierung gab es in der Vergangenheit mehrere Bemühungen, ein generisches Anlagenmetamodell zu entwickeln. Diese Bemühungen haben jedoch bis dato zu einer heterogenen Modellwelt mit diversen Standards geführt [Mahnke et al., 2011]. Statt ein weiteres, eigenentwickeltes Anlagenmetamodell einzuführen, werden in dieser Arbeit zunächst bestehende und für die Leitebene geeignete Standards vorgestellt. Anschließend werden die Modelleigenschaften der Standards an-

hand der in Abschnitt 4.1 formulierten Anforderungen überprüft, um einen für diese Arbeit geeigneten Meta-Modell-Kandidaten zu wählen.

4.4 Standardisierte Anlagenmetamodelle

In der Prozessautomatisierung haben sich in der Vergangenheit, entlang des Lebenszyklus technischer Anlagen sowie auf allen Ebenen der Automatisierungspyramide, getrennte Informationswelten mit unterschiedlichen Denkweisen und Disziplinen entwickelt. Heute benötigen jedoch Unternehmen einen durchgängigen Informationsfluss im gesamten Betrieb [vgl. Sauter 2005; Thiel et al. 2010, S. 35]. IT-Systeme wie z. B. Anlagenplanungssysteme, Bestellsysteme, Prozessleitsysteme sowie Produktionsplanungs- und Instandhaltungssysteme müssen Informationen untereinander austauschen können, um eine ganzheitliche Produktion zu unterstützen [vgl. Epple, 2011]. Daraus ist Mitte der 70ger Jahre die Idee des *Computer Integrated Manufacturing* (CIM) entstanden. CIM verfolgte das Ziel, in Unternehmen bereits vorhandene softwaretechnische Insellösungen in ein Gesamtmodell zu integrieren. Dieses Gesamtmodell war jedoch sehr umfangreich und technisch und praktisch nicht umsetzbar [vgl. Sauter vgl. 2005, S. 258; Sauter 2007, S. 11; Epple 2011]. Aus der ursprünglichen Idee sind schließlich diverse domänen- und unternehmensspezifische Standards hervorgegangen. Diese verfolgen das Ziel, Anlageninformationen mit Hilfe von Modellen standardisiert zu präsentieren, zu integrieren und zwischen Unternehmen auszutauschen [vgl. Sauter 2005; Mahnke et al. 2011]. Eine Auswahl dieser Standards, mit Blick auf die Leitebene, stellen die folgenden Abschnitte vor.

4.4.1 IEC 61346

Der internationale Standard IEC 61346 wurde ursprünglich von dem Unternehmen ABB[1] entwickelt und u.a. in der Leitstandsoftware ABB 800xA realisiert. Der Standard hat zum Ziel, mit Hilfe eines Anlagenmetamodells Strukturierungskonzepte für technische Systeme und damit verknüpfte In-

[1] Technologiekonzern für Energie- und Automatisierungstechnik - www.abb.com

formationen zu etablieren. Die Strukturierungskonzepte basieren auf der Idee, dass eine technische Anlage in verschiedenen Arbeitssituationen aus unterschiedlichen Sichten betrachtet werden kann. In jeder Sicht können dabei technische Zusammenhänge in einer hierarchischen Struktur präsentiert werden. Hierfür definiert der Standard drei grundlegende Konzepte: *Objekte*, *Aspekte* und *Strukturen* [vgl. García et al. 2004; Fröhlich et al. 2002b]:

- **Objekte** repräsentieren reale Anlagenteile, aber auch virtuelle „Dinge“, die für den zu steuernden und überwachenden Prozess relevant sind.
- **Aspekte:** Zwischen Anlagenteilen existieren diverse Beziehungen. Sie können räumlich nebeneinander angeordnet oder an dem gleichen Produktionsprozess beteiligt sein. Aspekte ermöglichen, Anlagenteile aus unterschiedlichen Sichtweisen zu betrachten. Der Standard definiert daher drei fundamentale Aspekte, die individuell erweitert werden können: ortsbasierte, funktionale und produktorientierte Aspekte. Der *ortsbasierte* Aspekt beantwortet die Frage, wo sich ein Anlagenteil räumlich befindet. Der *funktionale* Aspekt beantwortet die Frage, welche Aufgabe ein Anlagenteil im Prozess übernimmt. Der *produktorientierte* Aspekt beantwortet die Frage, wie ein Anlagenteil mechanisch zusammengesetzt ist.
- **Strukturen** organisieren Objekte entsprechend ihren Aspekten in einer Hierarchie.

Der Standard IEC 61346 gilt als internationaler Nachfolger des Kraftwerkskennzeichnungssystems (KKS), zielt jedoch nicht nur auf Kraftwerke, sondern auf die gesamte Industrie ab [vgl. Fröhlich et al., 2002b]. IEC 61346 wird einerseits zur Kennzeichnung (Namensgebung) von Anlagen und Betriebsmitteln verwendet [vgl. z. B. Brand et al. 2004; García und Gelle 2004; Johansson et al. 2006]. Andererseits existieren auch auf dem Standard aufbauende und in der Leitstandsoftware ablauffähige Anlagenmodelle [vgl. z. B. Bratthall et al. 2002; Fröhlich et al. 2002b; Hu 2003]. García und Gelle kritisieren an diesem Standard, dass die Konzepte *Aspekt* und *Objekt* nicht klar getrennt sind und sich bei der Umsetzung vermischen [García und Gelle, 2004]. In einer hierarchischen Struktur können die einzelnen Knoten und Blätter einerseits

Aspekte, andererseits auch Objekte sein. Objekte können wiederum physische Anlagenteile, virtuelle „Dinge“ oder auch Informationen zu Anlagenteilen sein [García und Gelle, 2006].

4.4.2 IEC 61850

In den vergangenen Jahren haben technologische Entwicklungen auch den Energiesektor, mit seinen Kraft- und Umspannwerken, nachhaltig verändert. Energieversorgungsunternehmen stehen aktuell vor der Herausforderung, ihre bewährte, aber veraltete Technologie des 20sten Jahrhunderts den neuen Anforderungen anpassen zu müssen [Higgins et al., 2011]. Während zu Beginn der 30er Jahre Anweisungen und Informationen telefonisch übertragen wurden, werden heute tausende analoge und digitale Daten in sog. *Intelligent Electronic Devices (IED)* bereitgestellt [Mackiewicz, 2006]. Diese werden u.a. in Umspannwerken zum Schutz der Anlagenteile wie Transformatoren, Schaltanlagen oder Starkstromleitungen eingesetzt. Im Betrieb müssen Informationen, zwischen und aus den IEDs, zu lokalen Bedienstationen und schließlich zum Leitstand übertragen werden. Diese Informationsübertragung hat die International Electrotechnical Commission (IEC) zwischen 2003 und 2005 standardisiert und dazu den Standard IEC 61850 „Communication Networks and Systems in Substations“ veröffentlicht.

Dieser Standard spezifiziert ein Anlagenmetamodell, mit dem Energieversorgungsunternehmen konkrete Anlagenmodelle erstellen können. Diese sind hierarchisch organisiert und bestehen aus physischen Geräten, denen logische Geräte und Knoten zugeordnet sind. Ingenieure können auf diese Weise die vielfältigen atomaren Anlageninformationen zu Anlagenteilen zuordnen und strukturiert organisieren [Mackiewicz, 2006]. Obwohl ursprünglich für die Umspannwerke entwickelt, wurde dieser Standard bald auf weitere Bereiche der Versorgungsindustrie wie z. B. Wasser- und Windkraftwerke übertragen. Nach Mahnke et al. und Lehnhoff et al. bietet der Standard relativ einfache und für die Domäne geeignete Strukturierungskonzepte. Diese können jedoch nicht ohne weiteres auf andere Domänen übertragen werden [Mahnke et al. 2011; Lehnhoff et al. 2011].

4.4.3 IEC 62424

In Abschnitt 2.1 wurde der Lebenszyklus technischer Anlagen in mehrere Phasen aufgeteilt. In jeder Phase setzen Ingenieure für die anfallenden Aufgaben spezialisierte und über Jahre erprobte Software-Werkzeuge ein. Diese Werkzeuge kapseln Daten üblicherweise in herstellerspezifischen Datenformaten, die einen Austausch der Daten zwischen Werkzeugen verschiedener Hersteller erschweren [Güttel und Fay 2008; Epple 2011; Drath et al. 2011]. Dadurch müssen in jeder Phase der Anlage Planungsdaten aufwändig reproduziert werden. Dabei wird nur ein Bruchteil der verfügbaren Daten wiederverwendet, um den Aufwand zu beschränken. Viele Anlageninformationen fehlen somit für spätere Simulations- oder Testprogramme [vgl. Drath, 2010, S. 12]. In der Vergangenheit gab es Bemühungen, den Datenaustausch zwischen Werkzeugen durch standardisierte Schnittstellen zu vereinfachen. Die Umsetzung dieser Bemühungen hat sich jedoch als schwierig erwiesen, da viele Hersteller die Austauschbarkeit ihrer Werkzeuge und Verluste von Marktanteilen befürchten und in Folge die Nutzung ihrer Schnittstellen nur unter Auflagen anbieten [Drath et al., 2011].

Als Lösungsansatz für den Austausch von Planungsdaten zwischen heterogenen Werkzeugen wurde das Datenaustauschformat *Computer Aided Engineering eXchange (CAEX)* entwickelt und 2008 in der internationalen Norm IEC 62424 standardisiert. Die Datenmodellierung erfolgt in CAEX auf Basis eines XML-Schemas, das den Aufbau der CAEX-Elemente wie Klassen, Schnittstellen, Attributen und Instanzen definiert. Diese Konzepte unterstützen eine objektorientierte Modellierung der Anlagentopologie mit konkreten Anlagen, Teilanlagen, ihren Attributen und Beziehungen. CAEX erlaubt zudem die Wiederverwendung bewährter Vorlagen bei der Anlagenplanung durch ein Bibliothekskonzept. Darin lassen sich häufig wiederkehrende Anlagenteile wie Ventile, Motoren oder Pumpen zu entsprechenden Typen mit dazugehörigen Merkmalen abstrahieren und als Vorlage für konkrete Exemplare der Anlagenteile nutzen [Drath, 2010]. CAEX bietet ein werkzeug- und herstellerunabhängiges Metamodell zur Strukturierung von Engineering-Daten und erlaubt gleichzeitig, dieses mit herstellerspezifischen Information zu erweitern [Runde und Fay, 2008].

Der standardisierte Datenaustausch in der Anlagenplanung befindet sich noch in einer frühen Phase [Runde und Fay, 2008]. Aktuelle Herausforderungen entstehen bei der Wahrung der Datenkonsistenz während des wiederholten Datenaustausches zwischen Werkzeugen. Hierfür wird ein ausgereiftes Änderungs- und Versionsmanagement über Werkzeuggrenzen hinweg benötigt [Drath et al., 2011]. Nach Hohpe und Woolf sowie Drath et al. bietet ein standardisiertes Austauschformat lediglich eine technische Infrastruktur für den Datenaustausch. Die ausgetauschten Daten müssen in anderen Systemen auch semantisch interpretiert und verstanden werden [vgl. Hohpe und Woolf 2004, S. 3; Drath et al. 2011]. CAEX ist schließlich ein neuer Standard und muss sich in der Praxis noch etablieren.

4.4.4 OPC Unified Architecture

Die technologischen Entwicklungen in der Prozessindustrie wurden Anfang der 90er Jahre maßgeblich durch die Verbreitung der Windows-Betriebssysteme und der PCs beeinflusst. Die Windows-Technologie wurde ständig weiterentwickelt. Dadurch wuchs bei den Unternehmen der Druck, Softwaretreiber für verschiedene Windows-Versionen zu entwickeln und warten zu müssen. Im Jahre 1995 entschieden sich daher mehrere namhafte Unternehmen, basierend auf Microsofts Windows-Technologie, einen Standard für den Zugriff auf Echtzeitdaten zu erarbeiten. Die daraus entstandene OPC[2]-Foundation veröffentlichte bereits ein Jahr später im August 1996 die OPC-Spezifikation. In den Jahren darauf folgten mehrere Erweiterungen wie die *Data Access Specification* zum Lesen und Schreiben von Prozessdaten, die *Alarms and Events Specification* zur Verarbeitung von Ereignissen und Alarmen sowie die *Historical Data Access Specification* zur Analyse historischer Daten (siehe Abschnitt 2.3).

„Während die einfache Inbetriebnahme und das unkomplizierte Einrichten einer Kommunikationsbeziehung zwischen OPC-Komponenten auf einem lokalen Rechner schnell zur Popularität der OPC-Technologie und ihrer rasanten Verbreitung beitrug, stellte sich die Realisierung einer OPC-Kommunikation

[2]OPC = OLE for Process Control (OPC). Object Linking and Embedding (OLE) ist eine Microsoft-Technologie zur Konstruktion objektorientierter Systeme [Lange et al., 2010, S. 2].

über entfernte Rechner hinweg vielfach als kompliziert und unzureichend dar“ [Lange et al., 2010, S.]. Grund dafür sind die Eigenschaften und Sicherheitseinstellungen der zugrunde liegenden Microsoft-Technologie. Diese Einschränkungen und der zunehmende Wunsch, Automatisierungslösungen auch auf anderen Systemen wie UNIX- oder Linux einzusetzen und diese mit modernen Kommunikationstechnologien wie Webservices zu verbinden, führte schließlich zur Erweiterung des OPC-Standards. Der neue OPC Unified Architecture (OPC UA) Standard soll heutige technische und wirtschaftliche Anforderungen besser unterstützen. Zur besseren Unterscheidung werden die früheren OPC-Konzepte als *Classic OPC* bezeichnet. Die grundlegenden OPC-UA-Konzepte sind Datentransport und Informationsmodellierung [Mahnke et al., 2009, S. 19]:

- **Datentransport:** OPC UA löst die Windows-Abhängigkeit auf und ermöglicht dadurch eine plattform- und protokollunabhängige sowie sichere und verlustfreie Kommunikation, auch über Firewall-Grenzen hinweg. OPC UA unterstützt zusätzlich Webservices und ermöglicht dadurch den Datenaustausch über das Internet [Lange et al., 2010, S. 96ff].

- **Informationsmodelle:** In OPC beschränkt sich die Semantik der verfügbaren Prozessdaten auf den Namen der Prozessvariable sowie einige zusätzlich Attribute [Mahnke et al., 2009, S. 19]. OPC UA bietet ein Anlagenmetamodell, mit dem Unternehmen domänenspezifische Anlagenmodelle entwickeln können, die semantisch reichhaltige Anlageninformationen enthalten. Zu diesen gehören u.a. hierarchische Anlagenstrukturen, Typinformationen zu Anlagenteilen sowie unterschiedliche Sichten auf Modelle [vgl. Mahnke et al. 2009, S. 19; Lange et al. 2010, S. 112].

OPC UA bietet nach Mahnke et al. und Lange et al. eine Infrastruktur, mit der Unternehmen offene, plattformunabhängige und sichere Leitstandlösungen entwickeln können [Mahnke et al. 2009; Lange et al. 2010]. Viele aktuelle Diskussionen zu OPC UA werden überwiegend auf einer rein technischen Ebene geführt mit Fokus auf Plattformunabhängigkeit, verbesserten Datentransport, Webservices sowie Sicherheitsaspekte [vgl. Hannelius et al. 2008; Mahnke

et al. 2009; Lange et al. 2010; Enste und Mahnke 2011]. Anwenderorientierte Lösungsansätze zur Verbesserung der Arbeitssituationen der Operateure in Leitständen sind bislang rar. OPC UA ist zudem ein relativ neuer Standard. Es muss sich noch zeigen, ob er langfristig die Anforderungen der Unternehmen erfüllen kann und ähnlich einfach wie sein Vorgänger umzusetzen ist.

4.4.5 Diskussion

In dieser Arbeit werden Anlagenmodelle als Fundament für eine Leitstandmodernisierung betrachtet, um darauf aufbauend Werkzeuge zu entwickeln. Das Anlagenmodell wirkt sich auf viele Bereiche einer Leitstandsoftware aus und konserviert langfristig ein über die Jahre angesammeltes unternehmerisches Wissen. Die Wahl eines standardisierten Anlagenmetamodells stellt daher für Anlagenbetreiber eine essentielle Entscheidung dar. Erweist sich das Anlagenmetamodell langfristig als zu unflexibel oder etabliert sich der Standard nicht in der Industrie, können die daraus resultierenden Folgen geschäftskritisch sein. Ein späterer Wechsel auf einen anderen Standard ist meist sehr kostspielig und technologisch aufwändig. Die Wahl eines Anlagenmetamodells aus den vorgestellten Standards wird daher in dieser Arbeit anhand des Dimensionenmodells der Softwareentwicklung nach WAM diskutiert (siehe Abbildung 4.2). Dieses beschreibt die unterschiedlichen Einflüsse auf die Softwareentwicklung und sein Ergebnis, das Anwendungssystem. Das Dimensionenmodell zeigt die Wechselwirkungen zwischen den Einflüssen und bietet gerade für folgenreiche und langwierige Entscheidungen eine Hilfestellung für den Umgang mit Änderungen der Anforderungen im Lebenszyklus einer Software.

- **Anwendungsbereich:** Im WAM-Ansatz nimmt die Dimension *Anwendungsbereich* einen besonderen Stellenwert ein, da der Anwendungsbereich die fachliche Ausgangsbasis der Softwareentwicklung sowie den späteren Einsatzkontext bildet. „Der Anwendungsbereich ist gleichzeitig Grundlage für die Konstruktion des fachlichen Modells“ [Züllighoven et al., 1998, S. 309]. Dieses sollte nach WAM weitgehend frei von tech-

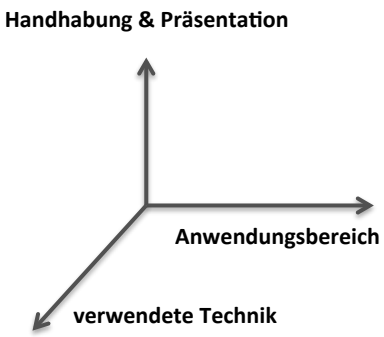


Abbildung 4.2: Dimensionenmodell der Softwareentwicklung nach WAM (Quelle: Züllighoven et al., 1998, S. 334)

nischen Faktoren erstellt werden, da es in erster Linie die Fachlichkeit vergegenständlicht.

- **Verwendete Technik:** Die Dimension *verwendete Technik* umfasst technische Faktoren, die eine unmittelbare Auswirkung auf die Konstruktion des Anwendungssystems haben, da sie potentielle Realisierungsmöglichkeiten auf das technisch Machbare eingrenzen.

- **Handhabung & Präsentation:** Die Dimension *Handhabung und Präsentation* zielt auf den Umgang der Anwender mit der Software ab. In interaktiven Systemen beeinflusst die Handhabung und Präsentation die softwaretechnische Realisierung. Grafische Oberflächen mit interaktiven Schaltflächen und Menüs stellen andere Anforderungen an eine Software, als beispielsweise Terminal-basierte Anwendungen.

In jeder Dimension verändern sich kontinuierlich die Anforderungen, die schließlich Veränderungen in einer Software herbeiführen. Gesetzesänderungen beeinflussen die fachlichen Anforderungen an Leitstandsoftware, technologische Innovationen wie z. B. mobile Geräte oder Plug-in-Technologien verändern die technische Basis, neue gestenbasierte Interaktionsformen mit Multitouch verändern die Handhabung und Präsentation einer Software [vgl. Beer et al., 2013, S. 66-70]. Entscheidend nach WAM ist, dass solche Veränderungen möglichst auf eine Dimension beschränkt werden. Dies hilft den Änderungsaufwand lokal zu halten und zu minimieren. Daraus erwächst die Anforderung, Veränderungen der Komponenten in der jeweiligen Dimension

so zu kapseln, dass die anderen Komponenten möglichst nicht betroffen sind [vgl. Züllighoven et al., 1998, S. 329ff].

In dieser Arbeit bildet das Anlagenmodell die fachliche Ausgangsbasis für die Leitstandmodernisierung. Es kann daher der Dimension *Anwendungsbereich* zugeordnet werden. Das Anlagenmodell sollte möglichst technologieunabhängig formuliert sein, da die Dimension *Anwendungsbereich*, im Vergleich zu anderen Dimensionen, über die Zeit deutlich weniger Änderungen unterworfen ist und das fachliche Wissen in Unternehmen über Jahre konserviert. Die technische Realisierung der Anlagenmodelle kann der Dimension *verwendete Technik* zugeordnet werden. Für diese Arbeit ist weniger die Wahl einer konkreten Technologie für die Realisierung eines Anlagenmodells relevant. Entscheidend ist, dass diese Technologie ein Metamodell auf der Ebene M_2 bietet, das die Konstruktion von Anlagenmodellen auf der Ebene M_1 anleitet und unterstützt. Das Metamodell sollte zudem eine flexible und erweiterbare Modellierungssprache aufweisen, die nicht nur aktuelle, sondern auch zukünftige Anforderungen an Anlagenmodelle erfüllen kann.

	Standards			
Kriterien	**IEC 61850**	**IEC 61346**	**CAEX**	**OPC UA**
Business Anforderungen				
Industrielle Verbreitung	**o**	+	–	–
Verständlichkeit	+	**o**	**o**	**o**
Technische Anforderungen				
Technologieunabhängigkeit	+	+	+	+
Standardisierter Datenaustausch	+	+	+	+
Usability Anforderungen				
Strukturähnlichkeit	**o**	+	+	+
Flexibilität	–	**o**	+	+
Metamodell-Eigenschaften				
Anlagenstruktur	+	+	+	+
Prozessvariablen	+	+	+	+
Metadaten zu Prozessvariablen	+	+	+	+
Typinformationen	**o**	**o**	+	+

Anforderung: – nicht erfüllt; **o** bedingt erfüllt; + zufriedenstellend erfüllt

Tabelle 4.1: Gegenüberstellung der Anforderungen an Anlagenmodelle und Standards (Quelle: eigene Erstellung)

In der Tabelle 4.1 sind aktuelle Standards sowie die Anforderungen an Anlagenmetamodelle gegenübergestellt und ihr Erfüllungsgrad bewertet. Diese Bewertung erfolgte subjektiv und auf Basis einer ausgiebigen Literaturrecher-

che [vgl. z. B. Epple 2003; Güttel und Fay 2008; Mahnke et al. 2011; Diedrich und Mühlhause 2011]. Prinzipiell bieten alle vorgestellten Anlagenmetamodelle die für diese Arbeit benötigten Metamodell-Eigenschaften. Die Auswahl eines Metamodells erfolgt daher anhand der in Abschnitt 4.1 vorgestellten technischen, Business- und Usability-Anforderungen.

Der Standard IEC 61850 hat einen klaren Fokus auf die Versorgungsindustrie. Er bietet ein für diese Domäne angepasstes Anlagenmetamodell mit einfachen Modellierungskonzepten, die jedoch nicht ohne weiteres auf andere Domänen übertragen werden können [vgl. Mahnke et al. 2011; Lehnhoff et al. 2011]. Der Standard IEC 61346 ist dagegen deutlich generischer und sein Metamodell auf diverse Domänen der Prozessindustrie übertragbar [Fröhlich et al., 2002b]. Gleiches gilt für den Standard CAEX (IEC 62424), dessen Metamodell ausreichend flexibel ist, um als Basis für ein Anlagenmodell zu dienen. Die in IEC 61346 und CAEX beschriebenen Modellierungskonzepte für technische Anlagen bilden nach Mahnke eine Teilmenge der Modellierungsmöglichkeiten des OPC-UA-Standards [vgl. Mahnke, 2011]. Dieser bietet somit eine umfangreichere und flexiblere Meta-Modellierungssprache. Das lässt darauf schließen, dass OPC UA das größere Potential zur Unterstützung zukünftiger Anforderungen birgt. Zusätzlich erweitert OPC UA den weit verbreiteten OPC-Standard und bietet somit eine gute Grundlage, in der Industrie akzeptiert zu werden. In dieser Arbeit wird daher die Leitstandmodernisierung mit einem Anlagenmodell auf Basis des OPC-UA-Standards vorgestellt. Zur Unterstützung der Kapselung der Dimensionen *Anwendungsbereich* und *verwendete Technik*, werden Ideen und Konzepte dieser Arbeit möglichst technologieunabhängig auf einer fachlich-konzeptionellen Ebene formuliert.

4.5 Anlagenmodellierung mit OPC UA

Bevor in dieser Arbeit ein OPC-UA-basiertes Anlagenmodell als Basis für die Leitstandmodernisierung verwendet wird, sollen zunächst das dem Standard zugrundeliegende Metamodell und ein dazu konformes Anlagenmodell exemplarisch vorgestellt werden. Diese sollen einen groben Eindruck der dahinter stehenden Konzepte vermitteln und werden im späteren Verlauf der Arbeit

wieder aufgegriffen. In der Abbildung 4.3 sind die grundlegenden Entitäten des OPC-UA-Metamodells ausschnittweise dargestellt. Eine ausführliche Beschreibung bieten beispielsweise Mahnke et al. [2009], Lange et al. [2010] und CAS [2010].

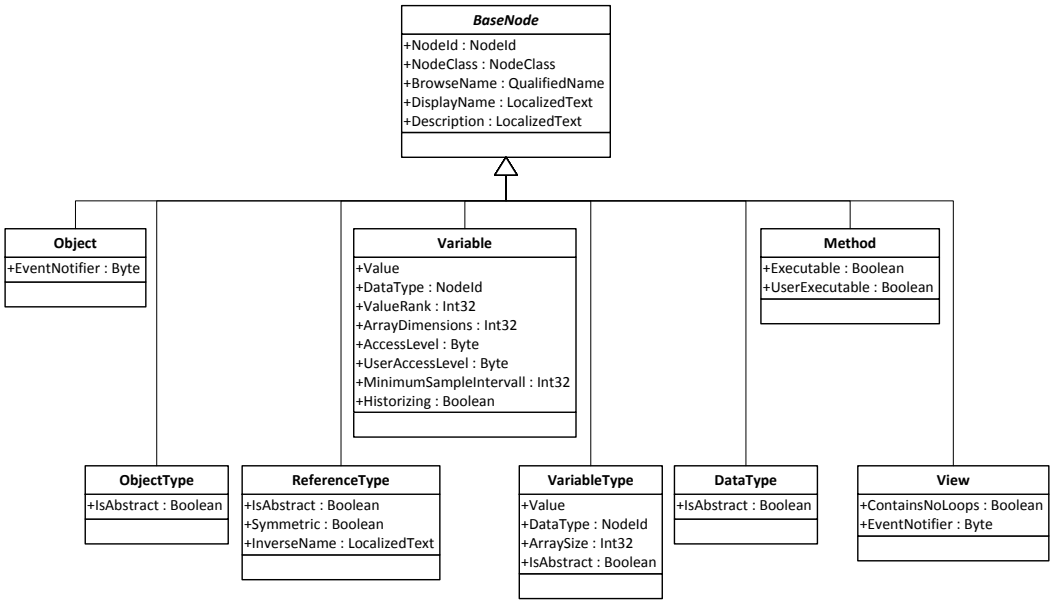


Abbildung 4.3: Abstrakte Darstellung des OPC-UA-Metamodells (Quelle: Enste und Mahnke, 2011)

Knoten (BaseNode) organisieren u.a. Anlagenteile (Object) und Prozessvariablen (Variable) in einer hierarchischen Struktur. Diese wird durch Referenztypen (ReferenceType) festgelegt, die auch andere Strukturen ermöglichen. Anlagenteilen, Prozessvariablen sowie Messwerten sind entsprechende Typen zugeordnet (ObjectType, VariableType, DataType). Sichten (Views) erlauben, Ausschnitte großer Modelle zu betrachten. Methoden übermitteln Befehle an Anlagenteile wie z. B. starte oder stoppe Motor. Damit nicht jedes Unternehmen den OPC-UA-Standard neu implementieren muss, bieten einige Produkthersteller kommerzielle Lösungen, mit denen Unternehmen individuelle Anlagenmodelle ausgestalten können[3]. Zum Zeitpunkt der Entstehung dieser Arbeit war jedoch keines der angebotenen Produkte von ausreichender Qualität für die Java-Plattform verfügbar, auf der die DESY-Leitstandsoftware basiert. Für diese Arbeit wurde daher ein zum OPC-UA-Metamodell konformes

[3]OPC UA SDKs and Toolkits: http://www.opcconnect.com/uakit.php

Anlagenmodell entwickelt. Dieses implementiert nur ausgewählte, speziell für diese Arbeit relevante Merkmale aus Abschnitt 4.

Die Abbildung 4.4 zeigt exemplarisch einen Ausschnitt des Anlagenmodells aus einer statischen Architektursicht als UML-Klassendiagramm. Die im Diagramm dargestellten Klassennamen sind in Abstimmung mit den DESY-Operateuren und Ingenieuren an die von ihnen verwendeten Begriffe angepasst[4]. Dies ist konform zum OPC-UA-Standard, der unternehmensspezifische Anpassungen explizit vorsieht [Mahnke et al., 2009, S. 123].

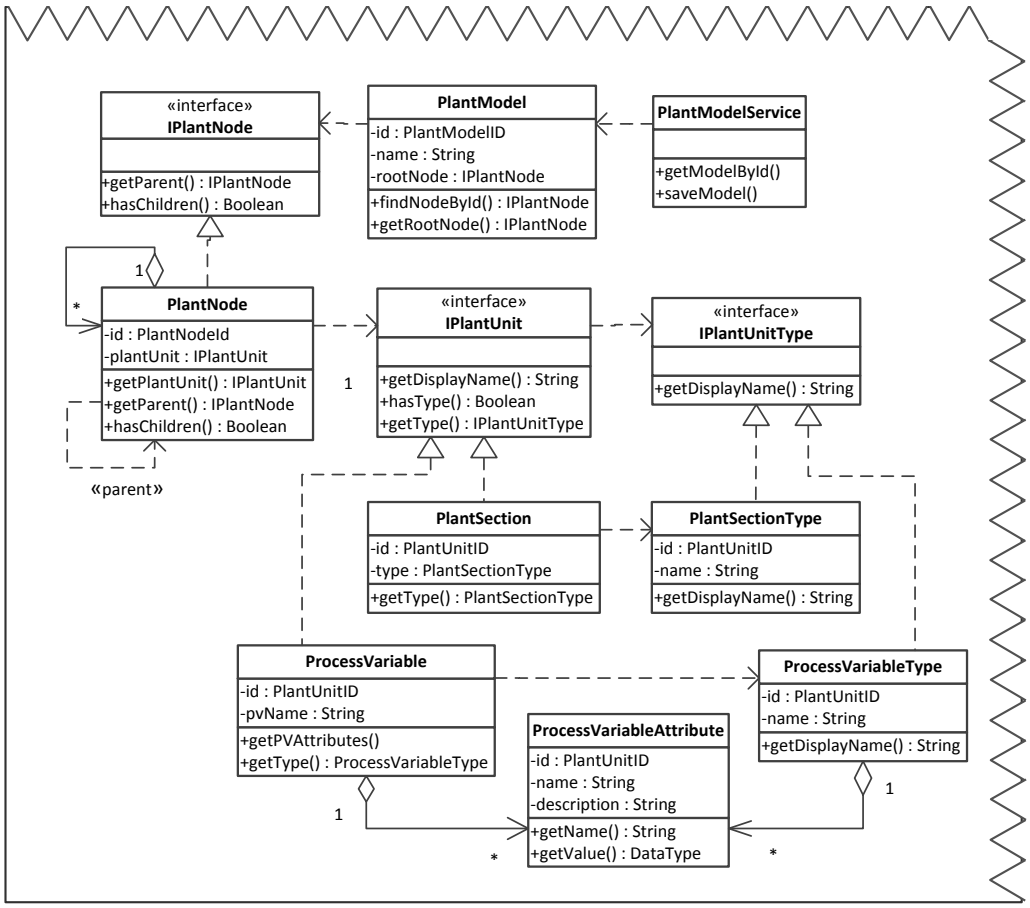


Abbildung 4.4: Ausschnitt eines OPC-UA-basierten Anlagenmodells (Quelle: eigene Erstellung)

Der `PlantModelService` verwaltet den Lebenszyklus der Anlagenmodelle. Er bietet Klienten eine Schnittstelle, mit der sie Anlagenmodell erzeu-

[4]Die Zusammenhänge zwischen dem OPC-UA-Metamodell und dem Klassendiagramm in Abbildung 4.4 sind dem Anhang B beigefügt.

gen, suchen und nach Änderungen persistieren können. Das Anlagenmodell (`PlantModel`) ist eine hierarchische Dekomposition aus Knoten (`PlantNode`). Diese referenzieren Anlagenteile (`PlantSection`) und Prozessvariablen (`ProcessVariable`) über ihre gemeinsame Schnittstelle `IPlantUnit`. Die Knoten-Klasse implementiert graphenspezifisches Verhalten wie z. B. die bidirektionale Navigation entlang der Baumstruktur oder Hinzufügen und Entfernen von Knoten und Blättern. Anlagenteilen und Prozessvariablen sind entsprechende Typen zugeordnet. Prozessvariablen können zudem über Attribute (`ProcessVariableAttribute`) mit Metadaten angereichert werden. Mit diesen Konzepten können Ingenieure unternehmensspezifische Anlagenmodelle ausgestalten und Operateuren in einer Leitstandsoftware zur Verfügung stellen. Die Abbildung 4.5 visualisiert exemplarisch solch ein Anlagenmodell für die DESY-Kryogenik-Anlage zur Laufzeit. Dieses Modell verdeutlicht die grundlegenden Konzepten wie hierarchische Anlagenstrukturierung, Typinformationen zu Anlagenteilen und Sichten auf Modelle.

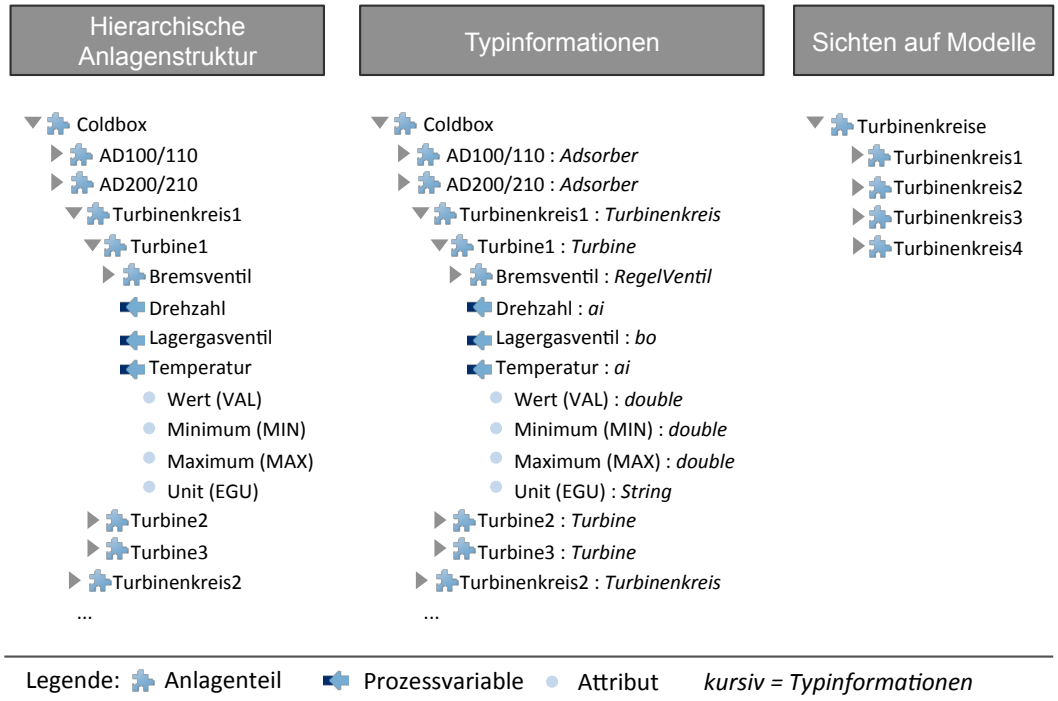


Abbildung 4.5: Visualisierung eines Kryogenik-Anlagenmodells (Quelle: eigene Erstellung)

In großen Industrieanlagen bestehen Anlagenmodelle oft aus tausenden Anlagenteilen und Prozessvariablen. Die manuelle Erstellung und Konfiguration

komplexer Modelle ist daher sehr aufwändig und fehleranfällig. In der Literatur werden unterschiedliche Ansätze diskutiert, wie Anlagenmodelle auf Basis bestehender Unternehmensdaten entstehen können [vgl. z. B. Schleipen 2008; Urbas und Doherr 2011; Reiswich und Fay 2012; Reiswich 2012a]. Zu den grundlegenden Herausforderungen dieser Aufgabe gehören die Analyse bestehender Datenquellen bezüglich ihrer Datenqualität und semantischer Äquivalenz, die Überführung dieser Daten in ein Anlagenmodell und die Synchronisation zwischen unterschiedlichen Datenquellen und Modellen. Da dieses Themengebiet sehr umfangreich ist und zudem weiterer Forschungsarbeit bedarf [vgl. z. B. Urbas und Doherr 2011 und Reiswich und Fay 2012], wird die Erstellung der Anlagenmodelle in dieser Arbeit nicht weiter vertieft, sondern erst im Ausblick wieder aufgegriffen. Diese Arbeit geht von einem bereits existierenden Anlagenmodell aus, in dem Ingenieure alle relevanten Anlagenteile entsprechend den Anforderungen im Abschnitt 4.1 modelliert haben.

4.6 Zusammenfassung

In diesem Kapitel wurden Anforderungen an Anlagenmodelle formuliert und ihre unternehmensübergreifenden Gemeinsamkeiten herausgearbeitet. Damit nicht jedes Unternehmen Anlagenmodelle grundlegend neu konstruieren muss, aber gleichzeitig seine speziellen Anforderungen bei der Anlagenmodellierung unterbringen kann, wurden in diesem Kapitel Anlagenmetamodelle diskutiert. Diese bieten eine Modellierungssprache, mit der Unternehmen konkrete, auf das jeweilige Unternehmen zugeschnittene Anlagenmodelle entwickeln können. Statt ein weiteres, eigenentwickeltes Anlagenmetamodell zu den bereits vorhandenen hinzuzufügen, wurden in diesem Kapitel aktuelle Metamodell-Standards vorgestellt, mit den Anforderungen dieser Arbeit verglichen und mit dem OPC-UA-Standard ein geeignetes Anlagenmetamodell gewählt.

Am Beispiel des Dimensionenmodells nach WAM wurde verdeutlicht, dass Anlagenmodelle Einflüssen verschiedener Dimensionen unterliegen. Diese Dimensionen haben einen abweichenden Lebenszyklus und sollen daher

möglichst lose gekoppelt werden. Schließlich wurden in diesem Kapitel ein OPC-UA-basiertes Metamodell und ein dazu konformes Anlagenmodell exemplarisch vorgestellt, um einen groben Eindruck der dahinter stehenden Konzepte zu vermitteln. Damit wurden in diesem Kapitel Aspekte der zweiten Forschungsfrage beantwortet und mit Anlagenmodellen sowie ihren Eigenschaften softwaretechnische Anforderungen an eine anwenderorientierte Leitstandmodernisierung formuliert.

Die OPC-UA-Technologie wird aktuell überwiegend aus einer technologiezentrierten und datenorientierten Sichtweise diskutiert. Dabei stehen die Plattformunabhängigkeit, Skalierbarkeit, Sicherheit, Datenübertragung sowie die Modellierung komplexer Datenstrukturen mit semantisch reichhaltigen Anlageninformationen im Vordergrund [vgl. z. B. Lange et al. 2010, S. 100; Mahnke et al. 2009, S. 19; Van Tan et al. 2009]. Bei dieser Diskussion sind für Anlagenbetreiber die Kosten der Einführung neuer Technologien, mit ihren weitreichenden Auswirkungen auf bestehende Systeme, deutlich sichtbar. Dagegen bleibt für sie der Nutzen neuer Technologien meist verborgen [vgl. Hannelius et al., 2008]. Bislang fehlen anwenderorientierte Ansätze, die das Potential der Anlagenmodelle aus Sicht der Operateure aufzeigen und die damit einhergehenden Vorteile methodisch auf die Konstruktion neuer Leitstandfunktionalität übertragen. Es fehlen ferner Ansätze, die neue Leitstandfunktionalität nach den Prinzipien der Softwaretechnik mit wenigen Eingriffen in bestehende Systeme einführen, ohne diese massiv zu verändern.

Im folgenden Kapitel werden die anwenderorientierten Lösungsansätze aus Abschnitt 3 auf Anlagenmodelle übertragen und am Beispiel der Leitstandaufgabe *Alarmierung* die grundlegende Herangehensweise dieser Arbeit vorgestellt.

Kapitel 5

Kurzvorstellung Lösungsansatz am Beispiel Alarmierung

In Kapitel 3 wurden aktuelle Leitstandprobleme und dazu passende Lösungsansätze aus einer anwenderorientierten und technologiezentrierten Sicht gegenübergestellt und diskutiert. Dabei wurde herausgearbeitet, dass technologiezentrierte Ansätze oft nicht den Kern der Leitstandprobleme adressieren und einen Großteil ungelöster Probleme auf Operateure verlagern. In dieser Arbeit wurde daher ein anwenderorientierter Ansatz motiviert, um aus Sicht der Operateure relevante Leitstandprobleme zu erkennen und dafür passende Lösungsansätze zu erarbeiten. Als gemeinsamen Kern dieser Lösungsansätze wurde ein strukturähnliches Anlagenmodell herausgearbeitet. Dieses dient als Fundament für eine Leitstandmodernisierung und gehört somit zur Anforderung an die Architektur einer Leitstandsoftware. In Kapitel 4 wurden aktuelle Anlagenmodell-Standards vorgestellt und mit dem OPC-UA-Standard ein geeigneter Modell-Kandidat gewählt. Zur Verdeutlichung der grundlegenden Herangehensweise dieser Arbeit, wird in diesem Abschnitt die anwenderorientierte Sichtweise mit strukturähnlichen Anlagenmodellen kombiniert und auf die Leitstandaufgabe Alarmierung übertragen.

5.1 Modellbasiertes Alarm-Werkzeug

Ausgangspunkt dieser Arbeit für den Umgang mit Alarmproblemen ist das Übereinkommen der Experten, dass Alarmprobleme, trotz eines ausgeklügelten Alarmmanagements, nicht gänzlich vermieden werden können (siehe Abschnitt 3.2). In solchen Situationen benötigen Operateure Werkzeuge, die ihnen den Umgang mit Alarmproblemen erleichtern. Statt bestehende Alarm-Werkzeuge zu ersetzen, werden in dieser Arbeit vorhandene Werkzeuge erhalten und gezielt durch neue Werkzeuge ergänzt. Die neuen Werkzeuge adressieren dabei systematisch die Unzulänglichkeiten vorhandener Werkzeuge und bieten Operateuren einen verbesserten Umgang mit wiederkehrenden Alarmproblemen. Aus dieser Zielsetzung resultiert das in der Abbildung 5.1 dargestellte Mock-up[1] für ein Alarm-Werkzeug, mit folgenden Verbesserungen für Operateure:

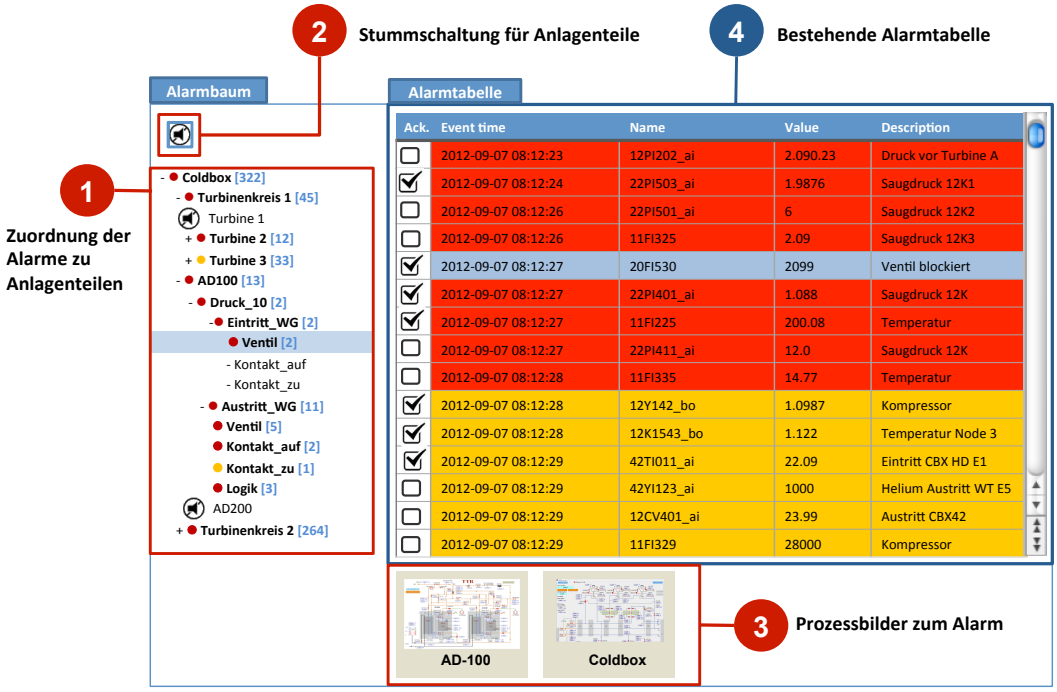


Abbildung 5.1: Modellbasiertes Alarm-Werkzeug am Beispiel der Fallstudie DESY (Quelle: eigene Erstellung aufbauend auf Pappert, 2012)

[1]Ein Mock-up ist in der Softwareentwicklung eine günstige, oft aus Papier oder mit einer Präsentations-Software gefertigte Attrappe einer grafischen Oberfläche des zu entwickelnden Systems, die in frühen Phasen der Softwareentwicklung für Demonstrations- und Diskussionszwecke eingesetzt wird [vgl. z. B. Bäumer et al., 1996].

- **Umgang mit Alarmfluten:** Im Abschnitt 3.2 wurden Alarmfluten vorgestellt. Dabei treffen in kurzer Zeit mehr Alarme ein, als Operateure verarbeiten können. Für dieses Problem wurden in Abschnitt 3.2.1 Ansätze herausgearbeitet, die Alarme den betroffenen Anlagenteilen zuordnen oder Alarme anhand der kausalen Beziehungen zwischen Anlagenteilen gruppieren. Wie in Abschnitt 2.3.2 beschrieben, entstehen die meisten Alarmfluten während Anlagenteile gewartet werden oder komplett ausfallen und dadurch Folgealarme auslösen. Eine Zuordnung der Alarme zu den betroffenen Anlagenteilen, wie von der EEMUA vorgeschlagen, kann daher die Übersicht in einer Alarmflut signifikant erhöhen. Dazu wird in dieser Arbeit die hierarchische Struktur der Anlagenmodelle aus Abschnitt 4.1 herangezogen. Damit können Alarme, neben der tabellarischen Darstellung (Punkt 4 der Abbildung 5.1), zusätzlich in einer Anlagenstruktur verortet werden (Punkt 1). Dieser Ansatz berücksichtigt nicht die kausalen Beziehungen zwischen Anlagenteilen und Alarmen. Er hilft jedoch Operateuren, mit weniger Modellierungs- und Konfigurationsaufwand, auf einen Blick zu erkennen, welche Anlagenteile ausgefallen sind und infolgedessen eine Alarmflut ausgelöst haben.

  Statt hunderte Alarme unstrukturiert in einem *Alarmtabellen-Werkzeug* zu verarbeiten, können Operateure Alarme zusätzlich in einem *Alarmbaum-Werkzeug* analysieren. Dabei können sie Alarme entlang der Anlagenhierarchie auf unterschiedlichen Aggregationsstufen betrachten und bei Bedarf die Hierarchiestufen bis zu den Prozessvariablen hinabsteigen. Aufeinander abgestimmte Farbcodes im Alarmbaum- und Alarmtabellen-Werkzeug weisen auf die Schwere darin aufgetretener Alarme hin. Treten in einem Anlagenteil mehrere Alarme oder Warnungen auf, wird im Alarmbaum-Werkzeug auf der obersten Hierarchieebene immer der schwerste Vorfall visualisiert. Zusätzlich wird jedem Anlagenteil die Anzahl darin aufgetretener Alarme zugeordnet. Die hierarchische und tabellarische Alarmdarstellung sind dabei keine losgelösten Werkzeuge die nebeneinander existieren. Sie ergänzen sich und interagieren miteinander. Während das Alarmbaum-Werkzeug eine strukturierte Darstellung ganzer Alarmfluten ermöglicht, bietet das Alarmtabellen-Werkzeug Detailinformationen zu ausgewählten Alarmen.

Wählt ein Operateur einen Alarm im Alarmtabellen-Werkzeug aus, wird im Alarmbaum-Werkzeug der Baum entsprechend aufgeblättert und das dazugehörige Anlagenteil selektiert. Umgekehrt werden bei Auswahl eines Anlagenteils im Alarmbaum-Werkzeug die dazugehörigen Alarme im Alarmtabellen-Werkzeug gefiltert.

- **Umgang mit unnötigen Alarmen:** Wie im Abschnitt 3.2 beschrieben, versenden Anlagenteile in Wartung oder Testbetrieb unnötige Alarme. Statt komplexe Filterregeln zu konfigurieren, die Operateure nach einer abgeschlossenen Wartung oder Testbetrieb nicht selten vergessen wieder zu entfernen, bieten anwenderorientierte Lösungsansätze Operateuren die Möglichkeit, die entsprechenden Anlagenteile unmittelbar stumm zu schalten. Dazu wird in dieser Arbeit die Anlagenstruktur genutzt, entlang der Operateure zu den Anlagenteilen in Wartung oder Testbetrieb navigieren und über eine Stummschaltung dazugehörige Alarme deaktivieren können (Punkt 2 der Abbildung 5.1). Stummgeschaltete Anlagenteile werden durch ein entsprechendes Symbol im Alarmbaum-Werkzeug explizit visualisiert. Dadurch haben Operateure stets im Blick, welche Anlagenteile aktuell aus der Alarmierung ausgeschlossen sind. Nach einer abgeschlossenen Wartung oder Testbetrieb können Operateure die Stummschaltung für ausgewählte Anlagenteile aufheben und dafür fortan Alarme empfangen.

- **Prozessbildsuche für Alarme:** In Abschnitt 3.2 wurden für die Suche nach passenden Prozessbildern zu Alarmen manuell konfigurierte Abbildungstabellen mit Sprungmarken vorgestellt. Diese statisch konfigurierten Abbildungstabellen erodieren über die Zeit. Anwenderorientierte Ansätze automatisieren die manuelle und fehleranfällige Konfiguration der Abbildungstabellen durch Algorithmen, die Beziehungen zwischen Prozessbildern und Alarmen automatisch erkennen und visualisieren. Dazu werden in dieser Arbeit die Prozessvariablen der Anlagenmodelle als Bindeglied zwischen Prozessbildern und Alarmen genutzt, da sie in beiden vorkommen. Dadurch können Algorithmen die Prozessvariable eines eingehenden Alarms extrahieren und dazu passende Prozessbilder suchen, in denen die extrahierte Prozessvariable repräsentiert ist. Da

eine Prozessvariable in mehreren Prozessbildern visualisiert sein kann, werden im *Prozessbildauflister-Werkzeug* auch mehrere Prozessbilder dargestellt (Punkt **3** der Abbildung 5.1). Aus diesen können Operateure aufgrund ihrer Erfahrung ein für die jeweilige Situation passendes Prozessbild auswählen und darin die gegenwärtige Alarmsituation nachvollziehen.

Dies sind nur einige Beispiele wie eine anwenderorientierte Sichtweise hilft, systematisch Lösungsansätze für relevante Leitstandprobleme zu identifizieren und Werkzeuge zu entwickeln, die besser an die jeweiligen Stärken und Schwächen von Mensch und Maschine angepasst sind. Die neuen *Alarmbaum*- und *Prozessbildauflister*-Werkzeuge ersetzen dabei nicht das vorhandene Alarmtabellen-Werkzeug , sondern ergänzen dieses gezielt dort, wo es Unzulänglichkeiten aufweist.

5.2 Zusammenfassung

In diesem Abschnitt wurde die anwenderorientierte und modellbasierte Herangehensweise der vorangegangenen Kapitel kombiniert und exemplarisch auf das Beispiel Alarmierung übertragen. Das in der Abbildung 5.1 dargestellte Mock-up verdeutlicht, wie neue Alarm-Werkzeuge Operateure beim Umgang mit der Komplexität in Alarmsituationen unterstützen können. Der bisherige Fokus der Arbeit und die dazugehörigen Ergebnisse sind in der Abbildung 5.2 im WAM-Dimensionenmodell verortet. In den vorangegangenen Kapiteln wurde vorwiegend die Dimension *Anwendungsbereich* betrachtet, um zunächst aus einer fachlichen Sicht die *Leitebene* technischer Anlagen (siehe Automatisierungspyramide in Abschnitt 2.2) zu verstehen und darin auftretende Probleme sowie dazu passende Lösungsansätze zu sondieren. Als gemeinsame Anforderung an die identifizierten Lösungsansätze wurden strukturähnliche Anlagenmodelle herausgearbeitet und in der Dimension *verwendete Technik* mit der OPC-UA-Technologie ein geeigneter Modell-Kandidat gewählt. Die Realisierung der anwenderorientierten und modellbasierten Lösungsansätze wurde am Beispiel des Mock-ups der Abbildung 5.1 verdeutlicht und dabei die Handhabung & Präsentation der neuen Werkzeuge demonstriert.

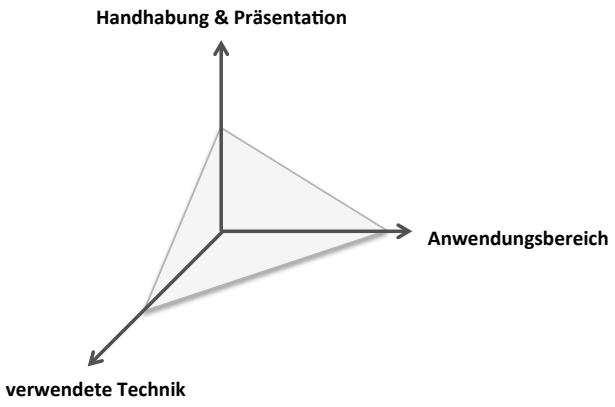


Abbildung 5.2: Einordnung der Ergebnisse im WAM-Dimensionenmodell (Quelle: eigene Erstellung)

Für die Umsetzung der identifizierten Lösungsansätze muss die neue Leitstandfunktionalität nach Sauter in der Lage sein, trotz ihrer technologischen Überlegenheit, bestehende Altsysteme einzubeziehen, um ökonomisch erfolgreich zu sein [vgl. Sauter, 2005, S. 262]. Altsysteme verfügen jedoch nur selten über geeignete Schnittstellen, um neue IT-Lösungen, wie z. B. Anlagenmodelle und darauf aufbauende Werkzeuge, nahtlos zu integrieren. Zudem weisen viele Altsysteme, aufgrund kontinuierlicher Anpassungen, erodierte Softwarearchitekturen auf. Dies erschwert Modifikationen und Erweiterungen in vorhandenen Systemen [vgl. Lehman 1984; Sommerville 2001, S. 589ff]. Vor einer Leitstandmodernisierung müssen daher Altsysteme zunächst in den Zustand versetzt werden, in dem sie strukturelle Modifikationen prinzipiell erlauben und neue Funktionalität aufnehmen können. Dieser Aspekt wird im nächsten Kapitel behandelt.

Kapitel 6

Umgang mit Legacy-Leitstandsystemen

Technische Anlagen haben eine durchschnittliche Lebensdauer von 20 bis 50 Jahren (siehe Abschnitt 2.1). In dieser Zeit werden die Anlagen laufend an neue Anforderungen angepasst. Dabei werden der physische Aufbau der Anlage und die zur Steuerung und Überwachung eingesetzten Softwaresysteme verändert, wobei letztere viel häufiger angepasst oder ausgetauscht werden [vgl. Sauer und Ebel, 2007b, S. 331]. Zu den in Abschnitt 4.4.5 vorgestellten Einflussfaktoren auf die Softwareentwicklung kommt gerade bei langlebigen Softwaresystemen ein weiterer Faktor hinzu: *Zeit*. Software muss sich nach Lehman über die Zeit verändern, um ihren Nutzen für die Umwelt zu erhalten [Lehman, 1984]. Mit jeder neuen Version nimmt dabei die Funktionalität, häufig aber auch die Softwarekomplexität zu [Lehman, 1984, S. 20]. Dadurch erodieren Softwaresysteme langsam zu sog. Legacy-Systemen[1], die Bennett informell wie folgt beschreibt: „[...] large software systems that we don't know how to cope with but that are vital to our organization“ [Bennett, 1995, S. 19].

Die Abbildung 6.1 demonstriert die Auswirkungen kontinuierlicher Erweiterungen und Anpassungen einer Software in ihrem Lebenszyklus. In dem sog. *Architekturkorridor* gehen strukturelle Änderungen und Erweiterungen

[1] engl. legacy = Vermächtnis

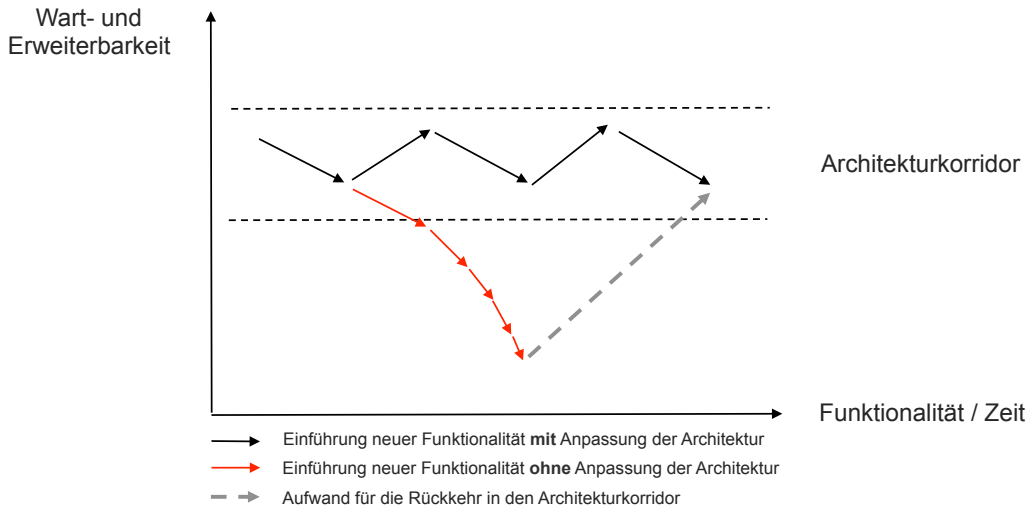


Abbildung 6.1: Architekturerosion in Softwaresystemen (Quelle: in Anlehnung an Magnussen, 2012)

einer Software mit einer kontinuierlichen Anpassung der Softwarearchitektur einher (schwarze Pfeile). Dadurch können sie mit ökonomisch vertretbaren Mitteln realisiert werden. Änderungen, die ohne Anpassungen der Architektur durchgeführt werden (rote Pfeile), führen zu einer schleichenden Softwareerosion. Diese beeinträchtigt die Wart- und Erweiterbarkeit einer Software. Dadurch kann immer weniger Funktionalität in der gleichen Zeit realisiert werden (kürzer werdende rote Pfeile). Mit jeder Erweiterung einer Software ohne Anpassung der Architektur steigen die Aufwände für die Rückführung der Software in den Architekturkorridor (gestrichelter grauer Pfeil). Diese Entwicklung führt langfristig zum Verfall einer Software zum Legacy-System. In Unternehmen mit Legacy-Systemen steigen die Kosten für Wartung und Erweiterung unvermeidlich. Die Ursachen dafür sind vielfältig und liegen beispielsweise darin begründet, dass Legacy-Systeme in unterschiedlichen Programmiersprachen implementiert sind. Dafür ist nur schwer Personal mit passenden Kenntnissen zu finden. Zudem stellt der Quellcode des Systems oft die einzig vorhandene Dokumentation dar [Sommerville, 2001, S. 90]. In der Softwaretechnik werden daher seit Anfang der 80er-Jahre vermehrt Ansätze zum Umgang mit Legacy-Systemen diskutiert. Dabei werden folgende Begriffe unterschieden [vgl. Sneed et al., 2010, S. 25ff]:

- **Neuentwicklung:** Von einer Neuentwicklung spricht man, wenn Software gänzlich neu geschrieben wird oder nur wenige Teile des alten Systems bei der Entwicklung wiederverwendet werden.

- **Softwareevolution:** Unter einer Softwareevolution wird eine inkrementelle Veränderung eines bestehenden Systems über den gesamten Lebenszyklus hinweg verstanden. Die Softwareevolution verfolgt das Ziel, die langfristige Wartbarkeit eines Systems zu erhalten, strukturelle Änderungen durchzuführen und die Architektur zu verbessern. „Softwareevolution ist nicht auf Legacy-Systeme begrenzt, sondern ist für alle Systeme erforderlich, die einen hohen Wert für ein Unternehmen haben“ [Riebisch und Bode, 2009].

- **Migration:** Bei einer Migration wird ein Softwaresystem in eine neue Umgebung und/oder Programmiersprache überführt. Dabei bleibt üblicherweise die Funktionalität im ersten Schritt unverändert. Eine Migration verlängert das Leben des migrierten Systems, in dem es eine Infrastruktur schafft, in dem das migrierte System zu angemessenen Kosten weiter betrieben werden kann [vgl. Sommerville, 2001, S. 601].

- **Integration:** Bei einer Integration werden verschiedene, meist heterogene Anwendungssysteme miteinander verbunden, damit diese auf gleiche Daten zugreifen oder Dienste austauschen können.

Langlebige Leitstandsysteme beinhalten ein über mehrere Jahre angesammeltes Wissen sowie Erfahrungen und sind somit für Unternehmen wertvoll [vgl. Bennett, 1995]. Ihr Betrieb ist häufig an eine gesamte Infrastruktur bestehend aus Hardware und Unterstützungssoftware gekoppelt, so dass eine Neuentwicklung, Migration oder der Wechsel zu Standardlösungen sehr kostspielig ist und viele Risiken birgt. Zudem laufen viele Leitstandsysteme in einem 24/7 Betrieb, wodurch größere Modifikationen den Anlagenbetrieb stark beeinträchtigen würden und daher nahezu ausgeschlossen sind. „Folglich suchen viele Unternehmen nach Verfahren des Software-Engineerings, welche die Lebensdauer von Legacy-Systemen verlängern und die Kosten für den Erhalt dieser Systeme verringern“ [Sommerville, 2001, S. 591]. Aus diesen Randbedingungen leitet sich das Vorhaben dieser Arbeit ab, bestehende

Leitstandsysteme zu erhalten und nach den Prinzipien der Softwaretechnik mit wenigen Eingriffen zu modernisieren. Diese Arbeit wird daher der Softwareevolution zugeordnet.

Im nächsten Abschnitt wird die Softwareevolution eingehender untersucht, Eigenschaften evolutionsfähiger Systeme herausgearbeitet und ausgewählte Reengineering-Maßnahmen vorgestellt, wie Legacy-Leitstandsysteme diese Eigenschaften wiedererlangen können.

6.1 Softwareevolution

Vor der Einführung neuer Funktionalität müssen Legacy-Leitstandsysteme zunächst in einen Zustand versetzt werden, in dem sie strukturelle Modifikationen und Erweiterungen ohne Beeinträchtigungen des vorhandenen Systems erlauben. In der Vergangenheit haben Softwareentwickler diese Aufgabe schlicht als *Softwarewartung* bezeichnet. Darunter versteht der Standard IEEE 1219-1998:

BEGRIFF 6.1: SOFTWARE MAINTENANCE
Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.

Lehman, Floyd et al., Rajlich und Bennett sowie Chapin et al. sehen es jedoch als notwendig an, die Aufgaben nach der initialen Auslieferung einer Software weiter zu differenzieren, da ihre Inhalte und die Intensität der Aufgaben deutlich variieren [Lehman 1984; Floyd et al. 1997; Rajlich und Bennett 2000; Chapin et al. 2001]. Zur besseren Strukturierung dieser Aufgaben führen Rajlich und Bennett ein Phasenmodell des Software-Lebenszyklus ein (Abbildung 6.2). In diesem Phasenmodell werden im Folgenden die Phasen *Evolution* und *Servicing* näher betrachtet, da darin der überwiegende Teil einer Leitstandmodernisierung stattfindet [Rajlich und Bennett, 2000]:

- **Evolution:** Im Falle einer erfolgreichen Entwicklung und initialen Auslieferung betritt eine Software die sog. Evolutionsphase. Unter einer

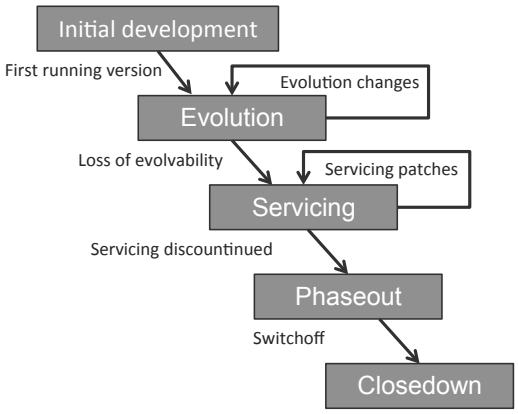


Abbildung 6.2: Phasenmodell des Softwarelebenszyklus (Quelle: Rajlich und Bennett, 2000)

Softwareevolution (engl. Software Evolvability) verstehen Breivold et al. [2007]:

> **BEGRIFF 6.1: SOFTWARE EVOLVABILITY**
> *Software evolvability is the ability of a software system to adjust to change stimuli, i.e. changes in requirements and technologies that may have impact on the software system in terms of software structural and/or functional enhancements, while still taking the architectural integrity into consideration.*

Die Softwareevolution trägt somit dazu bei, eine Software langfristig an sich ändernde wirtschaftliche, organisatorische und technologische Rahmenbedingungen anpassen zu können und dabei die Integrität der Softwarearchitektur zu erhalten.

- **Servicing:** Die Evolutionsfähigkeit einer Software wird nach Rajlich und Bennett maßgeblich durch die Softwarearchitektur sowie das Entwicklerteam beeinflusst [Rajlich und Bennett, 2000]. Weisen beide Unzulänglichkeiten auf, betritt die Software die *Servicing*-Phase, in der jegliche Codeänderungen mit viel Aufwand verbunden sind. In Folge vermeiden Entwickler Änderungen im System oder kapseln komplexe Programmteile durch Konzepte wie *Wrapper*, um die Einführung neuer Funktionalität zu vereinfachen. Jeder dieser Eingriffe erhöht jedoch die Softwarekomplexität und trägt zur Softwareerosion bei.

In Softwaresystemen, die den Architekturkorridor der Abbildung 6.1 verlassen haben und sich in der Servicing-Phase befinden, sind Änderungen nur unter hohem Aufwand möglich und im Extremfall sogar ausgeschlossen. Vor einer Modernisierung müssen bestehende Leitstandsysteme die Eigenschaften evolutionsfähiger Systeme wiedererlangen.

6.1.1 Eigenschaften evolutionsfähiger Systeme

Legacy-Systeme zeichnen sich üblicherweise durch eine schwer verständliche, schlecht wartbare und erweiterbare Software aus. In solchen Fällen helfen Reengineering-Maßnahmen, die Softwarequalität zu verbessern und dadurch zu gewährleisten, dass Systeme weiterhin in Betrieb bleiben [Sommerville, 2001, S. 632].

> *„It is not age that turns a piece of software into a legacy system, but the rate at which it has been developed and adapted without having been reengineered“* [Demeyer et al., 2008, S. 6].

Unter Reengineering verstehen [Demeyer et al., 2008, S. 3]:

BEGRIFF 6.2: REENGINEERING
The goal of reengineering is to reduce the complexity of a legacy system sufficiently that it can continue to be used and adapted at an acceptable cost.

Dieses Ziel kann nach Sneed et al. durch eine Verbesserung der Architektur (Reengineering im Großen) oder durch Code-Sanierung (Reengineering im Kleinen) erreicht werden [Sneed et al., 2010, S. 28]. „Reengineering im Großen erreicht die Verbesserung der Architektur durch Techniken wie *Clustering* und *Decoupling*. Reengineering im Kleinen verbessert den Code durch Techniken wie *Restructuring* und *Refactoring*“ [Sneed et al., 2010, S. 28]. Vor einem Reengineering müssen Softwareentwickler zunächst festlegen, welche Eigenschaften sie in einer Software verbessern wollen. Evolutionsfähige Softwaresysteme erfüllen nach Breivold et al. die folgende Kriterien [Breivold et al., 2007]:

- **Analysierbarkeit:** Die Fähigkeit, Änderungen in der fachlichen Welt schnell in der Software zu lokalisieren und die Auswirkungen einer Änderung auf das Softwaresystem abschätzen zu können. Riebisch und Bode schließen bei der Analysierbarkeit auch Verständlichkeit ein, denn Programmverstehen ist Voraussetzung für eine Analyse [vgl. Riebisch und Bode, 2009, S. 340].

- **Änderbarkeit:** Die Fähigkeit, eine Software an veränderte Anforderungen anpassen zu können, ohne negative Auswirkungen auf den Rest des Softwaresystems. Änderbarkeit umspannt Konzepte wie Kopplung, Kohäsion, Modularität und Softwarekomplexität [Breivold et al., 2007, S. 99].

- **Integrität:** Die Fähigkeit einer Software, nach jeder durchgeführten Änderung die ursprünglichen Architekturkonzepte und Richtlinien einzuhalten. Dies soll einen schleichenden Architekturverfall verhindern.

- **Erweiterbarkeit:** Die Fähigkeit, eine Software einfach mit neuer Funktionalität erweitern zu können, ohne dabei negative Auswirkungen auf das bestehende System zu verursachen.

- **Portierbarkeit:** Die Fähigkeit, ein System in eine andere Software- oder Hardwareumgebung zu überführen.

- **Testbarkeit:** Die Fähigkeit, eine Software nach durchgeführten Änderungen auf Fehler oder Seiteneffekte überprüfen zu können.

Bei der Einführung neuer Funktionalität in bestehende Leitstandsysteme werden diese Eigenschaften zu Schlüsselfaktoren, die eine Leitstandmodernisierung begünstigen oder erschweren können. Dabei ist nicht entscheidend, dass Leitstandsysteme alle aufgeführten Kriterien in vollem Maße erfüllen (was ohnehin schwer messbar ist), sondern dass sie sich in einem Zustand befinden „[...] in dem noch strukturelle Änderungen möglich sind und nicht nur (korrigierende) Wartungstätigkeiten“ [Riebisch und Bode, 2009, S. 342]. Von den aufgeführten Kriterien der Softwareevolution werden in dieser Arbeit Analysierbarkeit, Änderbarkeit sowie Erweiterbarkeit näher betrachtet, da diese Kriterien in zahlreichen Artikeln im Zusammenhang mit der Einführung

neuer Funktionalität wiederholt genannt werden [vgl. z. B. Lehman und Ramil 2002, S. 276; Bennett und Rajlich 2000, S. 77; Demeyer et al. 2008, S. 4ff]. In den folgenden Abschnitten werden Reengineering-Maßnahmen vorgestellt, die zur Verbesserung dieser Kriterien in einer Leitstandsoftware beitragen.

6.1.2 Maßnahmen zur Verbesserung der Analysierbarkeit

Nach Züllighoven et al. sowie Siltanen und Pärnänen kann die Analysierbarkeit eines Systems durch eine hohe Strukturähnlichkeit zwischen den softwaretechnischen Komponenten und den Gegenständen des Anwendungsbereichs verbessert werden [Züllighoven et al. 1998, S. 263; Siltanen und Pärnänen 2006, S. 348]. „Diese Strukturähnlichkeit bringt zwei entscheidende Vorteile: Die Anwender finden die Gegenstände ihrer Arbeit und die Begriffe ihrer Fachsprache im Anwendungssystem repräsentiert. Sie können dementsprechend ihre Arbeit in gewohnter Weise organisieren. Die Entwickler können Softwarekomponenten und Anwendungskonzepte bei fachlichen und softwaretechnischen Änderungen zueinander in Beziehung setzen und somit die wechselseitigen Abhängigkeiten erkennen“ [Züllighoven et al., 1998, S. 263f]. In einer technischen Anlage sind die primären Gegenstände des Anwendungsbereichs die physische Anlage sowie die Anlagenteile. Ihre softwaretechnischen Repräsentanten bilden gemeinsam das Anlagenmodell. In dieser Arbeit wurde in Abschnitt 3.4 eine hohe Strukturähnlichkeit für Anlagenmodelle gefordert, die Entwickler dabei unterstützen soll, Änderungen in der technischen Anlage schnell in der Leitstandsoftware zu lokalisieren und die daraus resultierenden Auswirkungen besser abzuschätzen. Dadurch leistet das Anlagenmodell einen wichtigen Beitrag zur Analysierbarkeit einer Leitstandsoftware.

Ein Anlagenmodell ist zwar ein wesentlicher, aber kein exklusiver Bestandteil einer Leitstandsoftware und muss sich in eine Gesamtarchitektur fügen. Nach Züllighoven et al. werden für die Konstruktion großer objektorientierter Systeme neben den Mikroelementen auch Makrostrukturen benötigt, die eine überschaubare und handhabbare Organisation dieser Systeme erlauben [Züllighoven et al., 1998, S. 662ff]. „Eine solche Strukturierung von Software

nach Modellierungseinheiten, die Makrostrukturen entsprechen, nennen wir Softwarearchitektur“ [Züllighoven et al., 1998, S. 663]. In Abschnitt 1.3.3 wurde die WAM-Architektur zur Konstruktion interaktiver Anwendungen eingeführt. In WAM geben Organisationen, die nach Sparten, Produkten oder anderen Prinzipien aufgebaut sind, anwendungsorientierte Makrostrukturen vor [vgl. Züllighoven et al. 1998, S. 696ff; Bäumer 1998, S. 116ff; Lilienthal 2008, S. 32f]. Neben den Mikrostrukturen, vergegenständlichen auch Makrostrukturen die fachliche Orientierung bei der Anwendungsentwicklung und tragen dadurch zur Analysierbarkeit eines Anwendungssystems bei.

6.1.3 Maßnahmen zur Verbesserung der Änderbarkeit und Erweiterbarkeit

Obwohl Breivold et al. die Eigenschaften *Änderbarkeit* und *Erweiterbarkeit* differenzieren, sind die Grenzen zwischen diesen Eigenschaften relativ unscharf [Breivold et al., 2007]. Das wird beispielsweise anhand der Vorschläge von Meyer zur Verbesserung der Erweiterbarkeit deutlich, die sich ebenfalls auf die Änderbarkeit übertragen lassen [Meyer, 1997, S. 7]:

- **Einfachheit:** Änderungen lassen sich besser mit einfachen Architekturen umsetzen als mit komplexen. Die Komplexität eines Softwaresystems ist jedoch nach Hasselbring und Lilienthal schwer zu fassen, ihre Bewertung sehr subjektiv und auf verschiedenen Ebenen der Softwareentwicklung anzutreffen: im Softwaresystem, beim Menschen selbst und in der Interaktion der Entwicklungsteams [vgl. Hasselbring 2006; Lilienthal 2008, S. 5]. Dadurch lassen sich treffende Aussagen, wie komplex oder einfach eine Softwarearchitektur ist, nur schwer formulieren. Nach Lilienthal entsteht die Komplexität in Softwaresystemen vor allem durch die vielen atomaren Softwareartefakte, die Entwickler neu erschaffen oder wiederkehrend anpassen und miteinander verknüpfen, um Software für unterschiedliche Problemfelder zu entwickeln. Lilienthal schlägt mehrere Strategien vor, wie diese Komplexität reduziert werden kann. Dazu gehören beispielsweise die Modellbildung, bei der ein vereinfachtes Modell der Umwelt in das Softwaresystem integriert wird, sowie eine

geeignete Architektur, die vor allem „... eine Sprachebene schafft, so dass Entwurfsdiskussionen spezielle Kristallisationspunkte bekommen, an denen das Team Entwurfsentscheidungen ausrichten kann“ [Lilienthal, 2008, S. 152]. Neben diesen Strategien tragen nach Riebisch und Bode auch agile Vorgehensmodelle dazu bei, einfache Lösungen zu finden und unnötige Komplexität zu vermeiden [Riebisch und Bode, 2009, S. 342].

- **Modularisierung:** Während die modulare Anwendungsentwicklung früher als eine Zusammenfügung von Subroutinen aufgefasst wurde, werden Module aktuell als in sich geschlossene und selbstorganisierende Einheiten verstanden [vgl. Meyer, 1997, S. 39]. Sie sollen über eine Selbstbeschreibungsfähigkeit verfügen, mit der sie ihre Funktionalität sowie die von ihnen benötigte Funktionalität explizit dokumentieren. Dadurch können Module in verschiedenen Anwendungskontexten zusammengesetzt werden und erlauben eine Überprüfung ihrer Komposition auf mögliche Fehler [vgl. Szyperski 2002, S. 3ff; Meyer 1997, S. 54ff]. Szyperski und Meyer argumentieren ferner, dass bei modularen Softwaresystemen mit autonomen Modulen die Wahrscheinlichkeit steigt, dass Änderungen nur lokale Auswirkungen haben, die nur ein oder einige wenige Module betreffen [Szyperski 2002, S. 139ff; Meyer 1997, S. 40ff]. Die Modularisierung begünstigt somit Änderbarkeit und Erweiterbarkeit eines Systems und verursacht weniger Seiteneffekte [vgl. Breivold et al. 2007; Riebisch und Bode 2009].

Der Umfang der vorgestellten Maßnahmen zur Verbesserung der Änderbarkeit und Erweiterbarkeit ist sehr projektspezifisch und variiert je nach Ausgangszustand bestehender Leitstandsysteme und ihrer Fähigkeit zur Evolution. Monolithische (nicht-modulare) Leitstandsysteme mit stark erodierter Architektur müssen ggf. aufwändig und invasiv überarbeitet werden, bevor sie ihre Evolutionsfähigkeit wiedererlangen und neue Leitstandfunktionalität aufnehmen können. Dagegen fallen die Reengineering-Maßnahmen in modularen Systemen, deren Architektur Softwareentwickler wiederkehrend an neue Anforderungen angepasst haben, deutlich geringer aus [vgl. Lilienthal, 2008].

6.1.4 Zusammenfassung

In langlebigen Leitstandsystemen finden Veränderungen permanent statt und sind nicht auf einen fixen Zeitpunkt im Lebenszyklus der Leitstandsysteme begrenzt [Ransom et al., 1998]. Die Leitstandmodernisierung ist somit ein kontinuierlicher Prozess, bei dem die Fähigkeit einer Software, strukturelle Änderungen wiederkehrend zu unterstützen, langfristig erhalten werden muss. In diesem Abschnitt wurden die von Rajlich und Bennett vorgestellten Software-Lebenszyklusphasen *Evolution* und *Servicing* differenziert und die Evolutionsphase für Unternehmen als die erstrebenswerte diagnostiziert [Rajlich und Bennett, 2000]. Zur Bewertung der Evolutionsfähigkeit von Softwaresystemen im Kontext der Leitstandmodernisierung wurden die von Breivold et al. vorgestellten Kriterien *Analysierbarkeit*, *Änderbarkeit* und *Erweiterbarkeit* näher untersucht und ausgewählte Reengineering-Maßnahmen vorgestellt, wie Legacy-Systeme diese Kriterien erreichen oder verbessern können [Breivold et al., 2007]. Sobald Softwaresysteme ihre Fähigkeit zur Evolution wiedererlangen, können sie aus ökonomischer und technischer Sicht einfacher modernisiert werden.

Als Fundament für die Modernisierung vorhandener Leitstandsysteme wurden in Kapitel 3 Anlagenmodelle eingeführt und in Kapitel 4 ein Modell-Kandidat auf Basis der OPC-UA-Technologie ausgewählt. Die Verbreitung der OPC-UA-Technologie in der Industrie wird federführend von der OPC-Foundation geleitet. Diese steht vor der Herausforderung, ihre neue OPC-UA-Technologie im Rahmen einer Softwareevolution in bestehende Leitstandsysteme einführen zu müssen, ohne diese massiv zu verändern. Dazu hat die OPC-Foundation mehrere Vorschläge erarbeitet, wie diese Einführung schrittweise verlaufen kann. Die Realisierung dieser Ansätze hat sich jedoch in ersten Pilotprojekten als problematisch erwiesen. Im folgenden Abschnitt werden diese Ansätze kurz vorgestellt und die damit verbundenen Schwierigkeiten herausgearbeitet, bevor Alternativen für eine Leitstandmodernisierung vorgestellt werden.

6.2 OPC-UA-Modernisierungsansätze

Leitstandsysteme werden heutzutage nur noch selten komplett neu entwickelt, sondern müssen sich meist in eine bestehende IT-Landschaft fügen. Die OPC-Foundation hat daher eine Strategie erarbeitet, die in drei Phasen neue OPC-UA-Konzepte erfolgreich in der Industrie etablieren soll [Mahnke et al. 2009, S. 293ff; Hannelius et al. 2008]:

- **1. Phase:** In der ersten Phase werden bestehende Leitstandsysteme gekapselt und mit neuer OPC-UA-Funktionalität ergänzt. Die Kapselung verhindert dabei weitreichende Modifikationen in vorhandenen Systemen.
- **2. Phase:** In der zweiten Phase kommen Integrationskonzepte zum Einsatz, die sukzessive mehr OPC-UA-Funktionalität in bestehende Leitstandsysteme einführen.
- **3. Phase:** In der letzten Phase werden gänzlich neue OPC-UA-Anwendungen entwickelt.

Im Folgenden werden Ansätze der ersten Phase zusammengefasst, da diese Phase der Softwareevolution zugeordnet werden kann und für Unternehmen den Einstieg in die OPC-UA-Technologie bereitet. Die Ansätze der ersten Phase adressieren die Leitstandarchitektur klassischer OPC-basierter Prozessleitsysteme. Diese wurden in Abschnitt 2.3 vorgestellt und sind nach einem Client/Server-Prinzip aufgebaut. Während beim klassischen OPC für den Zugriff auf Echtzeitdaten, historische Daten und Alarme mehrere dedizierte Server kontaktiert werden mussten, vereint OPC UA die verschiedenen Clients und Server, um den Datenzugriff zu vereinfachen [vgl. z. B. Gîrbea et al., 2010, S. 413]. Dies führt jedoch zu einem Architekturbruch zwischen der klassischen und neuen OPC-UA-Technologie. Um diesen Architekturbruch zu überwinden und neue OPC-UA-Konzepte mit wenigen Eingriffen in vorhandene Systeme einzuführen, schlägt die OPC-Foundation vier Ansätze vor: Wrapper, Proxies, Gateways und Adapter. Diese Ansätze werden im Folgenden vorgestellt und die damit verbundenen Schwierigkeiten herausgearbeitet.

6.2.1 Wrapper

Wrapper kapseln Daten und Programme des Altsystems in ihrer ursprünglichen Umgebung und bieten Schnittstellen, über die das neue System auf die Dienste des Altsystems zugreifen kann [Sneed et al., 2010, S. 12]. OPC-UA-Wrapper ermöglichen eine Interaktion zwischen einem OPC-UA-Client und klassischen Servern (Abbildung 6.3). Die Hauptaufgabe der Wrapper besteht darin, OPC-UA-Client-Anfragen auf die klassischen Servertechnologien und umgekehrt abzubilden. Der Vorteil dieser Lösung liegt darin, dass bestehende

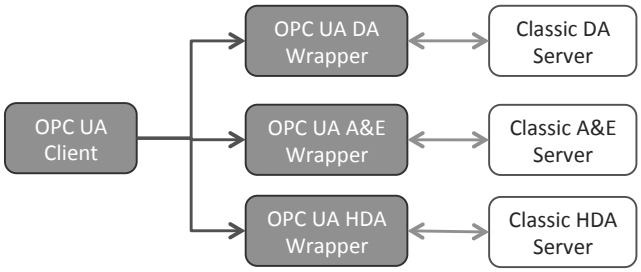


Abbildung 6.3: OPC UA Wrapper (Quelle: eigene Erstellung)

Server erhalten bleiben. Sie hat jedoch den Nachteil, dass klassische OPC-Server neue OPC-UA-Konzepte, insbesondere Anlagenmodelle und darauf aufbauende Funktionalität, nicht unterstützen. Dadurch sind OPC-UA-Clients, trotz des Integrationsaufwandes, ausschließlich auf die Funktionalität klassischer Server beschränkt [Hannelius et al., 2008]. Zusätzlich ist bei dieser Lösung unklar, ob bestehende und etablierte Clients abgelöst werden oder wie diese mit neuen OPC-UA-Clients interagieren sollen [vgl. Hannelius et al. 2008; Son und Yi 2010; Van Tan et al. 2009; Gîrbea et al. 2010]. Schließlich führt ein Wrapper eine Indirektion in den Kommunikationsverlauf hinzu, der zusätzliche Konvertierungszeit erfordert und potenziell neue Sicherheitslücken eröffnet [Lange et al., 2010, S. 375].

6.2.2 Proxies

Proxies ermöglichen klassischen Clients die Kommunikation mit einem OPC-UA-Server (Abbildung 6.4). Ihre Aufgabe besteht darin, Anfragen klassischer Clients auf OPC-UA-Technologien und umgekehrt abzubilden. Der Vorteil

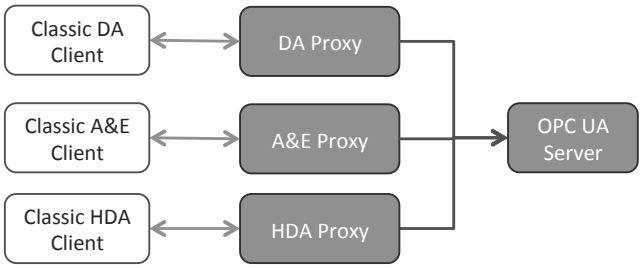


Abbildung 6.4: OPC-UAProxy (Quelle: eigene Erstellung)

dieser Lösung liegt darin, dass bestehende Clients erhalten bleiben. Sie hat jedoch den Nachteil, dass die klassischen Clients keine Kenntnis über neue OPC-UA-Konzepte haben und somit diese auch nicht nutzen können. Zusätzlich werden bei dieser Lösung bestehende, etablierte Server abgelöst, was mit erheblichen Risiken verbunden ist und die Anlagenverfügbarkeit beeinträchtigt [vgl. Hannelius et al. 2008; Van Tan et al. 2009; Son und Yi 2010; Gîrbea et al. 2010]. Auch Proxies führen eine Indirektion in den Kommunikationsverlauf zwischen Client und Server hinzu.

6.2.3 Gateways und Adapter

Gateways ermöglichen den Zugriff auf Echtzeitdaten, historische Daten sowie Alarme über eine einheitliche Schnittstelle, statt mehrere klassische Server zu kontaktieren [Gîrbea et al., 2010]. Adapter sind spezielle Gateways, die für jedes Unternehmen individuell entwickelt werden, um dadurch mehr OPC-UA-Funktionalität zugänglich zu machen und die Interoperabilität klassischer und neuer Ansätze zu erhöhen. Beide Ansätze exponieren serverseitige OPC-UA-Funktionalität und setzen einen Client voraus, der diese auch nutzen kann. Dies ist in klassischen OPC-Anwendungen nicht gegeben.

6.2.4 Zusammenfassung

Die vorgestellten Ansätze greifen entweder massiv in die bestehenden Systeme ein oder sie kapseln diese und führen Schnittstellen ein, die zwischen neuen und alten Systemen vermitteln. Diese Schnittstellen erlauben zwar neue Technologien in ein vorhandenes System einzuführen, dabei bleibt je-

doch die fachliche Funktionalität unverändert. Diese Vorgehensweise kann daher der Softwaremigration zugeordnet werden, deren Vorteile erst im Anschluss sichtbar werden, wenn fachliche Erweiterungen folgen [Sneed et al., 2010, S. 29ff]. Während für Anlagenbetreiber die Aufwände einer Migration direkt sichtbar sind, bleibt ihr Nutzen für die Arbeitssituationen der Operateure weitgehend verborgen. Anlagenbetreiber müssen daher auf das Potential neuer Technologien vertrauen, was mit hohen Risiken verbunden ist und die Einstiegshürde erhöht. Für Unternehmen bleibt daher die Frage offen, wie sie von neuen technologischen Entwicklungen profitieren und dabei ihre bereits getätigten Investitionen in bestehende Leitstandsysteme, Unternehmensdaten und Ausbildung ihrer Mitarbeiter schützen können. Da die vorgestellten Ansätze für das Vorhaben dieser Arbeit ungeeignet sind, werden Alternativen benötigt, die neue Leitstandfunktionalität mit wenigen Eingriffen in bestehende Systeme einführen und ihr Potential nutzen, um die Arbeitssituationen der Operateure zu verbessern.

6.3 Minimalanforderung an eine Leitstandmodernisierung

In dem vorangegangenen Abschnitt wurden aktuelle Modernisierungsansätze der OPC-Foundation vorgestellt und für den Ansatz dieser Arbeit als ungeeignet ermessen, da sie nur unzureichend bestehende Leitstandsysteme, Unternehmensdaten und Ausbildung der Mitarbeiter berücksichtigen. Folglich muss bei einer Leitstandmodernisierung nicht nur das vorhandene Leitstandsystem und die dazugehörige technische Infrastruktur, sondern auch das über mehrere Jahre angesammelte Wissen sowie Erfahrungen der Operateure einbezogen werden. Ransom et al. empfehlen bei der Modernisierung von Softwaresystemen nicht nur *technische*, sondern auch *wirtschaftliche* und *organisatorische* Faktoren in Unternehmen zu berücksichtigen [vgl. Ransom et al., 1998]. Diese Faktoren werden im Folgenden vorgestellt und auf die Leitstandmodernisierung übertragen:

- **Wirtschaftliche Faktoren:** Helfen in erster Linie, die Relevanz eines Systems für ein Unternehmen zu bestimmen und Business-kritische Anwendungen von Randsystemen zu separieren. Anschließend priorisieren die Unternehmen ihre Business-kritischen Systeme, stellen Kosten-Nutzen-Analysen für Änderungen oder Ablösung dieser Systeme auf und schätzen die daraus resultierenden Auswirkungen auf betriebliche Abläufe ab. Bei der Wirtschaftlichkeitsbetrachtung spielen auch monetär schwer erfassbare Faktoren eine entscheidende Rolle. Dazu gehören strategische Entscheidungen für z. B. Standard- oder Individualsoftware und Anbieter. Zudem müssen Unternehmen die aktuelle und langfristige Erweiterbarkeit und Anpassbarkeit ihrer Systeme an betriebliche Abläufe berücksichtigen [vgl. Schwarzer und Krcmar, 2004, S. 223ff].

- **Organisatorische Faktoren:** Sind nach Ransom et al. nicht einfach zu bestimmen, können jedoch erhebliche Auswirkungen auf die Entscheidung zum Erhalt oder Ablösung eines Legacy-Systems haben [Ransom et al., 1998, S. 5]. Dazu zählt beispielsweise der Reifegrad der IT-Prozesse und ihre mögliche Zertifizierung entsprechend eines in der Industrie anerkannten Standards wie [CMMI] oder [ISO 9000]. Weitere Einflussfaktoren sind die Ausbildung der Mitarbeiter, ihr Know-How im Bereich der verwendeten Systeme sowie ihre Verfügbarkeit für entsprechende Projekte. Schließlich spielt auch die Offenheit der Unternehmen für Veränderungen eine ausschlaggebende Rolle.

- **Technische Faktoren:** Können nach Sommerville weiter differenziert und in *umgebungsbedingte Faktoren* und *Softwarequalitätsfaktoren* unterteilt werden [Sommerville, 2001, S. 603f]. Umgebungsbedingte Faktoren umfassen beispielsweise Unterstützungssoftware wie Betriebssysteme, Hilfsprogramme oder die Hardwareplattform, auf der das Softwaresystem ausgeführt wird. Softwarequalitätsfaktoren umfassen nach Sommerville u.a. die Verständlichkeit des Systems und zugehörigen Quellcodes, Umfang der Dokumentation und die Konsistenz der im System verfügbaren Daten [vgl. Sommerville, 2001, S. 603f]. Häufig wiederkehrende und aufwändig zur korrigierende Hardwarefehler bzw. veraltete und nicht mehr zu beschaffende Hardware, Performanzpro-

bleme oder fehlende Schnittstellen zu externen Systemen können den wirtschaftlichen Betrieb einer Anwendungssoftware gefährden, selbst wenn diese von genügender Qualität ist [Sommerville, 2001, S. 600ff].

Die Bewertung dieser Faktoren kann nach Ransom et al. durch Expertenbefragungen oder für besonders kritische Anwendungen auch detailliert anhand von Metriken erfolgen [Ransom et al., 1998]. Die daraus abgeleiteten Ergebnisse ordnen die Autoren in einer Matrix ein (siehe Abbildung 6.5). Dabei vergleichen sie zunächst nur die technischen und Business-Faktoren, um einen initialen Eindruck von der Relevanz des untersuchten Systems zu erhalten. In

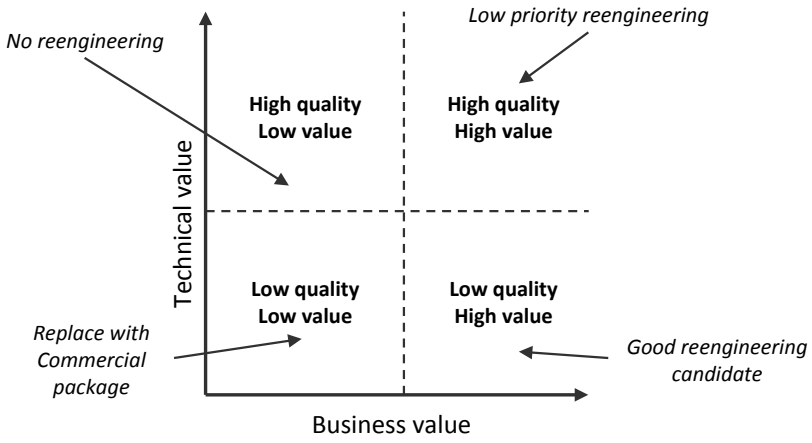


Abbildung 6.5: Bewertung von Legacy-Systemen anhand technischer und Business-Faktoren (Quelle: Ransom et al., 1998)

dieser Matrix sind Business-kritische Systeme mit niedriger Softwarequalität die primären Kandidaten für Reengineering-Maßnahmen und sollten in einer weitergehenden Analyse hinsichtlich organisatorischer Faktoren bewertet werden.

Wie eingangs dieses Kapitels beschrieben, sind Leitstandsysteme für Anlagenbetreiber überlebensnotwendig, da sie die gesamte technische Anlage und darin ablaufende Prozesse steuern und überwachen. Erodierte Leitstandsysteme können daher in den rechten unteren Quadranten der Abbildung 6.5 eingeordnet werden und sind probate Reengineering-Kandidaten. Die Bewertung von Leitstandsystemen hinsichtlich wirtschaftlicher, technischer und organisatorischer Faktoren ist sehr anspruchsvoll. So können beispielsweise in stark erodierten Leitstandsystemen Reengineering-Maßnahmen technisch

herausfordernd und umfangreich werden. In solchen Fällen kann eine Kosten-Nutzen-Analyse dafür ausschlaggebend sein, ein Leitstandsystem durch eine Neuentwicklung oder Standardsystem abzulösen. Dabei muss die Anlage meist für einen längeren Zeitraum stillgelegt und aufwändig rekonfiguriert werden. Aus einer gesamtwirtschaftlichen und organisatorischen Betrachtung kann dadurch die Modernisierung eines Leitstandsystems wiederum ökonomischer sein. Für Anlagenbetreiber entsteht daraus das Dilemma, die konkurrierenden Einflussfaktoren auf eine Leitstandmodernisierung in eine Balance zu bringen, in der sie von neuen technologischen Entwicklungen und Lösungen profitieren und gleichzeitig die Risiken bei ihrer Einführung reduzieren können. Die Bewertung einer Leitstandanwendung ist immer unternehmens- und projektspezifisch und muss konkret an dem zu untersuchenden Leitstandsystem erfolgen. An dieser Stelle sollen daher die Minimalanforderungen an eine Leitstandmodernisierung diskutiert werden:

- **Technische Anforderungen:** Im Idealfall sollten Leitstandsysteme vor einer Modernisierung die in Abschnitt 6.1 herausgearbeiteten Eigenschaften evolutionsfähiger Systeme aufweisen. Die Wiederherstellung der Evolutionsfähigkeit und Rückkehr in den Architekturkorridor ist jedoch nicht für alle Legacy-Leitstandsysteme aus den zuvor genannten technischen, wirtschaftlichen oder organisatorischen Gründen möglich. In diesem Fall sollten Legacy-Leitstandsysteme zumindest Schnittstellen bereitstellen, über die sie Dienste und Daten anbieten und mit anderen Systemen austauschen können (siehe dazu Abschnitt 8.3). Diese Schnittstellen können im Extremfall auch proprietär sein und sich beispielsweise Techniken wie Auslesen von Oberflächen (engl. screen scraping), Datenbank-Gateways oder Austausch von Dokumenten bedienen [vgl. z. B. Comella-Dorda et al., 2000].

- **Wirtschaftliche Anforderungen:** In einer gesamtwirtschaftlichen Betrachtung einer Kosten-Nutzen-Rechnung können, zu den Aufwänden für softwaretechnische Modernisierungs-Maßnahmen, auch solche Kosten hinzugerechnet werden, wie z. B. Produktionsausfälle, mögliche Abwanderung der Kunden und Lieferanten zu Konkurrenten, anfallende Gemeinkosten wie z. B. Löhne und Energiekosten in einer stillstehenden

Anlage oder auch Imageverluste. Diese Kosten können dazu beitragen, dass eine Leitstandmodernisierung trotz höherer Reengineering-Aufwände ökonomischer ist als eine Neuentwicklung oder der Wechsel zum Standardsystem. Zusammenfassend sollte der Umfang der Modernisierungs-Maßnahmen, aus einer gesamtwirtschaftlichen Betrachtung des Anlagenbetreibers, ökonomisch günstiger ausfallen und weniger Störungen betrieblicher Abläufe verursachen als zur Auswahl stehende Alternativen.

Leitstandsysteme, die diese Anforderungen erfüllen, bieten geeignete technische und wirtschaftliche Voraussetzungen, um mit vertretbarem Aufwand modernisiert zu werden. Auf der organisatorischen Ebene setzt dieses Vorhaben die Bereitschaft der Anlagenbetreiber und Operateure für Veränderungen voraus. In einer konservativen Leitstandwelt, in der Anlagenbetreiber und Operateure schon häufige technologische Veränderungen und die damit einhergehende Störungen des Anlagenbetriebs erfahren haben, spielt dieser Aspekt eine entscheidende Rolle und beeinflusst maßgeblich den Erfolg einer Leitstandmodernisierung. Die Offenheit der Anlagenbetreiber und Operateure für Veränderungen ist dabei kein rein organisatorischer Faktor, sondern umfasst auch wirtschaftliche und technische Aspekte, da diese sich gegenseitig beeinflussen. Zusammengenommen bilden diese Faktoren den fundamentalen Auslöser für das minimalinvasive Vorgehen dieser Arbeit. Aufgrund der hohen Relevanz dieses Vorgehens wird es im nächsten Abschnitt eingehender diskutiert.

6.4 Minimalinvasiver Ansatz

Anlagenbetreiber und Operateure sind in langlebigen Leitstandsystemen mit häufigen technologischen Veränderungen konfrontiert. Wie in Abschnitt 6.1 beschrieben, müssen sich Leitstandsysteme an wechselnde betriebliche Abläufe anpassen, um ihren Nutzen für die Umwelt zu erhalten. Mit jeder Veränderung muss die neue Technologie aufwändig auf die Anlage abgestimmt und die Anlage in der neuen technologischen Konstellation erprobt werden. Dabei muss die Verbindung zwischen Hard- und Software überprüft, das Personal

geschult und Prozessabweichungen korrigiert werden [Weber, 2006, S. 333ff]. In dieser, meist sehr langwierigen Phase, treten vermehrt Störungen auf, die den Anlagenbetrieb gefährden und die Produktion beeinträchtigen [vgl. Strasser et al. 2005; Sauer und Ebel 2007b]. Ein technologiezentriertes Vorgehen in dieser Phase bewirkt, dass während und nach einer Leitstandmodernisierung Operateure und Ingenieure mit neuen Systemen oft überfordert sind. Die eingeführten Neuheiten zwingen Operateure und Ingenieure meist, ihre bewährten Arbeitsabläufe zu verändern und den Umgang mit neuen Werkzeugen zu erlernen. Dies führt gerade in der Einarbeitungsphase zu weiteren Fehlern, die den Anlagenbetrieb gefährden. Für Anlagenbetreiber, Operateure und Ingenieure stellt sich daher die Frage, wie sie vorhandene Leitstandsysteme ohne massive Eingriffe modernisieren und dabei den Anlagenbetrieb aufrecht erhalten können.

Betrachtet man vergleichbare Probleme jenseits der Automatisierungs- und Softwaretechnik, so gibt es kaum größeren Bedarf, Eingriffe mit möglichst wenigen negativen Auswirkungen und Risiken vorzunehmen, als in der Medizintechnik, in der sich die sog. *minimalinvasive Chirurgie* etabliert hat. Dahinter verbergen sich eine Reihe von Verfahren, deren gemeinsames Ziel darin besteht „[. . .] möglichst wenig traumatisch bei diagnostischen oder therapeutischen Eingriffen vorzugehen, d. h. Haut und Weichteile möglichst wenig zu verletzen“ [Perleth, 2007]. Patienten haben dadurch weniger Schmerzen, ihre Heilung verläuft schneller und sie haben eine kürzere Aufenthaltsdauer im Krankenhaus. Das hat medizinische und ökonomische Vorteile. Der minimalinvasive Ansatz wird in der vorliegenden Arbeit auf die Modernisierung bestehender Leitstandsysteme übertragen. In diesem Kontext bewirkt ein minimalinvasives Vorgehen, dass der Anlagenbetrieb während einer Leitstandmodernisierung minimal beeinträchtigt wird. Die Modernisierung soll dabei schrittweise erfolgen, mit wenigen und lokal begrenzten Eingriffen, die Kollateralschäden minimieren. Diese Aspekte lassen sich auf die in Abschnitt 6.3 eingeführten wirtschaftlichen, technischen und organisatorischen Faktoren übertragen und aus ihrer jeweiligen Sicht diskutieren. Vor dem Hintergrund einer anwenderorientierten Vorgehensweise werden von den zahlreichen organisatorischen Faktoren vorwiegend Faktoren mit Einfluss auf Anwender

und ihre Arbeit betrachtet und im Folgenden aus einer anwenderorientierten Sicht diskutiert:

- **Anwenderorientierte Sicht:** Aus einer anwenderorientierten Sicht zeichnet sich eine minimalinvasive Leitstandmodernisierung dadurch aus, dass sie bewährte Arbeitsabläufe der Operateure möglichst wenig beeinträchtigt und Operateure mit der Fülle neuer Leitstandfunktionen nicht überfordert [vgl. Shneiderman, 2003]. Vielmehr leitet eine minimalinvasive Vorgehensweise Operateuren an, alternative und auf konkrete Situationen und ihre Bedürfnisse besser angepasste Werkzeuge zu erproben [vgl. Züllighoven et al., 1998, S. 261]. Gleichzeitig bietet eine minimalinvasive Vorgehensweise Operateuren stets die Möglichkeit, zu Altbewährtem zurückzukehren und neue Ansätze optional, in z. B. weniger kritischen Situationen zu erproben.

- **Technische Sicht:** Aus technischer Sicht hilft ein minimalinvasives Vorgehen, vorhandene Systeme weitgehend zu erhalten und durch wenige Eingriffe und Seiteneffekte mit neuer Funktionalität zu ergänzen. Die neue Funktionalität soll dabei nicht als Insellösung hinzugefügt, sondern über Schnittstellen mit der vorhandenen Leitstandfunktionalität lose verknüpft werden. Dadurch soll sich die neue und vorhandene Funktionalität gegenseitig ergänzen [vgl. Sullivan und Notkin, 1992, S. 230]. Die minimalen Eingriffe in vorhandene Systeme helfen, Risiken während einer Leitstandmodernisierung zu reduzieren und die Anlagenverfügbarkeit zu erhöhen. Ein minimalinvasives Vorgehen fokussiert nicht nur auf die kurzfristige Verbesserung der Leitstandfunktionalität, sondern berücksichtigt die Langlebigkeit der Leitstandsysteme und fördert ihre langfristige Evolutionsfähigkeit [vgl. Rajlich und Bennett, 2000].

- **Ökonomische Sicht:** Ein minimalinvasives Vorgehen hilft Anlagenbetreibern, ihre Softwaresysteme und die damit verbundene Produktionsmaschinerie nach einer Modernisierung schneller wieder in Gang zu setzen und mögliche Störungen lokal auf wenige Produktionsprozesse zu begrenzen. Dadurch können Anlagenbetreiber schneller auf Änderungen der Marktanforderungen reagieren und nach einer Modernisierung zügiger in den Wettbewerb zurückkehren. Diese Fähigkeit bezeichnen

Wiendahl et al. sowie Sauer und Ebel als strategischen Wettbewerbsvorteil von Fabriken und Produktionssystemen der Zukunft [Wiendahl et al. 2007; Sauer und Ebel 2007b].

Der minimalinvasive Ansatz dieser Arbeit strebt danach, während einer Leitstandmodernisierung möglichst wenige Nebenwirkungen zu verursachen, so dass der Patient *Anlagenbetreiber* schneller genesen und in den Wettbewerb zurückkehren kann. Dieses Ziel soll auf einer anwenderorientierten Ebene durch minimale Eingriffe in bewährte Handlungsmuster und Arbeitsabläufe der Operateure erreicht werden. Auf der technischen Ebene sollen minimale Anpassungen und Ergänzungen vorhandener Systeme helfen, Seiteneffekte und Risiken bei einer Leitstandmodernisierung zu reduzieren. Diese Ebenen werden im weiteren Verlauf der Arbeit eingehender untersucht und dazu passende Lösungsansätze herausgearbeitet.

6.5 Zusammenfassung

In diesem Kapitel wurde die Langlebigkeit technischer Anlagen und darin ablaufender Softwaresysteme diskutiert. Langlebige Systeme verfallen häufig zu Legacy-Systemen, wenn sie kontinuierlich geändert und erweitert werden, ohne entsprechende Anpassungen ihrer Architektur. In diesem Kapitel wurden Ansätze zum Umgang mit Legacy-Systemen differenziert und diese Arbeit der Softwareevolution zugeordnet. Statt Legacy-Systeme wiederkehrend abzulösen, hat die Softwareevolution eine langfristige und inkrementelle Verbesserung vorhandener Systeme zum Ziel. Die Leitstandmodernisierung ist somit ein kontinuierlicher Prozess und nicht auf einen fixen Zeitpunkt im Lebenszyklus einer Software beschränkt. Dieser Prozess ist technisch und ökonomisch einfacher zu realisieren, wenn sich Leitstandsysteme in dem Architekturkorridor der Abbildung 6.1 befinden. Für die Rückführung vorhandener Legacy-Leitstandsysteme in den Architekturkorridor wurden in diesem Kapitel die von Breivold et al. formulierten Eigenschaften evolutionsfähiger Systeme vorgestellt und ausgewählte Reengineering-Maßnahmen skizziert, wie Leitstandsysteme ihre Fähigkeit zur Evolution wiedererlangen können [Breivold et al., 2007]. Die Umsetzung dieser Maßnahmen ist immer projekts-

pezifisch und variiert je nach Ausgangszustand einer Software [vgl. Lilienthal, 2008, S. 169ff]. Die Eigenschaften evolutionsfähiger Softwaresysteme werden in dieser Arbeit herangezogen, um die Integrität der Leitstandarchitektur während und nach einer Modernisierung zu wahren und die langfristige Evolutionsfähigkeit der Leitstandsysteme zu erhöhen.

Als Lösungsansatz für eine Leitstandmodernisierung wurden in diesem Kapitel aktuelle Modernisierungsansätze aus dem Umfeld der OPC-UA-Technologie vorgestellt, ihre Vorgehensweise der Softwaremigration zugeordnet und die damit verbundenen Schwierigkeiten aufgezeigt. Diese wurden u.a. darauf zurückgeführt, dass aktuelle Modernisierungsansätze nur unzureichend die Investitionen der Anlagenbetreiber in bestehende Leitstandsysteme, Unternehmensdaten und Ausbildung ihrer Mitarbeiter berücksichtigen. Folglich wurden in diesem Kapitel die von Ransom et al. vorgestellten technischen, ökonomischen und organisatorischen Faktoren zur Bewertung von Softwaresystemen herangezogen und auf die Leitstandmodernisierung übertragen [Ransom et al., 1998]. Leitstandsysteme sind aus ökonomischer Sicht probate Modernisierungs-Kandidaten, da ihre Ablösung aufwändig und mit vielen Risiken verbunden ist. Auf der technischen Ebene sollten Leitstandsysteme evolutionsfähig sein, um strukturelle Änderungen zu unterstützen. Die Wiederherstellung der Evolutionsfähigkeit in Legacy-Systemen ist jedoch nicht immer möglich. In diesem Fall sollten Legacy-Leitstandsysteme zumindest Schnittstellen bereitstellen, über die sie ihre Dienste und Daten mit anderen Systemen austauschen können. Die Modernisierung vorhandener Systeme muss schließlich die begrenzte Offenheit der Anlagenbetreiber und Operateure für Veränderungen berücksichtigen und bewährte Handlungsmuster und Arbeitsabläufe weitgehend erhalten. Dazu wurde in diesem Kapitel das minimalinvasive Vorgehen eingeführt und motiviert. Dieses Vorgehen hilft, während und nach einer Leitstandmodernisierung bei dem Patienten *Anlagenbetreiber* möglichst wenige Nebenwirkungen zu verursachen und den Anlagenbetrieb minimal zu beeinträchtigen. Diese Aspekte beantworten zusammengenommen die zweite Forschungsfrage, welche softwaretechnischen, ökonomischen und organisatorischen Anforderungen Anlagenbetreiber erfüllen müssen, um eine anwenderorientierte Leitstandmodernisierung zu ermöglichen.

In den folgenden Kapiteln werden anwenderorientierte und softwaretechnische Einflussfaktoren auf eine Leitstandmodernisierung eingehender untersucht und methodische sowie softwaretechnische Konzepte vorgestellt, die ein minimalinvasives Vorgehen ermöglichen.

Kapitel 7

Minimalinvasive Leitstandmodernisierung aus anwenderorientierter Sicht

Die Einführung neuer Funktionalität in bestehende Systeme impliziert meist eine Veränderung routinierter Arbeitsabläufe der Anwender, die infolgedessen ihre Arbeit reorganisieren und den Umgang mit neuen Werkzeugen erlernen müssen [vgl. Floyd et al., 1989, S. 51]. Unternehmen verknüpfen mit der neuen und verbesserten Funktionalität häufig die Erwartung, Arbeitsabläufe zu optimieren und die Produktivität ihrer Mitarbeiter zu steigern. Diese Erwartungen bleiben nach Davis wirkungslos, wenn Anwender die neue Funktionalität ablehnen [Davis, 1993, S. 475]. Die Ursachen dafür sind vielfältig und liegen nach Shneiderman beispielsweise darin begründet, dass Hersteller neuer Technologien in erster Linie mit ihrer Funktionsvielfalt werben und die Erlernbarkeit neuer Technologien in den Hintergrund stellen [Shneiderman, 2003]. Demeyer et al. begründen die Ablehnung der Anwender damit, dass diese neuen Konzepten prinzipiell skeptisch gegenüberstehen und wenig Vertrauen für neue Funktionalität aufbringen [vgl. Demeyer et al., 2008, S. 189].

Gerade in technischen Anlagen, die über mehrere Jahrzehnte betrieben werden, kristallisieren sich bei Operateuren bewährte Handlungsmuster heraus,

bei denen selbst kleinste Veränderungen kritisch aufgenommen werden. Mit jeder Veränderung müssen Operateure meist ihre Arbeit reorganisieren und den Umgang mit neuer Leitstandfunktionalität erlernen. Dies erhöht in der Einarbeitungsphase das Risiko für Anlagenausfälle, wenn potenziell unzuverlässige Software eingeführt und der Lernprozess der Operateure mit neuer Leitstandfunktionalität unzureichend berücksichtigt wird [vgl. Carroll und Carrithers, 1984, S. 800]. Folglich suchen Anlagenbetreiber nach einer Möglichkeit, ihre bestehenden Systeme sanft zu modernisieren, um Anlagenausfälle durch Fehlbedienung der Operateure zu reduzieren. In den folgenden Abschnitten werden Methoden der Softwaretechnik herangezogen, um für diese Zielsetzung passende Konzepte zu erarbeiten.

7.1 Evolutionäre Softwareentwicklung

Technische Anlagen verändern sich, „[. . .] weil sich Änderungen an den Produkten ergeben, Kapazitäten aufgrund schwankender Bedarfe neu justiert werden müssen oder rationellere Fertigungstechnologien eingesetzt werden“ [Sauer und Ebel, 2007b, S. 331]. Für Anlagenbetreiber folgt daraus die Notwendigkeit, ihre Produktionsprozesse kontinuierlich anzupassen. In Kapitel 6 wurde gezeigt, dass Veränderungen im Lebenszyklus technischer Anlagen nicht auf einen fixen Zeitpunkt begrenzt sind, sondern permanent stattfinden. Dies erfordert eine Methodik, die eine kontinuierliche und kleinschrittige Anpassung technischer Anlagen ermöglicht, statt mit jeder Änderung große Teile der Anlage und darin ablaufende Softwaresysteme zu ersetzen.

Zu Beginn der Softwareentwicklung waren überwiegend lineare, phasenorientierte Vorgehensmodelle verbreitet wie z. B. das Wasserfallmodell, als ihrem bekanntesten Vertreter. Lineare Vorgehensmodelle betrachten die Softwareentwicklung als eine zeitliche Folge von in sich geschlossenen Phasen, deren Ergebnis eine einsatzfähige Software ist. Rückgriffe auf vorangegangene Phasen waren zwar erlaubt, hatten aber den Charakter von Fehlern, die bei optimalem Projektverlauf nicht nötig wären [vgl. Züllighoven et al., 1998, S. 1100]. Bei linearen Vorgehensmodellen haben Anwender nur wenig Einfluss auf das zu entwickelnde Softwareprodukt und die fehlenden

Rücksprünge in vorangegangene Phasen ignorieren den Lernprozess aller Beteiligten [vgl. Züllighoven et al., 1998, S. 1102]. Dadurch entsteht am Ende des Projektes eine potenziell unzuverlässige Software. Diese wird meist in einem sog. *Big-Bang* eingeführt und stellt Anwender vor vollendete Tatsachen, ohne viele Optionen, das Resultat noch zu beeinflussen. Phasenorientierte Vorgehensmodelle gelten heute als überholt, da sie nur unzureichend mit Veränderungen der Organisationen umgehen und starre Architekturen hervorbringen. Diese Architekturen berücksichtigen lediglich initiale Anforderungen an Software und unterstützen kaum ihren Lebenszyklus nach der Auslieferung [vgl. Rausch, 2000].

Heute favorisieren immer mehr Entwickler evolutionäre Vorgehensmodelle. Diese heben den strikt linearen Ablauf auf und ersetzen ihn durch ein iterativ-inkrementelles Vorgehen. Dabei werden Anwender schrittweise und früh an neue Funktionalität herangeführt und können durch ihr Feedback das Resultat entlang des gesamten Entwicklungsprozesses beeinflussen [vgl. z. B. Gilb 1985; Züllighoven et al. 1998; Beck und Andres 2004]. Evolutionäre Vorgehensmodelle betrachten die Softwareentwicklung als einen Lernprozess, bei dem Kommunikation und Feedback als ein wichtiger Bestandteil der Entwicklung betrachtet wird. Dieser Lernprozess wird nach Floyd und Züllighoven et al. durch die folgenden Vorgehensweisen gefördert [Floyd 1984; Züllighoven et al. 1998]:

- **Inkrementelle Entwicklung:** Dabei werden komplexe Probleme in kleinen Schritten angegangen. Bei jedem Schritt sammeln Entwickler und Anwender durch ihre Zusammenarbeit Erfahrungen und lassen diese in die nächsten Schritte einfließen.

- **Zyklisches Vorgehen:** Statt alle Anforderung zu Beginn eines Projektes aufzunehmen und für den gesamten Entwicklungsprozess einzufrieren, werden bei diesem Vorgehen die Softwareentwicklungsphasen Design, Implementation und Evaluation wiederkehrend durchlaufen, um eine Software an die sich stetig ändernde Umwelt anzupassen.

Evolutionäre Vorgehensmodelle basieren nach Floyd auf zwei grundlegenden Erfahrungen [Floyd, 1984]:

- Organisationen verändern sich über die Zeit und mit ihnen verändern sich auch die Anforderungen an die Software.
- Sobald Anwender Software einsetzen, entstehen durch ihren Gebrauch neue Anforderungen.

Die evolutionäre Vorgehensweise bedarf einer klaren Zielsetzung, um die Menge der potentiellen Änderungen aus einer anwendungsfachlichen Sicht zu segmentieren und ihre Umsetzung in einem für Unternehmen ökonomisch sinnvollen Rahmen zu ermöglichen [vgl. Züllighoven et al., 1998, S. 1152]. Im WAM-Ansatz wird diese Vorgehensweise durch das Konzept *Kernsystem und Spezialsysteme* adressiert. Ein Kernsystem sollte nach WAM einen wichtigen Beitrag zur Erreichung des ersten Teilziels eines Softwaresystems leisten. Es sollte dazu die wesentlichen Aufgaben in einer Abteilung unterstützen und einen Satz an elementaren Dienstleistungen als Komponenten zur Verfügung stellen [vgl. Züllighoven et al., 1998, S. 1152f]. Vom Kernsystem lassen sich Spezialsysteme unterscheiden, die weitgehend eigenständige Aufgaben unterstützen. Sie können relativ lose mit dem Kernsystem verbunden werden. Die zeitliche Reihenfolge bei ihrer Einführung kann von Anwendern bestimmt werden. „Grundlage für die Festlegung des Kernsystems und möglicher Spezialsysteme ist ein Verständnis wesentlicher Aufgaben und Arbeitsteilungen im Anwendungsbereich, die sich auf eigenständige Komponenten abbilden lassen“ [Züllighoven et al., 1998, S. 1153].

In Kapitel 3 wurde das Anlagenmodell als Grundlage für darauf aufbauende Leitstandwerkzeuge identifiziert. Damit erfüllt das Anlagenmodell prinzipiell die Eigenschaften eines Kernsystems. Auf der fachlichen Ebene bieten Anlagenmodelle Operateuren u.a. die Möglichkeit, komplexe technische Zusammenhänge einer Anlage strukturiert zu erkunden und Anlagenteile zu suchen (siehe Abschnitt 3.4). Dies ist zwar eine hilfreiche, doch keine wesentliche Aufgabe der Operateure. Die Vorteile einer (Anlagen-)modellbasierten Leitstandsoftware entstehen vornehmlich durch darauf aufbauende Funktionalität wie z. B. Alarmierung (siehe Abschnitt 5). Dadurch erfüllt das Anlagenmodell zwar in der logischen Reihenfolge der Softwareentwicklung die Eigenschaften eines Kernsystems, jedoch nicht zwangsläufig in der Reihenfolge primär anzugehender Leitstandprobleme aus Sicht der Operateure.

In Kapitel 3 wurde herausgearbeitet, dass jegliche Modernisierungsmaßnahmen in Leitstandsystemen ohne ein zugrundeliegendes Anlagenmodell mit hohen Aufwänden verbunden und teilweise gänzlich ausgeschlossen sind. In dieser Arbeit wird daher das Anlagenmodell in der logischen Reihenfolge der Softwareentwicklung eingeführt und als Kernsystem für eine Leitstandmodernisierung aufgefasst. Auf dieser Basis können Softwareentwickler weitere unabhängige Spezialsysteme für die elementaren Leitstandaufgaben Datenzugriff, Alarmierung und historischen Datenzugriff realisieren. Die Einführungsreihenfolge dieser Spezialsysteme kann dabei durch unternehmensspezifische Faktoren wie z. B. Dringlichkeit, Problemrelevanz und Wünsche der Operateure beeinflusst werden. Dies ist konform zu dem Vorschlag von Demeyer et al. sowie Beck und Andres: „Start working on the aspects which are most valuable to your customer“ [Demeyer et al. 2008, S. 29; Beck und Andres 2004].

Die evolutionäre Vorgehensweise in Kombination mit Kernsystem und Spezialsystemen gibt Aufschluss darüber, wie vorhandene Softwaresysteme kontinuierlich angepasst und mit neuer Leitstandfunktionalität ergänzt werden können. Damit ist jedoch die Eingangs dieses Kapitels formulierte Fragestellung noch nicht ausreichend geklärt, wie die Einführung neuer Leitstandfunktionalität mit möglichst wenigen Veränderungen der Arbeitsweise der Operateure realisiert und ihre Skepsis für Neuerungen überwunden werden kann. Diese Aspekte werden im nächsten Abschnitt diskutiert.

7.2 Minimalinvasive Softwareerweiterung

Nach Davis wird die Akzeptanz der Anwender für neue Softwaresysteme und Technologien primär durch die beiden Merkmale *Funktionalität* (engl. Functionality) sowie *Gebrauchstauglichkeit* (engl. Usability) beeinflusst. Die Norm [ISO/IEC 9126] definiert Funktionalität und Gebrauchstauglichkeit wie folgt:

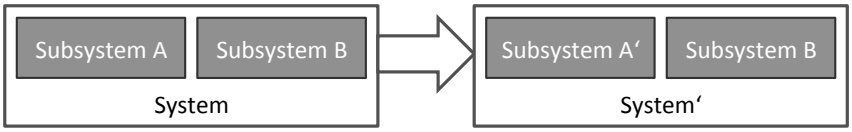
Begriff 7.1: Functionality
The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

Begriff 7.2: Usability
The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.

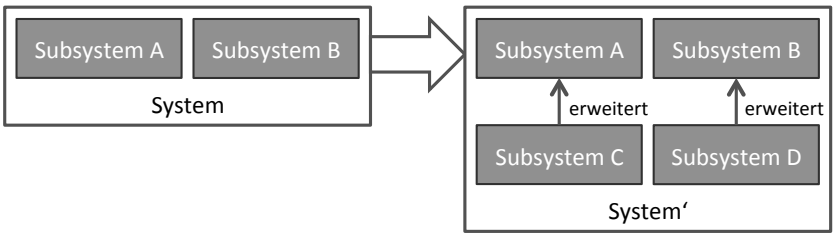
Demeyer et al. betonen, dass Anwender neben hoher Funktionalität und Gebrauchstauglichkeit zusätzlich verlangen, dass ihre bewährten Arbeitsabläufe, trotz vieler hilfreicher Neuerungen, so wenig wie möglich gestört werden:

> *„Users want improved functionality without disrupting their established work patterns“ [Demeyer et al., 2008, S. 23].*

Bei der Einführung neuer und gebrauchstauglicher Leitstandfunktionalität besteht die Gefahr, Operateure mit Neuerungen und ihrer Funktionsvielfalt zu überfordern. Das kann zu einer höheren Fehlbedienung und Ausfällen der Anlage führen. Jegliche Bemühungen, diese Risiken zu minimieren, führen potenziell dazu, dass Operateure Neuerungen kaum wahrnehmen und nur wenige bis keine Verbesserungen für ihre Arbeitssituationen erfahren. Ein minimalinvasives Vorgehen aus anwenderorientierter Sicht hilft, dieses Dilemma zu überwinden. Es strebt danach, neue Leitstandfunktionalität mit hoher Gebrauchstauglichkeit einzuführen und dabei die gewohnten Arbeitsabläufe der Operateure möglichst wenig zu beeinträchtigen. Für diese Zielsetzung werden im Folgenden, in Anlehnung an Davis und Fildebrandt, die Konzepte *Softwareredefinition* und *Softwareerweiterung* unterschieden [Davis 1993, S. 477; Fildebrandt 2012, S. 4f]. Dazu wird ein Softwaresystem als eine Komposition von Subsystemen aufgefasst. Diese Zusammenhänge sind in der Abbildung 7.1 illustriert.



(a) Softwareredefinition



(b) Softwareerweiterung

Abbildung 7.1: Softwareredefinition vs. Softwareerweiterung (Quelle: eigene Erstellung in Anlehnung an Fildebrandt, 2012, S. 4)

Begriff 7.3: Softwareredefinition
Bei einer Softwareredefinition wird neue Funktionalität durch Anpassung bestehender Subsysteme eingeführt, wodurch ihr bisheriges Verhalten und ihre Darstellung verändert werden.

Begriff 7.4: Softwareerweiterung
Bei einer Softwareerweiterung wird neue Funktionalität als zusätzliche Subsysteme eingeführt, ohne jedoch das Verhalten sowie die Darstellung der ursprünglichen Subsysteme gravierend zu verändern.

Eine *Softwareredefinition* bietet Entwicklern die Möglichkeit, das Verhalten sowie die Darstellung neuer und vorhandener Funktionalität aufeinander abzustimmen und Anwendern ein homogenes Bild der veränderten Software zu präsentieren. Dabei müssen jedoch bestehende Systeme meist umfassend und unwiderruflich verändert werden. Diese Veränderungen sind mit Risiken verbunden und zwingen Anwender häufig, neue Konzepte zu erlernen und ihre gewohnten Arbeitsabläufe anzupassen. Als Beispiel für eine Softwareredefinition kann die Umstellung Microsoft's Word 2003 auf Word 2007 angeführt werden. Bei dieser hat Microsoft sowohl die Funktionalität als auch

die gesamte Handhabung und Darstellung der Werkzeuge umgestellt, was kontroverse Debatten bei Anwendern und Usability-Experten ausgelöst hat [vgl. z. B. Theng et al. 2010; Colazzo et al. 2008].

Eine *Softwareerweiterung* ermöglicht, neue Funktionalität ohne Änderungen der vorhandenen Subsysteme einzuführen, so dass Anwender lediglich den Umgang mit der zusätzlichen Funktionalität erlernen müssen. Eine Softwareerweiterung hat jedoch den Nachteil, dass neue und alte Funktionalität meist unterschiedliche Schnittstellen und Modelle implementieren. Sie erfordern eine unterschiedliche Handhabung und Präsentation und müssen parallel gepflegt werden. Beispiele für eine Softwareerweiterung bietet die Eclipse Entwicklungsumgebung. Hier können Entwickler neue Werkzeuge wie z. B. CheckStyle[1] oder PMD[2] installieren, die den Quellcode auf Regelverstöße überprüfen und diese visualisieren. Diese Erweiterungen existieren neben den bestehenden Werkzeugen, wie z. B. dem Quellcode-Editor, und interagieren mit ihnen, ohne jedoch deren ursprüngliches Verhalten oder Darstellung zu verändern.

In Interviews und Workshops mit DESY-Operateuren zur Leitstandmodernisierung wurde deutlich, dass sie, aufgrund ihrer Erfahrungen mit vergangenen Technologiewechseln und oft ausbleibenden Verbesserungen für ihre Arbeitssituationen, Änderungen in ihren Softwaresystemen nur begrenzt tolerierten. Vielmehr favorisierten sie Konzepte, bei denen ihre gewohnten Arbeitsweisen und Softwarewerkzeuge weitgehend erhalten und lediglich an ausgewählten Stellen ergänzt werden. Dies beschränkt den Lernaufwand für Neuerungen auf die zusätzliche Funktionalität und reduziert erheblich das Risiko für Fehlbedienung in der Einarbeitungsphase. Aufgrund dieser Ergebnisse wird in dieser Arbeit für die Einführung neuer Leitstandfunktionalität das Konzept der *Softwareerweiterung* verwendet. Dieses Konzept wird mit Fokus auf das Zusammenspiel zwischen neuer und bestehender Funktionalität im nächsten Abschnitt anhand einer Entwurfsmetapher verdeutlicht, die gleichzeitig die Herangehensweise dieser Arbeit im Rahmen der Softwareerweiterung skizziert.

[1] http://eclipse-cs.sourceforge.net/
[2] http://pmd.sourceforge.net/

7.3 Entwurfsmetapher *Modernisierung der Altstadt*

Metaphern im allgemeinen sind ein „[...] sprachliches Ausdrucksmittel, bei dem ein bildhafter Ausdruck (z. B. Müll) aus einem Kontext (Haushalt) genommen und an die Stelle des eigentlichen Ausdrucks (ungenutzter Speicherplatz) in einem anderen Zusammenhang gestellt wird“ [Züllighoven et al., 1998, S. 164]. In der Softwareentwicklung werden Metaphern zur Gestaltung von Softwaresystemen verwendet. Züllighoven et al. sprechen daher auch von Entwurfsmetaphern und verstehen darunter [Züllighoven et al., 1998, S. 165]:

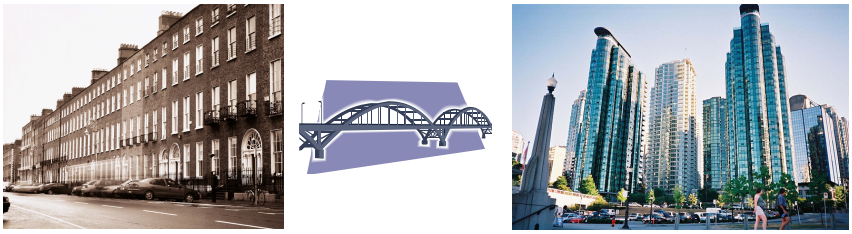
BEGRIFF 7.5: ENTWURFSMETAPHER
[...]

Eine Entwurfsmetapher dient der Gestaltung von Softwaresystemen, indem sie Handhabung und Funktionalität für die Beteiligten verständlicher macht.

Eine Entwurfsmetapher hat im WAM-Ansatz immer auch eine technisch konstruktive Interpretation in Form von Konstruktionsanleitungen und Entwurfsmustern.

In dieser Arbeit soll eine Entwurfsmetapher dabei helfen, die minimalinvasive Einführung neuer Technologien und Leitstandfunktionalität im Rahmen einer Softwareerweiterung verständlicher zu machen. Als Basis hierfür soll eine verwandte Entwurfsmetapher von Keller aus dem Bereich der Softwaremigration dienen: *„The Bridge to the New Town“* [Keller, 2000]. Dort betrachtet Keller bestehende, in die Jahre gekommene Systeme, als eine alte Stadt (engl. Legacy-Town), neben der eine neue errichtet wird. Die neue Stadt symbolisiert eine moderne technologische Umgebung mit verbesserten Werkzeugen und Konzepten für Anwender. Damit die Migration möglichst risikoarm verläuft, schlägt Keller vor, Brücken zwischen den alten und neuen Gebäuden zu errichten, die einen inkrementellen Umzug der Stadtbewohner (der Daten) ermöglichen. Dieser Ansatz verfolgt langfristig das Ziel, die alte Stadt sowie die Brücken zur neuen Stadt abzureißen, so dass langfristig allein die neue Stadt bestehen bleibt. In dieser Arbeit wird jedoch das Ziel verfolgt, bestehen-

de Systeme zu erhalten und zu modernisieren. Aus diesem Grund wird die Entwurfsmetapher von Keller modifiziert hin zu *Modernisierung der Altstadt* (Abbildung 7.2).



(a) The Bridge to the New Town

(b) Modernisierung der Altstadt

Abbildung 7.2: Modernisierung bestehender Städte durch Übertragung des Musters von Keller, 2000 (Quelle: (a) eigene Erstellung (b) Foto: Heiner Leiska)

Die grundlegende Idee dieses Ansatzes besteht darin, das typische Ortsbild der Altstadt und der prägenden historischen Gebäude weitgehend zu erhalten, diese behutsam zu sanieren und an ausgewählten Stellen zu modernisieren. Dazu gehört eine eingehende Prüfung der vorhandenen Bausubstanz, um möglichst viele erhaltenswerte Gebäude zu identifizieren. Die nicht erhaltenswerten Gebäude erfordern Maßnahmen, wie z. B. Abriss mit anschließendem Neubau. Im Sinne der evolutionären Softwareentwicklung aus Abschnitt 7.1 sollen die Einwohner der Altstadt bei der Sanierung und Modernisierung der Gebäude mitwirken und diese entsprechend ihren Bedürfnissen mitgestalten. Die Bewohner der Altstadt sollen zudem die Möglichkeit haben, die überarbeiteten Gebäude nach ihrer Fertigstellung zunächst zu begehen und zu erkunden, ohne sie zwangsläufig zu beziehen. Mit diesen Konzepten kann die bestehende Altstadt kontinuierlich und in kleinen Schritten entsprechend den

Wünschen der Bewohner modernisiert werden, ohne sie komplett abzureißen oder massiv zu verändern.

Abbildung 7.2 (b) verdeutlicht dieses Vorgehen am Beispiel des Hauptgebäudes der Technischen Universität Hamburg-Harburg. Die Flügelbauten des Gebäudes wurden im zweiten Weltkrieg durch Bomben beschädigt. Sie wurden daher teilweise abgerissen und durch moderne Gebäude ersetzt. Die erhaltenswerte Bausubstanz des Mittelbaus wurde hingegen grundsaniert. Dazu gehören eine aufwändige Innendämmung, Erneuerung des Dachstuhls und Austausch der Geschossdecken durch Betondecken. Die Modernisierung des Hauptgebäudes wurde durch Studierende mit finanziert und auch maßgeblich von ihnen geplant [TU Hamburg-Harburg, 2012].

Übertragen auf Leitstandsoftware, werden bei diesem Ansatz vorhandene Leitstandwerkzeuge erhalten und an ausgewählten Stellen im Rahmen der Softwareerweiterung aus Abschnitt 7.2 durch moderne Werkzeuge ergänzt. Das Beispiel der *Alarmierung* (Kapitel 5) verdeutlicht diesen Ansatz. Bei dem überarbeiteten Alarm-Konzept können Operateure Alarme sowohl in der bewährten Alarmtabelle als auch in einem neuen, modellbasierten und hierarchisch organisierten Werkzeug, dem *Alarmbaum*, betrachten. Dieser hilft Operateuren, insbesondere bei Alarmfluten, die Übersicht zu behalten und adressiert damit gezielt die Nachteile der bestehenden, wenig strukturierten Alarmtabelle. Diese bietet wiederum Detailinformationen zu Alarmen, die im Alarmbaum fehlen. Zusätzlich bietet das neue Alarm-Konzept eine verbesserte Prozessbildsuche zu Alarmen und erleichtert Operateuren den Umgang mit Wartungsalarmen. Die neuen Werkzeuge sind zudem optional einsetzbar, so dass Operateure diese schrittweise, z. B. in weniger kritischen Alarmsituationen, erproben und erlernen können.

Die Entwurfsmetapher *Modernisierung der Altstadt* verdeutlicht die Herangehensweise dieser Arbeit. Dabei werden vorhandene Leitstandsysteme substanzerhaltend saniert und an ausgewählten Stellen modernisiert, ohne bewährte Systeme abzulösen oder massiv zu verändern. Die wenigen Eingriffe in vorhandene Systeme tragen zur Verständlichkeit und Erlernbarkeit der neuen Leitstandfunktionalität bei und erhöhen ihre Gebrauchstauglichkeit. Diese Maßnahmen und der optionale Einsatz neuer Werkzeuge helfen, die

Skepsis der Operateure zu überwinden und sie behutsam an Neuerungen heranzuführen.

7.4 Zusammenfassung

Technische Anlagen sind langlebige Systeme, die während ihres Lebenszyklus wiederkehrend an neue Anforderungen angepasst werden müssen. Statt regelmäßig das gesamte System oder große Teile davon in einem Big-Bang abzulösen und durch neue, potenziell unzuverlässige Software zu ersetzen, wurden in diesem Abschnitt evolutionäre Vorgehensmodelle vorgestellt. Diese unterstützen kontinuierliche Änderungen in vorhandenen Systemen. Die Konzepte Kernsystem und Spezialsysteme erlauben zudem, diese Änderungen in fachlich zusammengehörige Pakete zu segmentieren. Dies hilft während ihrer Umsetzung, Risiken früh zu erkennen und regelmäßig Feedback von Anwendern zu erhalten. Jede dieser Veränderungen wirkt sich potenziell auf die Arbeitsabläufe der Anwender aus, die Veränderungen häufig skeptisch gegenüberstehen und sich Verbesserungen wünschen, ohne ihre bewährten Arbeitsabläufe massiv zu verändern. Für diese Anforderung wurde in diesem Abschnitt die Softwareerweiterung diskutiert. Diese führt neue Funktionalität zusätzlich zur vorhandenen ein, ohne die ursprüngliche Funktionalität zu manipulieren.

Die in diesem Kapitel erarbeiteten Konzepte wurden in der Entwurfsmetapher *Modernisierung der Altstadt* vereint und auf die Leitstandmodernisierung übertragen. Bei diesem Ansatz bleiben bestehende Leitstandwerkzeuge weitgehend erhalten und werden an ausgewählten Stellen, dort wo sie Unzulänglichkeiten aufweisen, durch neue Werkzeuge ergänzt. Operateure erhalten durch die neue Werkzeugkombination eine bessere Unterstützung für ihre Arbeitssituationen und können neue Werkzeuge optional und schrittweise einsetzen. Dabei haben sie stets die Möglichkeit, zu Altbewährtem zurückzukehren, wodurch ihre gewohnten Arbeitsabläufe weniger beeinträchtigt werden. Diese Vorgehensweise steht im Einklang mit der Anwendungsorientierung, bei der die Bedürfnisse der Anwender im Vordergrund stehen und in einem evolutionären Entwicklungsprozess realisiert werden (siehe

Abschnitt 1.3.3). Gleichzeitig wirkt sich diese Vorgehensweise positiv auf die Gebrauchstauglichkeit der Werkzeuge aus, da Operateure bei ihrer Entstehung mitwirken, sie dadurch besser verstehen und einfacher erlernen können.

Der Fokus dieses Kapitels ist in der Abbildung 7.3 im WAM-Dimensionenmodell verortet. Die in diesem Kapitel vorgestellten methodischen und soft-

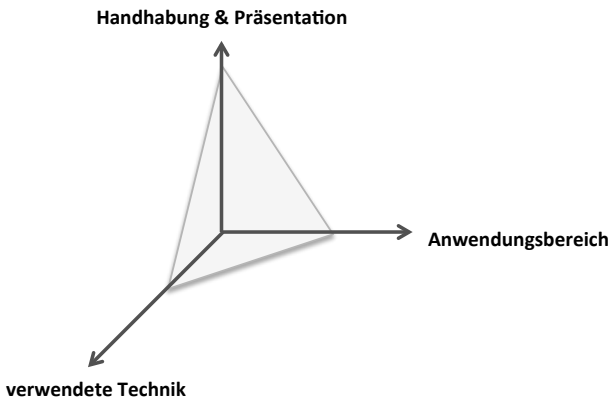


Abbildung 7.3: Einordnung minimalinvasive Leitstandmodernisierung aus anwenderorientierter Sicht im WAM-Dimensionenmodell (Quelle: eigene Erstellung)

waretechnischen Ansätze sowie die Entwurfsmetapher *Modernisierung der Altstadt* beantworten die Forschungsfrage, wie in einer konservativen Leitstandwelt neue Funktionalität eingeführt werden kann, ohne dabei bewährte Arbeitsabläufe der Operateure massiv zu verändern. Damit wurde in diesem Kapitel die konzeptionelle Grundlage bereitet, an der sich die Einführung neuer Leitstandfunktionalität aus einer softwaretechnischen Sicht orientieren kann, was Gegenstand des nächsten Kapitels ist.

Kapitel 8

Minimalinvasive Leitstandmodernisierung aus technischer Sicht

Für die softwaretechnische Realisierung der Metapher *Modernisierung der Altstadt* müssen vorhandene Systeme an möglichst wenigen Stellen angepasst und mit neuer Leitstandfunktionalität ergänzt werden. In Kapitel 6 wurde verdeutlicht, dass Änderungen und Erweiterungen in vorhandenen Systemen zur Architekturerosion beitragen und dadurch die Evolutionsfähigkeit einer Software beeinträchtigen können. Smolander und Hasselbring argumentieren ferner, dass Veränderungen in einem Softwaresystem nicht nur die technische Basis, sondern auch das gemeinsame Verständnis aller Beteiligten über das System beeinflussen. Die im Lebenszyklus einer Software getroffenen Architekturentscheidungen bilden die Grundlage und den Rahmen für zukünftige Anpassungen. Diese Entscheidungen dürfen daher nicht nur implizit, sondern müssen explizit in einer Software modelliert und dokumentiert werden [vgl. Smolander 2002; Hasselbring 2006]. Gerade in langlebigen Leitstandsystemen, in denen viele technische und personelle Veränderungen stattfinden und das Wissen über ein System möglichst lange erhalten werden muss, kommt der Architektur eine entscheidende Rolle zu.

In diesem Kapitel werden aktuelle Architekturansätze mit Fokus auf Leitstandsysteme untersucht, eine passende Architektur herausgearbeitet und Ansätze zur Einführung neuer Leitstandfunktionalität unter Beibehaltung der Integrität der Softwarearchitektur vorgestellt. Dies soll eine strukturierte und organisierte Leitstandmodernisierung ermöglichen. Damit die neue Leitstandfunktionalität nicht losgelöst von der vorhandenen Funktionalität besteht, werden in diesem Kapitel minimalinvasive Ansätze vorgestellt, die neue und vorhandene Leitstandfunktionalität lose gekoppelt verbinden und ihre Interaktion ermöglichen. Die in diesem Kapitel erarbeiteten Ansätze orientieren sich wie zuvor an der Leitstandaufgabe *Alarmierung*, um die technische Realisierung an einem fachlichen Beispiel auszurichten.

8.1 Modellbasierte und modulare Leitstandarchitektur

In Abschnitt 4 wurde die explizite Modellierung der Anlagenstruktur in einem Anlagenmodell diskutiert und in Abschnitt 6.1 die Einordnung des Anlagenmodells und darauf aufbauender Leitstandfunktionalität in eine Gesamtarchitektur motiviert. Diese Architektur sollte möglichst einfach aufgebaut und modular sein, um Änderungen und Erweiterungen besser zu unterstützen. Wie in Kapitel 6.1.3 beschrieben, sind diese Eigenschaften prinzipiell schwer zu quantifizieren und werden im Allgemeinen sehr subjektiv bewertet. Die Auswahl und Konzeption einer geeigneten Architektur ist zudem unternehmens- und projektspezifisch, so dass in dieser Arbeit lediglich allgemeine Architekturaspekte diskutiert werden können. Als Grundlage für die Architekturdiskussion wird in dieser Arbeit die inzwischen etablierte Drei-Schichten-Architektur herangezogen [vgl. Züllighoven et al. 1998, S. 724ff; Hasselbring 2006; Lilienthal 2008, S. 30f]. Diese besteht aus der Präsentationsschicht für die grafische Darstellung, der funktionalen Schicht mit der Anwendungslogik und der Datenschicht mit Schnittstellen zur Datenbank [vgl. z. B. Bäumer 1998, S. 109; Züllighoven et al. 1998, S. 724].

Karagiannis und Kuhn präzisieren diese Schichten und erweitern sie mit Architekturelementen, die sie speziell auf die Konstruktion (meta-)modellbasierter Anwendungen anpassen und in einer modularen, verteilten und skalierbaren Metamodellierungs-Plattform einordnen [Karagiannis und Kuhn, 2002, S. 5]. Diese, in der Abbildung 8.1 dargestellt Architektur, bietet sich zur Strukturierung (anlagen-)modellbasierter Leitstandfunktionalität an und wird im Folgenden vorgestellt. Dabei werden nur ausgewählte, für diese Arbeit besonders relevante Architekturelemente betrachtet:

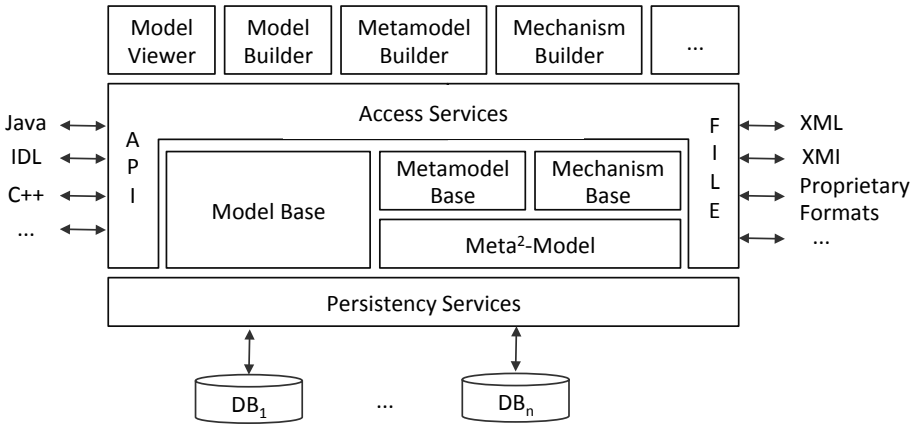


Abbildung 8.1: Generische Architektur für eine Metamodellierungs-Plattform (Quelle: Karagiannis und Kuhn, 2002)

- *Model Viewer:* bietet eine grafische Schnittstelle, mit der Anwender Modelle und Metamodelle betrachten und bearbeiten können.
- *Access Services:* bieten eine Schnittstelle zur Überprüfung der Zugriffsrechte und den verteilten Zugriff auf Modelle und Metamodelle.
- *Model Base:* verwaltet die auf Basis des Metamodells entwickelten Modelle und bietet Schnittstellen mit sondierenden und manipulierenden Operationen.
- *Persistency Service:* persistiert Modelle und Metamodelle in einer Datenbank.

Bei dieser Architektur kann der *Model Viewer* der Präsentationsschicht, die *Access Services* sowie die *Model Base* der funktionalen Schicht und die *Persistency Services* der Datenschicht zugeordnet werden. Nach Züllighoven et al.

ist die klassische Schichtenarchitektur für die Konstruktion großer Softwaresysteme nur bedingt geeignet, da sie die Dimensionen Anwendungsbereich, verwendete Technik sowie Handhabung und Präsentation vermischt. Dadurch können die Schichten nicht unabhängig voneinander entwickelt werden und Änderungen in einer Schicht müssen meist in den anderen Schichten aufwändig nachgepflegt werden [Züllighoven et al., 1998, S. 724ff]. Lilienthal kritisiert, dass die klassische Schichtenarchitektur keine Subsysteme innerhalb der Schichten und keine Einschränkungen der Beziehungen zwischen den Subsystemen abbildet [Lilienthal, 2008, S. 32ff]. Gerade diese Eigenschaft wurde in Abschnitt 6.1.3 als ein wichtiger Einflussfaktor für eine Leitstandmodernisierung herausgearbeitet. In autonomen Subsystemen mit wohldefinierten Schnittstellen steigt die Wahrscheinlichkeit, dass Änderungen nur lokale Auswirkungen haben, die nur ein oder wenige Subsysteme betreffen. Für die Konstruktion großer Softwaresysteme, zu denen Leitstandsysteme zählen, werden daher zusätzliche Strukturierungskonzepte benötigt.

Lilienthal präsentiert dafür eine erweiterte Schichtenarchitektur, bei der die klassischen Schichten um Subsysteme ergänzt und in vertikale Schnitte unterteilt werden (siehe Abbildung 8.2). Für den Begriff *Schnitt* existiert nach Lilienthal bislang keine allgemein übliche Definition und dementsprechend keine allgemein gültige Regel, wie Anwendungen in Schnitte unterteilt werden [vgl. Lilienthal, 2008, S. 34].

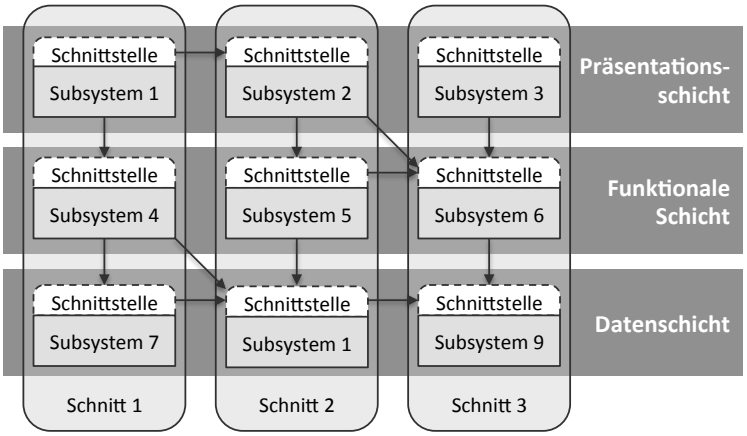


Abbildung 8.2: Erweiterte Drei-Schichten-Architektur (Quelle: eigene Erstellung in Anlehnung an Lilienthal, 2008, S. 34)

In WAM orientieren sich die Konzeptions- und Gestaltungsprinzipien für Architekturen an den fachlichen Gegenständen und Begriffen des Anwendungsbereichs (siehe Abschnitt 6.1.2). Diese Gegenstände und Begriffe bilden oft Kristallisationspunkte um fachliche Anwendungsfälle und können dadurch zu einer Einheit zusammengefasst werden [Kornstädt und Reiswich, 2010b]. Diese Einheiten lassen sich auf Subsysteme übertragen und zu fachlich motivierten Schnitten bündeln. Bäumer und Züllighoven et al. bezeichnen diese Bündelung als Partitionierung eines Geschäftsfeldes in *Produktbereiche* [Bäumer 1998, S. 116; Züllighoven et al. 1998, S. 696ff]. In dieser Arbeit wird der Architekturvorschlag von Karagiannis und Kuhn mit den WAM-Konzepten zur Konstruktion interaktiver Anwendungen aus Abschnitt 1.3.3 kombiniert und in fachlich zusammengehörige Schnitte mit Subsystemen unterteilt. Jeder Schnitt kann dabei folgende Subsysteme und Beziehungen zwischen diesen enthalten [vgl. Kornstädt und Reiswich, 2010b]:

- **Werkzeuge:** In WAM werden *Model Viewer* nach [Karagiannis und Kuhn, 2002] als Werkzeuge realisiert. Eine Anwendung besteht aus mehreren Werkzeugen, die Materialien wie z. B. Anlagenmodelle visualisieren und bearbeiten. Werkzeuge sind oft unabhängig voneinander, bilden aber Cluster um fachliche Anwendungsfälle wie z. B. Alarmierung oder Projektierung. Diese Werkzeuge werden meist nur in Kombination verwendet und sollten daher in eigenen Subsystemen zusammengefasst werden.

- **Materialien:** Während Werkzeuge Materialien bearbeiten und ohne diese nicht funktionieren, können Materialien wie z. B. Alarme auch ohne Werkzeuge existieren und werden in großen Anwendungen über mehrere Kanäle wie z. B. Web- oder auch mobile Clients bearbeitet. Daher ist es vorteilhaft, fachlich zusammengehörige Materialien als separate Subsysteme zu modellieren.

- **Services:** Die *Model Base* nach [Karagiannis und Kuhn, 2002] wird in WAM durch fachliche Services realisiert (siehe dazu Abschnitt 8.2.2). Services verwalten Materialien und bieten einen fachlich motivierten Umgang mit diesen. Für Services existieren alternative Implementie-

rungen, die bei Bedarf ausgetauscht werden können. Ihre Kapselung in eigenen Subsystemen erleichtert den Austausch.

- **Datenzugriffsschicht:** In einer Client/Server-Leitstandarchitektur interagieren Clients und Server über eine konkrete Remote-Technologie. Die Datenzugriffsschicht kapselt diese Technologie, um sie von der Anwendungslogik zu entkoppeln und einfacher austauschen zu können (siehe dazu Abschnitt 8.3.3). Die Remote-Technologie sollte daher im eigenen Subsystem gekapselt werden und entspricht den *Access Services* nach [Karagiannis und Kuhn, 2002].

- **Persistenzschicht:** Um die Persistenz-Technologie oder die Datenbank einfacher austauschen zu können, sollten die Persistenz-Dienste in separaten Subsystemen gekapselt werden. Diese Dienste entsprechen dem *Persistency Service* nach [Karagiannis und Kuhn, 2002].

Diese Architekturkonzepte bieten eine Grundlage für die Konstruktion modellbasierter und modularer Anwendungen von der Persistenzschicht bis zu den Werkzeugen. Die vorgestellten Schnitte und Subsysteme erweitern die klassische Drei-Schichten-Architektur und bieten zusätzliche Strukturierungskonzepte für die Entwicklung großer Leitstandsysteme. Die Kombination der Metamodellierungs-Plattform nach Karagiannis und Kuhn mit der WAM-Architektur fördert das Verständnis der Entwickler über die Architektur, erleichtert Entwurfsdiskussionen und führt zu einer hohen Strukturähnlichkeit zwischen den softwaretechnischen Modellen und den Gegenständen des Anwendungsbereich. Die Datenzugriffs- und Persistenzschicht entkoppeln zudem fachliche Konzepte von ihren technischen Realisierungen und helfen, Änderungen in der jeweiligen Schicht lokal zu halten. Diese Eigenschaften tragen zur Verbesserung der Analysierbarkeit und Erweiterbarkeit der Leitstandsysteme bei und unterstützen somit die Leitstandmodernisierung.

In den folgenden Abschnitten wird die Metapher *Modernisierung der Altstadt* auf der technischen Ebene konkretisiert und zunächst Ansätze zur Einführung neuer Leitstandfunktionalität im Rahmen der in Abschnitt 7.2 eingeführten Softwareerweiterung vorgestellt. Anschließend werden Ansätze zur losen Kopplung neuer und vorhandener Funktionalität herausgearbeitet.

8.2 Einführung neuer Leitstandfunktionalität

Die Einführung neuer Leitstandfunktionalität, wie das Kernsystem oder die Spezialsysteme, betrifft häufig alle Schichten einer Softwarearchitektur. Sie bedarf geeigneter Konzepte, um diese Änderungen möglichst lokal zu halten und negative Seiteneffekte auf das vorhandene System zu reduzieren (siehe Abbildung 8.3). Im vorangegangenen Abschnitt sowie in Abschnitt 6.1.3

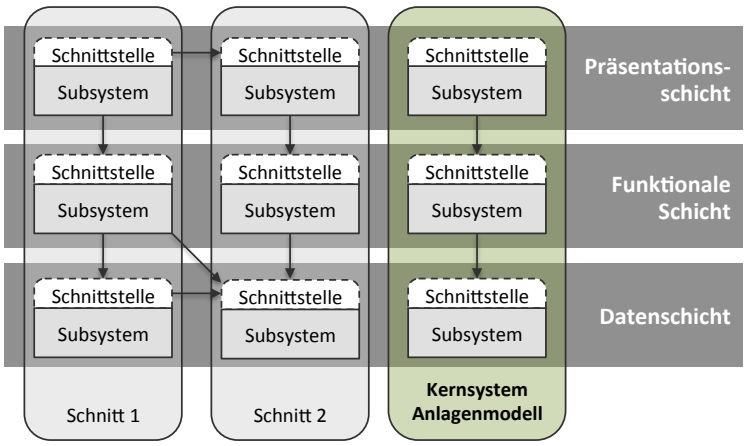


Abbildung 8.3: Einführung des Kernsystems *Anlagenmodell* (Quelle: eigene Erstellung)

wurde die Modularisierung als ein Ansatz diskutiert, der Änderbarkeit und Erweiterbarkeit eines Systems mit lokalen Auswirkungen unterstützt. In der Vergangenheit haben sich Softwaresysteme, im Hinblick auf die Modularisierung, von rein monolithischen Kernen, über erweiterbare Anwendungskerne mit sog. Add-ons [Kornstädt, 2002], hin zu Plug-in-basierten Systemen entwickelt (siehe Abbildung 8.4). In Plug-in-basierten Anwendungen besteht das gesamte System ausschließlich aus modularen Plug-ins, die durch eine minimale Plattform geladen und ausgeführt werden [vgl. Daum 2007; McAffer und Lemieux 2005]. Die Plug-in Plattform bietet standardisierte Mechanismen, die Änderungen und Erweiterungen mit lokalen Auswirkungen unterstützen. Dafür müssen Plug-ins möglichst autonom sein, d.h. nur wenige Abhängigkeiten mit dedizierten Schnittstellen zu anderen Plug-ins haben. Dadurch können sie Daten und Dienste austauschen und sich gleichzeitig weitgehend unabhängig voneinander entwickeln [vlg. Meyer 1997, S. 39ff; Hohpe und Woolf 2004, S. 39f].

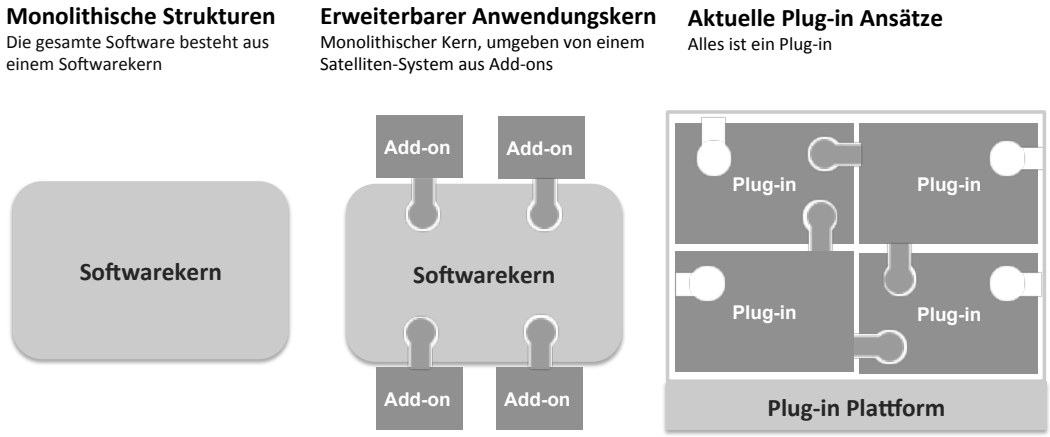


Abbildung 8.4: Historische Entwicklung modularer Systeme (Quelle: eigene Erstellung)

In einer Plug-in-basierten Anwendung können Plug-ins neue Funktionalität über wohldefinierte Schnittstellen anbieten und dabei selbst neue Schnittstellen einführen, an die sich wiederum andere Plug-ins andocken können. Dadurch entsteht ein Plug-in-Ökosystem, bei dem die Funktionalität einer Software kontinuierlich an die individuellen Wünsche der Kunden angepasst werden kann. Die wohldefinierten Schnittstellen ermöglichen, neue Funktionalität in vorhandene Anwendungen einzuführen, ohne dass Entwickler dafür während der initialen Softwareentwicklung Schnittstellen für potentielle Erweiterungen antizipieren müssen. Plug-ins können neue Funktionalität über Erweiterungspunkte und Erweiterungen, Dienste oder Bibliotheken einführen (Abbildung 8.5):

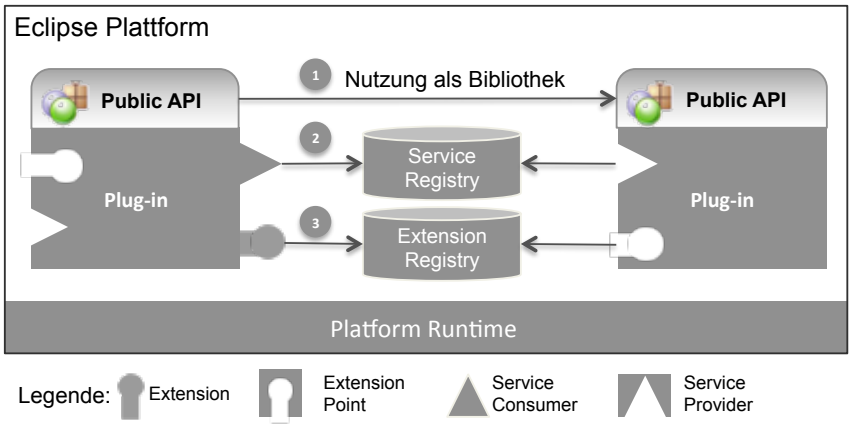


Abbildung 8.5: Interaktionsmöglichkeiten zwischen Plug-ins (Quelle: eigene Erstellung)

- **Bibliothek:** Plug-ins können andere Plug-ins als Bibliothek nutzen. Hierfür verfügen Plug-ins über eine Schnittstelle (engl. public Application Programming Interface, API), auf die andere Plug-ins zugreifen können. Zusätzlich verfügen Plug-ins über einen privaten Bereich, der von außen weder sichtbar noch zugreifbar ist. Die *Platform Runtime* stellt die öffentliche Funktionalität systemweit zur Verfügung und verhindert gleichzeitig Zugriffe auf den privaten Bereich.

- **Services:** Plug-ins können anderen Plug-ins ihre Funktionalität über Dienste anbieten, die entweder technisch oder fachlich motiviert sind. Dies geschieht lose gekoppelt über eine Service-Registratur. Die Eclipse Plattform bietet eine Reihe vorgefertigter Dienste für wiederkehrende Aufgaben wie z. B. Benutzerverwaltung, Logging oder Administration der Plug-ins [vgl. McAffer et al. 2010; Wütherich et al. 2008].

- **Erweiterungspunkte und Erweiterungen:** Erweiterungspunkte (engl. extension points) sind Schnittstellen, an denen Plug-ins andere Plug-ins durch eine dazu passende Implementation (engl. Extensions), mit neuer Funktionalität erweitern können. Erweiterungspunkte und Erweiterungen werden in einer entsprechenden Registratur angemeldet und dort von der *Platform Runtime* ausgeführt. Die Eclipse Plattform bietet eine Reihe vorgefertigter Erweiterungspunkte für z. B. die Einbindung neuer Werkzeuge, Aktionen in Menüs, Toolbars und Kontextmenüs [vgl. Daum 2007; McAffer und Lemieux 2005].

Plug-ins können als Bibliothek entwickelt oder genutzt werden, Services bereitstellen oder nutzen und Erweiterungspunkte definieren oder dafür passende Erweiterungen bereitstellen. Diese Konzepte lassen sich einzeln nutzen oder auch kombinieren, wobei nicht alles was technisch machbar ist, auch praktisch sinnvoll ist. Kornstädt und Reiswich differenzieren diese Konzepte und ordnen sie dem jeweiligen Verwendungszweck zu [Kornstädt und Reiswich, 2010b]. Dazu übertragen sie zunächst die in Abschnitt 8.1 vorgestellten fachlichen Architekturkonzepte auf die Plug-in-Technologie und konkretisieren, wie modulare Anwendungen mit Plug-ins realisiert werden können. In ihren Architekturempfehlungen schlagen sie vor, Erweiterungspunkte und Erweiterungen in erster Linie für die Einführung neuer UI-Elemente, wie z. B.

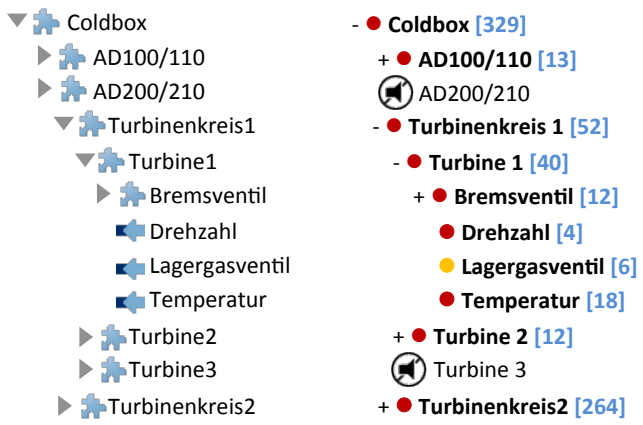
Werkzeuge oder Menüs zu verwenden. Für diese clientseitige Funktionalität bietet Eclipse bereits viele Vorlagen, die Entwicklern die Werkzeugkonstruktion erleichtert. Services eignen sich besonders für die Entwicklung fachlich motivierter Dienste. Diese werden in mehreren Clients benötigt und laufen daher zentral auf einem Server ab. Mit den vorgestellten Plug-in-Konzepten können Softwareanbieter client- als auch serverseitige Funktionalität entwickeln und diese wiederkehrend an neue (Business-) Anforderungen anpassen. Dazu können neue Plug-ins minimalinvasiv in vorhandene Anwendungen eingeführt, aber auch veraltete Plug-ins mit wenigen Eingriffen aus dem System herausgelöst werden [vgl. z. B. Marples und Kriens 2001; Wütherich et al. 2008; McAffer et al. 2010].

In dieser Arbeit wird die Leitstandmodernisierung am Beispiel der modularen Eclipse Plug-in Technologie vorgestellt, die aktuell in der Eclipse Rich Client Platform (RCP) umgesetzt ist [vgl. McAffer und Lemieux 2005; Daum 2007]. Die Entscheidung für Eclipse RCP resultiert in erster Linie aus der Tatsache, dass die DESY-Leitstandsoftware auf der Eclipse Plug-in Technologie basiert. Zudem ist diese Technologie in der Industrie weit verbreitet und gehört weltweit zu den bekanntesten Vertretern modularer Technologien [vgl. z. B. Birsan 2005; McAffer und Lemieux 2005; Kornstädt und Reiswich 2010a].

Die Eclipse-Erweiterungskonzepte unterstützen eine kleinschrittige Einführung neuer Leitstandfunktionalität als Kernsystem mit Spezialsystemen. Da die Spezialsysteme auf dem Kernsystem aufbauen, verwenden sie einen Teil seiner Funktionalität wieder. Dabei besteht die Gefahr, die Spezialsysteme zu eng an das Kernsystem zu binden und diese Systeme langfristig zu einem großen Monolithen zu verschmelzen. Die eingangs dieses Kapitels vorgestellte Gesamtarchitektur und die Plug-in-Konzepte müssen daher durch weitere Architekturkonzepte begleitet werden, die eine langfristige Evolutionsfähigkeit der neuen Leitstandfunktionalität ermöglichen.

8.2.1 Erweiterung des Anlagenmodell-Werkzeugs auf der Präsentationsschicht

In Kapitel 4.5 wurde das Anlagenmodell-Werkzeug und in Kapitel 5 ein darauf aufbauendes Alarmbaum-Werkzeug eingeführt. Beim direkten Vergleich dieser Werkzeuge wird ihre große Ähnlichkeit deutlich (siehe Abbildung 8.6). Dadurch ist das Anlagenmodell-Werkzeug dafür prädestiniert, wiederverwendet und an ausgewählten Stellen mit alarmspezifischer Funktionalität ergänzt zu werden. In der Literatur werden dazu verschiedene Ansätze diskutiert, von denen im Folgenden eine Auswahl vorgestellt wird. Diese Auswahl fokussiert speziell auf Ansätze zur Werkzeug-Wiederverwendung:



(a) Anlagenmodell (b) Alarmbaum

Abbildung 8.6: Modellbasierte Werkzeuge (Quelle: eigene Erstellung)

- **Duplizieren:** Dabei wird der Quelltext des Anlagenmodell-Werkzeugs kopiert und an den entsprechenden Stellen um alarmspezifische Funktionalität ergänzt. Dadurch wird zwar das ursprüngliche Anlagenmodell-Werkzeug vollständig erhalten, die Werkzeug-Kopie führt jedoch zu Code-Duplikaten, wodurch bereits minimale Änderungen der gleichen Funktionalität zumindest in zwei Werkzeugen nachgepflegt werden müssen.

- **Implementationsvererbung:** Die neue Funktionalität wird in der erbenden Werkzeug-Unterklasse ergänzt. Implementationsvererbung ver-

ursacht das Problem der änderungsempfindlichen Oberklassen (engl. *fragile base class problem*) [Szyperski, 2002, S. 115]. Dabei können sich Änderungen in den Oberklassen auf alle erbenden Klassen auswirken und müssen dort nachgepflegt werden. Dies führt zu einer engen Kopplung zwischen den Werkzeugen und bricht ihre Kapselung auf.

- **Dekorierer:** Erlaubt, die Funktionalität einzelner Klassen zu erweitern, ohne sie zu verändern. Der Dekorierer umschließt ein zu kapselndes Objekt, leitet alle Operationsaufrufe an das eingeschlossene Objekt weiter und führt möglicherweise davor oder danach zusätzliche Operationen aus [Gamma et al., 2009, S. 199ff]. Das Dekorierer-Muster ähnelt dem Adapter bzw. Wrapper-Muster, das eine Zusammenarbeit zwischen Klienten mit unterschiedlichen Schnittstellen ermöglicht. Die Muster unterscheiden sich nach Gamma et al. dadurch, dass „[. . .] ein Dekorierer nur die Funktionalität eines Objektes verändert, nicht aber seine Schnittstelle“ [Gamma et al., 2009, S. 211]. Softwaresysteme mit vielen Dekorierern, sind häufig schwerer zu verstehen und zu debuggen.

Das Anlagenmodell-Werkzeug bildet die Basis für viele darauf aufbauende Leitstand-Werkzeuge und wird dadurch in vielen Kontexten wiederverwendet. Diese Werkzeuge werden im Lebenszyklus einer Leitstandsoftware wiederkehrend angepasst. Damit diese Änderungen möglichst lokale Auswirkungen haben und wenige Seiteneffekte auf andere Werkzeuge verursachen, ist eine lose Kopplung der Werkzeuge und eine hohe Wiederverwendung der gemeinsamen Code-Basis nötig. Für diese Anforderungen bietet das Dekorierer-Muster von allen vorgestellten Alternativen die größtmögliche Flexibilität und lose Kopplung der Werkzeuge. In dieser Arbeit wird das Anlagenmodell-Werkzeug mit Hilfe des Dekorierer-Musters um kontextspezifische Funktionalität ergänzt und auf dieser Basis Werkzeuge für Spezialsysteme wie z. B. das Alarmbaum-Werkzeug konstruiert.

In großen Anlagen arbeiten Operateure räumlich verteilt und müssen in Alarmsituationen kooperieren und die Bearbeitung der Alarme untereinander abstimmen. Für diese Zusammenarbeit reichen unter den Operateuren gemeinsam geteilte Konventionen allein nicht aus. Vielmehr muss die Kooperation und Koordination explizit modelliert werden, da sie die Vorstellung der

Operateure von der gegenwärtigen Arbeitssituation und ihre Zusammenarbeit beeinflusst. In aktuellen Leitständen fehlt häufig diese Modellierung und führt wiederkehrend zu Problemen und Unfällen und bedarf einer geeigneten Lösung, die im nächsten Abschnitt erarbeitet wird.

8.2.2 Erweiterung des Anlagenmodells auf der funktionalen Schicht

In Abschnitt 3.2.2 wurde die kooperative Zusammenarbeit der Operateure in Alarmsituationen vorgestellt und die damit einhergehenden Probleme auf ein fehlendes Alarmmodell zurückgeführt. Aus einer anwendungsfachlichen Sicht soll ein Alarmmodell alle in der Anlage aufgetretenen Alarme vorhalten, Berechnungen über und Stummschaltung von Alarmen ermöglichen und die kooperative Zusammenarbeit der Operateure unterstützen. Für die Realisierung dieser Funktionalität müssen Anlagenteile mit alarmspezifischen Informationen verknüpft werden. Diese Informationen müssen zudem auf die Struktur der Anlage abgebildet werden, denn der Alarmzustand eines Anlagenteils ergibt sich aus den Alarmzuständen darin verbauter Subkomponenten. Beispielsweise kann die Frage, welche Alarme in einem Motor existieren, nur dann sinnvoll beantwortet werden, wenn die strukturelle Zusammensetzung des Motors aus seinen Subkomponenten, wie z. B. Öl- und Wassertemperatursensoren und ihren Alarmzuständen, explizit modelliert und von anderen Anlagenteilen abgegrenzt ist. Fehlt diese strukturelle Modellierung, müssen zur Beantwortung der Frage nach Alarmen in einem Motor potenziell alle verfügbaren Anlagenteile sondiert und ihre Zugehörigkeit zum Motor überprüft werden. In großen Anlagen kann diese Aufgabe tausende Anlagenteile umfassen. Für das Alarmmodell wird somit eine zum Anlagenmodell ähnliche hierarchische Struktur benötigt. Das Anlagenmodell ist somit dafür prädestiniert, im Kontext der Alarmierung wiederverwendet zu werden.

Die Alarmierung ist zwar ein wesentlicher, jedoch kein exklusiver Anwendungsfall für die Wiederverwendung der Anlagenmodelle. Die Leitstandaufgaben Datenzugriff und historischer Datenzugriff eröffnen weitere Anwendungsfelder, in denen das Anlagenmodell wiederverwendet und mit kontextspezifi-

scher Funktionalität erweitert werden kann. Mit jeder Erweiterung droht das Anlagenmodell unübersichtlich und schwerer wart- und erweiterbar zu werden (siehe Abschnitt 4.4). Für diese Arbeit wird daher ein Ansatz benötigt, der eine Wiederverwendung der Anlagenmodelle in diversen Kontexten erlaubt, ohne langfristig die Evolutionsfähigkeit der Anlagenmodelle einzuschränken.

Wie in Abschnitt 4.5 beschrieben, ist ein Anlagenmodell eine hierarchische Dekomposition aus Anlagenteilen, die softwaretechnisch als Knoten in einem Baum modelliert werden. Für die Erweiterung des Anlagenmodells mit kontextspezifischer Funktionalität bieten sich prinzipiell folgende Optionen an, die in der Abbildung 8.7 auf einer konzeptionellen Ebene visualisiert sind. Die vorgestellten Optionen erheben keinen Anspruch auf Vollständigkeit und sollen lediglich ausgewählte und in der Literatur oft diskutierten Alternativen mit ihren Vor- und Nachteilen kurz vorstellen:

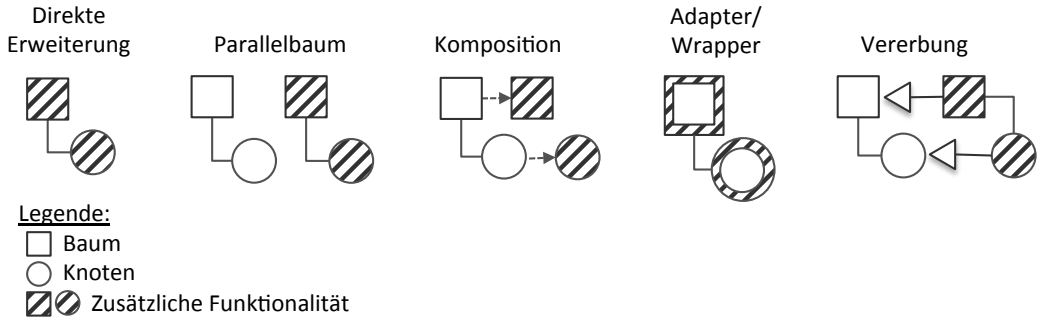


Abbildung 8.7: Mögliche Alternativen bei der Erweiterung eines Anlagenmodells mit neuer Funktionalität (Quelle: eigene Erstellung)

- **Modell direkt erweitern:** Die zusätzliche Funktionalität kann direkt im Anlagenmodell und dessen Knoten ergänzt werden. Diese Lösung ist einfach zu implementieren, verlagert jedoch kontextspezifische Funktionalität in das Anlagenmodell und vermischt Zuständigkeiten. Da die gesamte fachliche Funktionalität in einem Modell gekapselt ist, können Klienten in beliebigen Kontexten stets die gesamte Funktionalität des Modells nutzen, wie z. B. Alarme oder Anlagenteile löschen, ohne Möglichkeiten dies zu unterbinden. Mit jedem neuen Anwendungskontext muss zudem das Anlagenmodell erweitert werden, was gerade bei

langlebigen Leitstandsystemen langfristig zu einem komplexen, schwer verständlichen und schwer wartbaren Modell führen würde.

- **Modell duplizieren:** Die kontextspezifische Funktionalität kann in einem Duplikat ergänzt werden. Dadurch werden die Zuständigkeiten zwischen dem originalen und duplizierten Modell klar getrennt und die Modelle können sich unabhängig voneinander entwickeln. In langlebigen Leitstandsystemen sind Änderungen im Anlagenmodell unvermeidlich, so dass diese in jedem Duplikat aufwändig nachgepflegt werden müssen. Mit jeder Änderung steigt zudem das Risiko, dass diese nicht konsistent durchgeführt werden und dadurch Modelle mit unterschiedlichen Versionen und Verhalten entstehen.

- **Komposition:** Die neue Funktionalität kann aus dem Anlagenmodell und seinen Knoten herausgelöst und in separaten Komponenten bereitgestellt werden. Das Anlagenmodell und seine Knoten referenzieren die zusätzliche Funktionalität und geben sie bei Bedarf an Klienten heraus. Dieser Ansatz ist in Gamma et al. als Kompositum-Muster bekannt [Gamma et al., 2009, S. 239]. Er trennt klar die Zuständigkeiten und bietet ausreichend Flexibilität für Erweiterungen. Auch bei diesem Muster können unberechtigte Zugriffe auf ausgewählte Modellfunktionalität nicht ohne weiteres unterbunden werden.

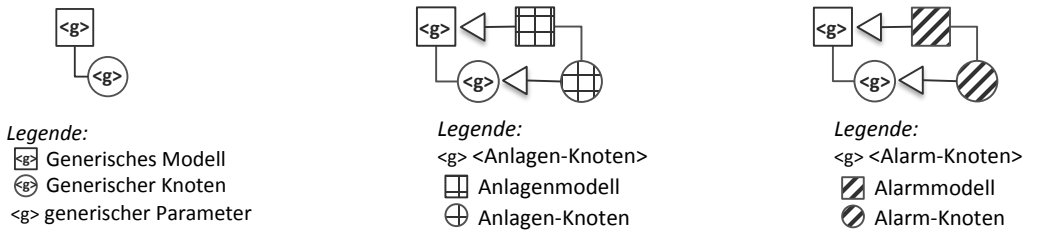
- **Wrapper:** Die neue Funktionalität kann in einer Hülle ergänzt werden, die das Anlagenmodell und seine Knoten umschließt. Diese Hülle bezeichnen Gamma et al. als Adapter oder Wrapper [Gamma et al., 2009, S. 71]. Ein Wrapper delegiert bei Bedarf Operationsaufrufe der Klienten an das eingehüllte Objekt und bietet neue Operationen für zusätzliche Funktionalität. Dadurch kann das eingehüllte Objekt in diversen Anwendungsbereichen wiederverwendet werden, ohne dieses mit kontextspezifischer Funktionalität zu kontaminieren. Damit ein Wrapper Operationsaufrufe der Klienten an das gekapselte Objekt delegieren kann, muss er dessen Schnittstelle implementieren. Ändert sich diese, müssen alle Wrapper aufwändig nachgepflegt werden. Für Klienten ist zudem nicht explizit ersichtlich, ob und in welchem Kontext Wrap-

per zur Verfügung stehen, was die Analysierbarkeit einer Anwendung erschwert.

- **Implementationsvererbung:** Die zusätzliche Funktionalität kann in Subklassen des Anlagenmodells und seiner Knoten ergänzt werden. Dies reduziert die Code-Duplikation und erhöht die Wartbarkeit der Modelle. Wie in Abschnitt 8.2.1 beschrieben, verursacht Implementationsvererbung das Problem der änderungsempfindlichen Oberklassen. Auch bei der Implementationsvererbung können alle erbenden Klassen direkt auf die interne Implementierung der Oberklassen zugreifen, wodurch das Prinzip der Kapselung verletzt wird.

Für diese Arbeit wird ein Lösungsansatz benötigt, der bei langlebigen Leitstandsystemen nicht nur aktuelle, sondern auch zukünftige Anforderungen an Anlagenmodelle in diversen Anwendungskontexten flexibel unterstützt. Er sollte die Zuständigkeiten der Modelle klar trennen und den Wartungsaufwand reduzieren. Die Modelle sollen zudem langfristig kompatibel bleiben, um ihre einheitliche Handhabung in z. B. gemeinsam genutzten Werkzeugen zu ermöglichen. Für diese Anforderungen bieten die vorgestellten Ansätze nur partiell Lösungen. Während die Implementationsvererbung die Wiederverwendbarkeit erhöht und die Code-Duplikation reduziert, bietet sie weniger Flexibilität bei der Erweiterung der Anlagenmodelle mit klar getrennten Zuständigkeiten. Dies bietet wiederum die Duplikation der Modelle, was jedoch langfristig auf Kosten der Wartbarkeit und Kompatibilität der verschiedenen Modelle geht.

Als Lösungsansatz für die aufgezeigten Probleme diskutiert Meyer anhand der objektorientierten Konzepte *Implementationsvererbung* und *Generizität* Möglichkeiten zur Verbesserung der Wiederverwendbarkeit und Flexibilität, bei gleichzeitiger Erhaltung der Kompatibilität zwischen Softwarekomponenten [Meyer, 1986]. Generizität bietet in statisch typisierten Programmiersprachen die Möglichkeit, den Typ der Variablen in Klassen nicht explizit festzulegen, sondern erst bei der konkreten Verwendung einer Klasse zu übergeben. Dadurch wird die Typsicherheit gewahrt und gleichzeitig die Flexibilität einer Klasse erhöht [vgl. Meyer, 1986]. Implementationsvererbung und Generizität weisen nach Meyer Gemeinsamkeiten und Unterschiede auf, können aber



(a) Generisches Modell

(b) Spezialisierung *Anlagenmodell*

(c) Spezialisierung *Alarmmodell*

Abbildung 8.8: Konzept generisches Modell mit Spezialmodellen (Quelle: eigene Erstellung)

bei ihrer komplementären Verwendung die Wiederverwendbarkeit und Flexibilität erhöhen, bei gleichzeitiger Erhaltung der Kompatibilität zwischen Softwarekomponenten [vgl. Meyer, 1986, S. 404]. In dieser Arbeit werden diese Konzepte auf die Anlagenmodellierung übertragen und erweitert. Abbildung 8.8 präsentiert diesen Ansatz auf einer konzeptionellen Ebene am Beispiel des Anlagenmodells und des Alarmmodells.

Statt die vielfältige und umfangreiche Leitstandfunktionalität in einem großen Modell zu vereinen, sind in der Abbildung 8.8 das *Anlagenmodell* (b) und das *Alarmmodell* (c) als eigenständige *Spezialmodelle* realisiert. Dabei entstehen mit jedem Spezialmodell dedizierte und auf den jeweiligen fachlichen Anwendungsfall zugeschnittene Modelle mit dazu passenden Schnittstellen. Im Falle der Alarmierung bietet beispielsweise ein Alarmmodell die Funktionalität, Alarme zu Anlagenteilen hinzuzufügen oder zu löschen sowie Berechnungen über Alarme entlang der Anlagenstruktur durchzuführen. Ein Anlagenmodell kapselt wiederum die Funktionalität, die Anlagenstruktur und Konfiguration der Anlagenteile zu verwalten und zu verändern. Diese Trennung ermöglicht Zugriffe auf die Funktionalität der Modelle auf den jeweiligen Kontext zu beschränken. Zudem können beide Modelle weitgehend unabhängig voneinander erweitert und überarbeitet werden.

Das Anlagenmodell und das Alarmmodell sind hierarchisch organisiert und strukturähnlich. Beide Modelle müssen die dafür erforderliche baumspezifische Funktionalität implementieren. Dazu gehören beispielsweise das Er-

stellen und Löschen von Knoten im Baum, Zugriffe auf Knoten und darin gekapselte Informationen sowie die Navigation entlang der Baumstruktur. Um die Code-Duplikation und den Wartungsaufwand beider Modelle zu reduzieren, ist die baumspezifische Funktionalität in einem *generischen Modell* ausgelagert (Bild (a) in der Abbildung 8.8). Das generische Modell kann in diversen Anwendungskontexten durch konkrete Spezialmodelle über Vererbung erweitert und mit kontextspezifischer Funktionalität ergänzt werden. Dabei müssen die Spezialmodelle bei der Konkretisierung des generischen Modells den Typ ihrer Knoten als *generischen Parameter* übergeben. Dadurch kann die Typsicherheit der Spezialmodelle gewährleistet und sichergestellt werden, dass nur kompatible Knoten in Spezialmodellen miteinander kombiniert werden. Die im generischen Modell gekapselte und über die Zeit relativ stabile baumspezifische Funktionalität reduziert das mit der Implementationsvererbung einhergehende Problem der änderungsempfindlichen Oberklassen. Da in langlebigen Leitstandsystemen Änderungen im generischen Modell nicht ausgeschlossen sind, müssen diese ggf. in den Spezialmodellen nachgepflegt werden.

Abbildung 8.9 präsentiert das Klassendiagramm zum generischen Modell mit Spezialmodellen. `GenericModel` und `GenericNode` kapseln die generische Modell-Funktionalität. Diese kapselt die Navigation in und Organisation einer Baumstruktur. Spezialmodelle wie das `AlarmModel` oder `PlantModel` erweitern das generische Modell mit kontextspezifischer Funktionalität. Im Falle der Alarmierung bietet beispielsweise die zum `AlarmModel` zugehörige Knoten-Klasse `AlarmNode` Operationen wie `getAlarmQuantity()` oder `getAlarmSeverity()`.

Der Einsatz von Spezialmodellen eignet sich in Anwendungsbereichen, in denen Klienten Anfragen an Anlagenteile stellen und diese nicht über die erforderlichen Informationen verfügen, um die Anfrage selbständig zu beantworten. Dazu muss ein Anlagenteil die Anfrage entlang der hierarchischen Anlagenstruktur delegieren, bis diese von einem Knoten mit ausreichenden Kontextinformationen beantwortet werden kann [vgl. Riehle, 1997]. In vielen Anwendungsfällen benötigen Klienten Kontextinformationen zu Anlagenteilen, für die weder die strukturelle Zusammensetzung, noch der Zustand eines

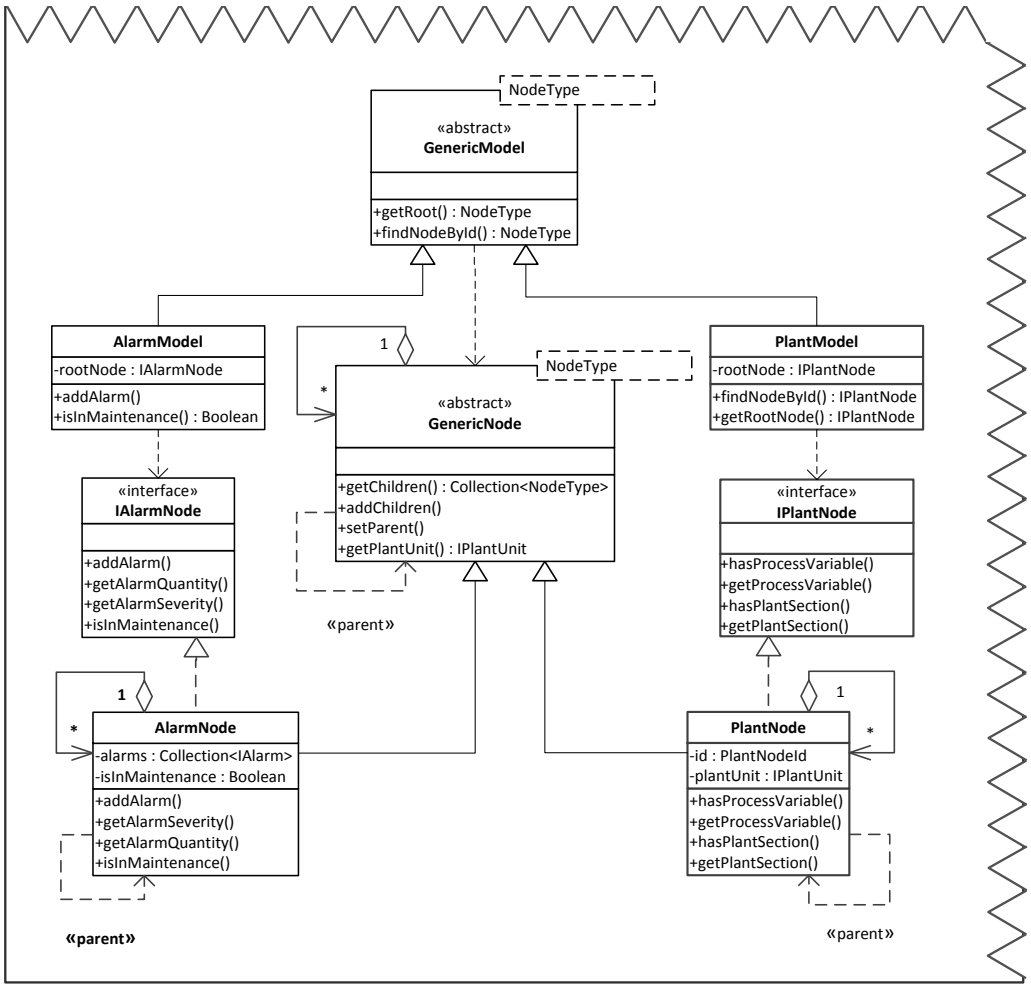


Abbildung 8.9: Generisches Modell mit Spezialmodellen (Quelle: eigene Erstellung)

Anlagenteils relevant ist. Um die Seriennummer, Leistung, Preis oder das Datum der letzten Wartung eines Motors zu erfahren, muss dieser lediglich eindeutig identifiziert und dazugehörige Kontextinformationen bezogen werden. In dieser Arbeit werden daher *anlagenstrukturbezogene* und *anlagenstrukturunabhängige* Informationen differenziert, da ihre Unterscheidung Auswirkungen auf die softwaretechnische Modellierung hat:

Begriff 8.1: Anlagenstrukturbezogene Informationen
Erfordern eine Ordnung der Anlagenteile in einer Anlagenhierarchie und kapseln zu Anlagenteilen zugehörige Zustände. Die strukturelle Zusammensetzung der Anlagenteile hat eine semantische Bedeutung und ermöglicht dadurch, den Zustand einzelner Anlagenteile über Berechnungen entlang der Anlagenstruktur zu höherwertigen Informationen zu aggregieren.

Begriff 8.2: Anlagenstrukturunabhängige Informationen
Liefern Kontext- oder Metainformationen zu Anlagenteilen, unabhängig von ihrer Ordnung in einer Anlagenhierarchie. Sie setzen lediglich eine eindeutige Abbildung von Anlagenteilen auf dazugehörige Kontextinformationen voraus.

Für die Verwaltung und Zugriffe auf anlagenstrukturunabhängige Informationen müssen diese in einer Client/Server-Leitstandarchitektur mehreren Klienten als Materialien nach WAM zur Verfügung stehen, um ihre gemeinsame Verwendung zu ermöglichen. Zudem müssen diese Materialien mit Anlagenteilen verknüpft und ihre Beziehung synchronisiert werden, wenn beispielsweise Anlagenteile entfernt oder ausgetauscht werden. Es existiert demnach ein fachlicher Umgang mit anlagenstrukturunabhängigen Materialien, für deren Modellierung einfache Datenbehälter nicht ausreichen, da sie lediglich einen technischen Umgang mit Materialien abbilden [Züllighoven et al., 1998, S. 188ff]. Für die aufgeführten Anforderungen werden in der Literatur fachliche Services vorgeschlagen, die Materialien verwalten und einen fachlichen Umgang mit diesen ermöglichen [vgl. z. B. Allen und Frost 1998, S. 32ff; Otto und Schuler 2000, S. 67ff; Züllighoven 2005, S. 173ff]. Fachliche Services bündeln zusammengehörige Funktionalität und stellen diese zentral allen Klienten zur Verfügung. Sie können im WAM-Dimensionenmodell (siehe z. B. Abschnitt 4.4.5) dem *Anwendungsbereich* zugeordnet werden und sollten daher möglichst unabhängig von konkreten Benutzungsschnittstellen und Technologien sein. In dieser Arbeit werden Zugriffe auf anlagenstrukturunabhängige Informationen als fachliche Services realisiert.

8.2.3 Zusammenfassung

In diesem Abschnitt wurde die Einführung neuer Leitstandfunktionalität in vorhandene Systeme im Rahmen einer Softwareerweiterung diskutiert. Dazu wurden modulare Ansätze auf Basis der Plug-in-Technologie vorgestellt, die Änderungen und Erweiterungen mit lokalen Auswirkungen unterstützen. Diese Konzepte ermöglichen, das Kernsystem und darauf aufbauende Spezialsysteme (wie Alarmierung) in eine Leitstandsoftware einzuführen, ohne dass Entwickler während der initialen Entwicklung dafür Schnittstellen antizipieren müssen. Zur Entkopplung des Kernsystems von seinen Spezialsystemen wurden in den vorangegangenen Abschnitten zunächst Ansätze auf der Präsentationsschicht vorgestellt und mit dem Dekorierer-Muster ein passender Ansatz für die Wiederverwendung des Anlagenmodell-Werkzeugs gewählt. Im weiteren Verlauf des Kapitels wurde die explizite Modellierung der Alarme und die Unterstützung der kooperativen Alarmbearbeitung durch Operateure über das Kooperationsmedium *Alarmmodell* diskutiert. Für die technische Realisierung des Alarmmodells wurden mehrere Ansätze zur Wiederverwendung des Anlagenmodells untersucht und ein generisches Modell mit Spezialmodellen herausgearbeitet. Schließlich wurden in diesem Abschnitt anlagenstrukturbezogene und anlagenstrukturunabhängige Informationen differenziert und für letztere fachliche Services als softwaretechnische Lösung vorgestellt.

Anlagenmodelle, Alarmmodelle, dazugehörige Werkzeuge und fachliche Services stellen keine Insellösungen dar und müssen ihre Funktionalität in Zusammenarbeit mit der vorhandenen Leitstandfunktionalität erbringen. Diese Zusammenarbeit wird oft dadurch erschwert, dass die vorhandene Leitstandfunktionalität ursprünglich ohne die Absicht entwickelt wurde, sie später zu erweitern und mit neuer Funktionalität zu ergänzen. Zudem werden in Leitständen heterogene Systeme verschiedener Hersteller eingesetzt, die über unterschiedliche Schnittstellen und Datenformate kommunizieren. Neue Leitstandkomponenten müssen mit diesen Systemen zusammenarbeiten. Sie bedürfen daher geeigneter Konzepte, die diese Zusammenarbeit ohne massive Eingriffe in vorhandene Systeme ermöglichen. Diese Aspekte werden in den folgenden Abschnitten diskutiert.

8.3 Lose Kopplung neuer und vorhandener Leitstandfunktionalität

In Abschnitt 8.2 wurden Plug-in-basierte Konzepte vorgestellt, die Änderungen und Erweiterungen vorhandener Systeme mit lokalen Auswirkungen unterstützen. Dafür müssen Plug-ins möglichst autonom sein, d.h. nur wenige Abhängigkeiten zu anderen Plug-ins haben. Je autonomer die Plug-ins einer Anwendung sind, desto geringer fällt auch ihre Interaktion aus. Im Extremfall besteht das gesamte System aus isolierten und unabhängigen Komponenten [vgl. Szyperski, 2003].

> *„Software evolution, which is unavoidable in practice, is easiest when system components are largely independent; but integration requires fine-grained coordination of tools, and hence of system components“* [Sullivan und Notkin, 1992, S. 230].

In dieser Arbeit soll die neue und vorhandene Leitstandfunktionalität auf unterschiedlichen Schichten einer Schichtenarchitektur interagieren (siehe Abbildung 8.10). Jede Schicht wartet dabei mit eigenen Herausforderun-

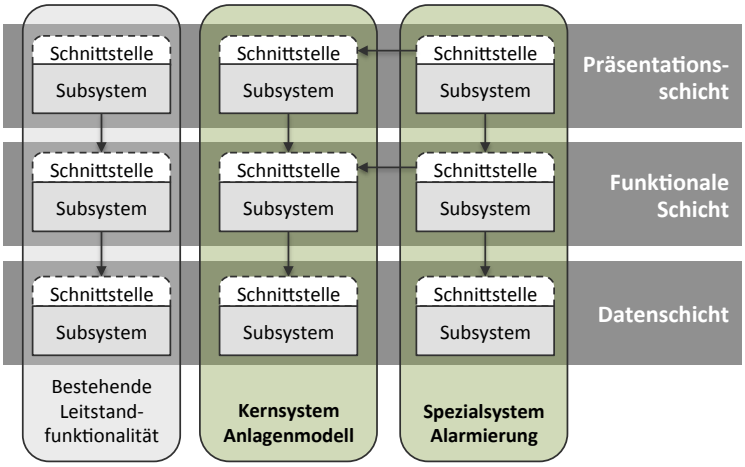


Abbildung 8.10: Neue und vorhandene Leitstandfunktionalität (Quelle: eigene Erstellung)

gen auf und bedarf dazu passender Lösungen. In den folgenden Abschnitten werden Ansätze zur losen Kopplung zwischen Werkzeugen auf der Präsentationsschicht sowie neuer und vorhandener Leitstandfunktionalität auf der

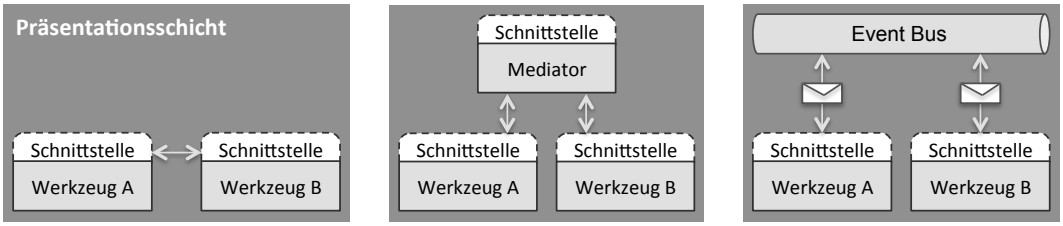
funktionalen Schicht diskutiert. Die Interaktion auf der Datenschicht wird meist vor dem Hintergrund der Skalierbarkeit großer und verteilter Datenmengen geführt und ist nicht im Fokus dieser Arbeit [vgl. z. B. Milliner et al., 1995].

8.3.1 Interaktion auf der Präsentationsschicht

Werkzeuge werden üblicherweise ohne die Absicht entwickelt, sie zukünftig mit anderen Werkzeugen zu kombinieren. Infolgedessen propagieren sie nur den aktuellen Anforderungen genügende interne Ereignisse nach außen und bieten eingeschränkte oder auch gar keine Möglichkeiten, auf neue externe Ereignisse zu reagieren [vgl. Sullivan und Notkin, 1992, S. 250]. In dieser Arbeit sollen beispielsweise das Alarmtabellen-Werkzeug und das Alarmbaum-Werkzeug interagieren und gemeinsam Operateuren die Analyse der Alarmsituationen erleichtern (siehe Kapitel 5). Wählt beispielsweise ein Operateur einen Alarm im Alarmtabellen-Werkzeug aus, soll sich im Alarmbaum-Werkzeug das Anlagenmodell bis zum selektierten Anlagenteil entfalten und den selektierten Alarm in der Anlagenstruktur verorten. Umgekehrt lösen Selektionen eines Anlagenteils im Alarmbaum-Werkzeug einen Filter aus, der im Alarmtabellen-Werkzeug die zum selektierten Anlagenteil zugehörige Alarme filtert. Für diesen Anwendungsfall muss das vorhandenen Alarmtabellen-Werkzeug Selektionen propagieren, auf Selektionen im Alarmbaum-Werkzeug reagieren und Alarme filtern. Das Alarmtabellen-Werkzeug wurde jedoch ursprünglich nicht dafür konzipiert, zukünftig mit anderen Werkzeugen zu interagieren und unterstützt daher nicht die erforderliche Funktionalität.

Entwickler können dieser Problematik durch geeignete Reengineering-Maßnahmen und einer besseren Abstimmung neuer und bestehender Werkzeuge begegnen. Dabei laufen sie jedoch Gefahr, zuvor autonome Werkzeuge zu eng aneinander zu koppeln. Jegliche Bemühungen, Werkzeuge zu entkoppeln, führen wiederum potenziell zu einer verminderten Interaktion zwischen neuer und bestehender Funktionalität [vgl. Hohpe und Woolf, 2004, S. 40]. In der Literatur werden verschiedene Vorschläge zur Werkzeugkopplung und Interaktion diskutiert [vgl. Sullivan und Notkin 1992; Beach 1992; Riehle

und Züllighoven 1995]. Die Abbildung 8.11 präsentiert drei der gängigen Konzepte:



(a) Direkte Verdrahtung (b) Kapselung (c) Nachrichtenaustausch

Abbildung 8.11: Mechanismen zur Werkzeugkopplung (Quelle: eigene Erstellung)

- **Direkte Verdrahtung:** Die Werkzeuge kennen sich untereinander, Zustandsänderungen eines Werkzeugs werden direkt an andere Werkzeuge propagiert und dort verarbeitet. Diese Variante der Werkzeugkopplung führt häufig zu schwer verständlicher und schwer wartbarer Software, da bereits kleine Änderungen in einem Werkzeug meist Auswirkungen auf andere Werkzeuge haben. Die Komplexität dieser Variante erhöht sich mit jedem weiteren Werkzeug. Im schlimmsten Fall muss jedes Werkzeug mit jedem anderen interagieren, was die Wart- und Erweiterbarkeit zusätzlich erschwert.

- **Kapselung:** Die Interaktion zwischen Werkzeugen übernimmt eine separate Komponente, die alle Anfragen und Antworten zwischen den Werkzeugen vermittelt. In Gamma et al. wird dieses Muster als *Vermittler* oder *Mediator* bezeichnet [Gamma et al., 2009]. Bei dieser Variante sind Werkzeuge und ihre Beziehungen sauber getrennt. Werden neue Werkzeuge hinzugefügt und müssen diese mit bestehenden Werkzeugen interagieren, so muss der Vermittler oder Mediator um zusätzliche Koordinationsfunktion erweitert werden. Dadurch nimmt seine Komplexität mit jedem neuen Werkzeug zu, so dass der Mediator langfristig zu einem schwer wartbaren und erweiterbaren Monolithen degeneriert [vgl. Gamma et al., 2009, S. 91].

- **Nachrichtenaustausch:** Statt die Beziehungen zwischen den Werkzeugen explizit zu modellieren, verschicken Werkzeuge Zustandsänderun-

gen über Ereignisse (engl. Events) an einen Event-Bus, ohne die Empfänger der Nachricht direkt zu kennen. Die potentiellen Empfänger können sich wiederum für Ereignisse an dem Event-Bus registrieren, ohne die Absender explizit zu kennen. Bei diesem Ansatz muss der Event-Bus weder die Menge der Sender noch der Empfänger kennen und pflegen. Dadurch können neue Werkzeuge einfacher an den Event-Bus andocken, UI-Aktionen senden und empfangen und dadurch mit bestehenden Werkzeugen interagieren [vgl. Hohpe und Woolf, 2004, S. 137].

Von den vorgestellten Alternativen bietet der Event-Bus-Ansatz die größte Flexibilität einer lose gekoppelten Werkzeuginteraktion und unterstützt das minimalinvasive Vorgehen dieser Arbeit, da vorhandene Werkzeuge im Idealfall ohne Änderungen mit neuen Werkzeugen interagieren können. Die aktuelle Eclipse Plattform implementiert noch den Mediator-Ansatz. Eclipse versucht zwar, der wachsenden Komplexität im Mediator zu entrinnen und die Vermittlung der Ereignisse zwischen Werkzeugen zu standardisieren. Dabei ist lediglich der Austausch der Ereignisse, jedoch nicht deren Inhalt standardisiert. Neue Werkzeuge können dadurch Ereignisse austauschen, ohne dass der Mediator dafür überarbeitet werden muss. Die Eclipse Plattform kann jedoch die wachsende Komplexität mit zunehmenden Werkzeugen nicht gänzlich vermeiden. Da die Beziehung zwischen dem Mediator und den Werkzeugen über das Beobachter-Muster nach Gamma et al. realisiert ist, müssen sich bei diesem Ansatz Werkzeuge an dem Mediator an- und abmelden [Gamma et al., 2009, S. 287ff]. Dies führt in großen Anwendungen zu einer erheblichen Anzahl an Listen, in denen Beobachter und beobachtete Werkzeuge konsequent gepflegt werden müssen. Die Eclipse Gemeinschaft hat die daraus resultierenden Nachteile erkannt und die Umstellung des Nachfolgers auf eine nachrichtenbasierte Kommunikation geplant [Eclipse-Foundation, 2012]. Dieser Nachfolger war zum Zeitpunkt der Entstehung dieser Arbeit nicht verfügbar, weshalb die Beispiele dieser Arbeit auf dem Mediator-Ansatz basieren.

8.3.2 Kommunikation heterogener Systeme auf der funktionalen Schicht

In großen Unternehmen existieren diverse eigenentwickelte oder zugekaufte Anwendungen. Diese werden für gewöhnlich auf unterschiedlichen Plattformen und an diversen Standorten ausgeführt. Viele dieser Anwendungen wurden ursprünglich als Black-Box-Lösungen und ohne die Absicht entwickelt, sie später mit anderen Anwendungen zu verbinden. Sie bieten daher entweder keine geeigneten Schnittstellen für den Austausch der Daten und (Business-)Funktionalität oder die Schnittstellen bieten nur unzureichende Informationen und inkompatible Datenformate. Dies erschwert erheblich die Integration heterogener Systeme [Hohpe und Woolf, 2004, S. 1]. Für den Datenaustausch zwischen heterogenen Systemen existieren aktuell nur wenige Standards wie XML oder Webservices. Ihre zunehmende Popularität hat jedoch dazu geführt, dass immer wieder neue Interpretationen und Erweiterungen erscheinen, die auch diese Standards zunehmend fragmentieren [Hohpe und Woolf, 2004, S. 3]. XML und Webservices bieten lediglich eine technische Infrastruktur für den Datenaustausch. Diese Daten müssen in anderen Systemen auch interpretiert und verstanden werden. Hohpe und Woolf unterscheiden vier grundlegende Ansätze für den Austausch der Daten und (Business-)Funktionalität zwischen Systemen [Hohpe und Woolf, 2004, S. 41ff]:

- **Dateiaustausch:** Jede Anwendung produziert Dateien, die andere Anwendungen in regelmäßigen Intervallen auslesen. Für den Datenaustausch müssen Anwendungen meist die Dateiformate übersetzen und an die Zielformatierung anpassen. Der Dateiaustausch entkoppelt Systeme und erlaubt ihre unabhängige Evolution, vorausgesetzt die Systeme produzieren weiterhin Dateien im gleichen Format. Dies ist jedoch nicht immer möglich, wodurch Änderungen im Dateiformat Auswirkungen auf andere Anwendungen nach sich ziehen. In großen Anwendungen mit vielen Dateien ist es zudem aufwändig, zwischen aktuellen und veralteten Dateien zu unterscheiden.

- **Gemeinsame Datenbank:** Anwendungen nutzen eine gemeinsame Datenbank, um Daten auszutauschen. Datenbanken helfen, große Datenmengen effizient, konsistent und unabhängig von einem Dateiformat zu speichern. Die Daten in einer Datenbank liegen jedoch nicht unstrukturiert vor, sondern entsprechen einem festgelegten Schema. Jede Anwendung stellt unterschiedliche Anforderungen an das Schema, wodurch Datenbankschemas langfristig schwer zu verstehen und zu warten sind.

- **Interprozesskommunikation:** Ermöglicht den Austausch von Daten und (Business-)Funktionalität zwischen Anwendungen, die in unterschiedlichen Adressräumen ablaufen. Dazu stellen Anwendungen ihre Funktionalität anderen Anwendungen in Form von Diensten zur Verfügung. Dienste entkoppeln Anwendungen und ermöglichen jeder Anwendung, Daten intern in unterschiedlichen Formaten oder Schemas vorzuhalten. Die Interprozesskommunikation ist häufig an die verwendete Plattform und Programmiersprache gebunden, was die Interoperabilität zwischen Anwendungen einschränkt. Die Entkopplung von spezifischen Dateiformaten führt bei der Interprozesskommunikation häufig zu einer Kopplung zwischen Diensten, wenn Anwendung für die Bereitstellung ihrer Funktionalität auf andere Dienste angewiesen sind.

- **Nachrichtenaustausch (engl. Messaging):** Anwendungen sind lose über einen Nachrichten-Bus gekoppelt, über den sie Daten austauschen. Der Datenaustausch erfolgt dabei asynchron, so dass Sender und Empfänger den Datenaustausch zu unterschiedlichen Zeitpunkten initialisieren und verarbeiten können. Messaging kombiniert die Vorteile einer losen Kopplung beim Dateiaustausch mit der schnellen Datenverarbeitung einer Datenbank. Es ermöglicht den Austausch von Daten und (Business-)Funktionalität zwischen Anwendungen, ohne dabei die Dienste der Anwendungen zu eng aneinander oder an Plattformen und Programmiersprachen zu koppeln. Die Vorteile einer losen Kopplung mit Messaging gehen einher mit einer höheren Architekturkomplexität bei der Interprozesskommunikation (siehe dazu Abschnitt 8.3.3).

Für die Einführung neuer Leitstandfunktionalität in eine heterogene Systemlandschaft wird ein Ansatz benötigt, der neue und vorhandene Anwendungen möglichst plattform- und programmiersprachenunabhängig verbindet, ohne dabei massiv in die Systeme einzugreifen. Zusätzlich soll der Ansatz in langlebigen Leitstandsystemen die unabhängige Evolution heterogener Systeme erlauben und einen zuverlässigen und schnellen Austausch von Anlageninformationen ermöglichen. Für diese Anforderungen bietet der Nachrichtenaustausch das größte Potential zur Entkopplung der Systeme, Datenformate und Technologien. Neue Leitstandanwendungen können an einem Nachrichten-Bus andocken und die für sie relevanten Informationen abgreifen, ohne dabei bestehende Systeme zu verändern. Die lose Kopplung zwischen der bestehenden und neuen Leitstandfunktionalität erlaubt zudem eine unabhängige Evolution der Systeme. Der Nachrichtenaustausch unterstützt somit eine minimalinvasive Leitstandmodernisierung und wird in dieser Arbeit für die Kommunikation mit heterogenen Systemen verwendet.

Der Nachrichtenaustausch existiert in zwei verschiedenen Varianten. Zum einen unterstützt der Nachrichtenaustausch, wie im vorangegangenen Abschnitt 8.3.1 beschrieben, innerhalb einer Anwendung die Interaktion zwischen Werkzeugen und Subsystemen. Zum anderen existiert eine verteilte Variante, bei der Nachrichten zwischen mehreren Anwendungssystemen ausgetauscht und synchronisiert werden (siehe Abbildung 8.12). Um Be-

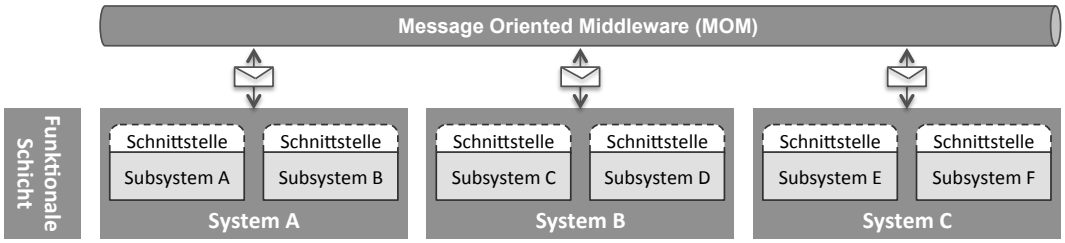


Abbildung 8.12: Verteilte Variante des Nachrichten-Bus-Ansatzes (Quelle: eigene Erstellung)

griffsverwirrungen vorzubeugen, wird in dieser Arbeit zwischen einem lokalen *Event-Bus* und einem verteilten *Messaging-System* unterschieden. Für Messaging-Systeme wird meist eine Message-Oriented-Middleware (MOM) eingesetzt. Diese reduziert die erhöhte Komplexität der verteilten Nachrichtenübermittlung, für die Router, Nachrichtenkanäle, Transformatoren usw. benötigt werden [vgl. Hohpe und Woolf, 2004, S. 14f]. Eine MOM kann die

Komplexität einer verteilten Nachrichtenübermittlung nicht gänzlich vermeiden. Für die Anwendungsentwicklung werden daher Architektur-Maßnahmen benötigt, die die steigende Komplexität der Nachrichtenübermittlung von der Anwendungslogik entkoppeln. Im folgenden Abschnitt wird eine Auswahl dieser Maßnahmen vorgestellt.

8.3.3 Kapselung der Komplexität in Messaging-Systemen

In Messaging-Systemen tauschen heterogene Anwendungen Nachrichten asynchron aus. Die Asynchronität ermöglicht eine *send-and-forget*-Kommunikation. Dabei muss eine Anwendung nicht blockierend darauf warten, bis ein Empfänger die gesendete Nachricht erhalten und verarbeitet hat [Hohpe und Woolf, 2004, S. xxxiv]. Dies erhöht die Performanz und den Durchsatz beider Anwendungen, führt aber auch zu den folgenden Problemen:

- Nicht jede Anwendung kann in einem *send-and-forget*-Modus operieren, wenn beispielsweise die Antwort auf eine Client-Anfrage für den weiteren Programmverlauf unmittelbar erforderlich ist. Für solche Situationen wird ein Adapter zwischen synchronen und asynchronen Lösungen benötigen.
- Der asynchrone Nachrichtenaustausch führt zu einem Programmiermodell, das die Softwareentwicklung und Fehleranalyse deutlich erschwert [Hohpe und Woolf, 2004, S. xxxvi].
- Die Komplexität asynchroner Kommunikation birgt die Gefahr, in die Anwendungslogik einzudringen und dadurch die gesamte Entwicklung client- und serverseitiger Lösungen zu beeinträchtigen.

Zur Eindämmung dieser Gefahr präsentieren Hohpe und Woolf eine reichhaltige Auswahl an Mustern [Hohpe und Woolf, 2004]. Von diesen werden ausgewählte, für diese Arbeit besonders relevante Muster vorgestellt:

- **Message Endpoints:** Bevor Sender und Empfänger Daten über Nachrichten austauschen können, müssen sie sich zunächst am Messaging-System über eine konkrete Remote-Technologie anmelden. Gamma et al.

empfehlen dabei, Anwendungslogik möglichst frei von jeglicher Remote-Technologie zu halten, um sie bei Bedarf einfacher austauschen zu können. Sie führen dafür das Remote-Proxy-Muster ein [Gamma et al., 2009, S. 256]. Ein Remote-Proxy ist ein lokaler Stellvertreter im Client, für ein Objekt im Adressraum des Servers. Hohpe und Woolf realisieren dieses Muster in Messaging-Systemen über sog. *Message Endpoints* [Hohpe und Woolf, 2004, S. 95ff]. Message Endpoints kapseln den Auf- und Abbau der Verbindung zum Messaging-System sowie den Versand und Empfang von Nachrichten, so dass Sender und Empfänger die dahinterstehende Technologie nicht kennen müssen (Abbildung 8.13).

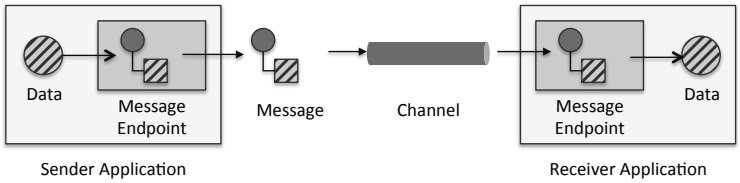


Abbildung 8.13: Message Endpoints (Quelle: Hohpe und Woolf, 2004, S. 95ff)

- **Messaging Gateway:** In einem Messaging-System müssen Sender und Empfänger dedizierte Methoden zum Senden und Empfangen von Nachrichten anbieten. Hohpe und Woolf empfehlen jedoch, dass Anwendungen generell nicht wissen sollten, dass sie über Nachrichten kommunizieren [Hohpe und Woolf, 2004, S. 63]. Das liegt vor allem an der asynchronen Kommunikation, bei der die Ergebnisse für Operationsaufrufe zu einem späteren Zeitpunkt eintreffen und aufwändig dem ursprünglichen Operationsaufruf zugeordnet werden müssen. Dieser Programmierstil ist sehr anspruchsvoll und führt daher zu vielen Fehlern [Hohpe und Woolf, 2004, S. xxxvi]. Hohpe und Woolf empfehlen, die zuvor eingeführten *Message Endpoints* in Kombination mit einem *Messaging Gateway* zu verwenden [Hohpe und Woolf, 2004, S. 97]. Ein Messaging Gateway übersetzt Operatiosaufrufe einer Anwendung in eine oder mehrere Nachrichten, sendet sie an den Empfänger und übersetzt dort die Nachrichten wieder in Operationsaufrufe. Dieser Ansatz reduziert die Komplexität der asynchronen Kommunikation, da

Anwendungen nicht wissen, dass sie über Nachrichten kommunizieren (siehe Abbildung 8.14).

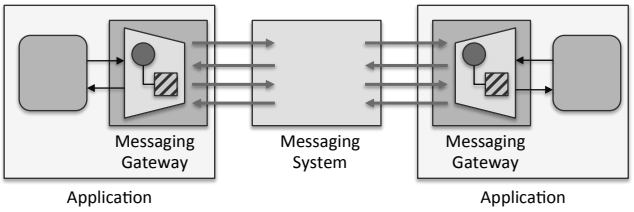


Abbildung 8.14: Messaging Gateway (Quelle: Hohpe und Woolf, 2004, S. 468ff)

Zur Eindämmung der wachsenden Komplexität in nachrichtenbasierten Systemen werden die vorgestellten Muster in dieser Arbeit gemeinsam in einer Abstraktionsschicht realisiert, die bei Bedarf mit weiteren Mustern ergänzt werden kann. Diese Schicht kapselt sämtliche für die Kommunikation benötigte Funktionalität und verbirgt die damit einhergehende Komplexität vor der Anwendungslogik. Nock bezeichnet diese Schicht als Datenzugriffsschicht (engl. Data Access Layer, DAL) [vgl. Nock, 2004, S. 9]. Die Datenzugriffsschicht erleichtert Softwareentwicklern die Programmierung und die Fehleranalyse ihrer Anwendungen und fördert die Einfachheit einer Architektur (siehe Abschnitt 6.1.2). In der Schichtenarchitektur der Abbildung 8.2 kann die Datenzugriffsschicht einerseits zwischen der Präsentations- und der funktionalen Schicht, zur Entkopplung der Anwendungslogik und der verwendeten Remote-Technologie, eingeordnet werden. Andererseits kann die Datenzugriffsschicht zwischen der funktionalen und der Datenschicht, zur Kapselung der Persistenz mit einer konkreten Datenbanktechnologie, eingeordnet werden [vgl. Nock, 2004].

8.4 Zusammenfassung

In diesem Kapitel wurde die Metapher *Modernisierung der Altstadt* konkretisiert und die Modernisierung einer vorhandenen Leitstandanwendung aus technischer Sicht beleuchtet. Zu Beginn dieses Kapitels wurde eine Gesamtarchitektur zur Konstruktion (meta-)modellbasierter Anwendungen vorgestellt,

die die klassische Drei-Schichten-Architektur um fachliche Schnitte und Subsysteme erweitert. Diese Gesamtarchitektur hilft, neue Leitstandkomponenten in der Architektur zu verorten, ihre Beziehungen und Zuständigkeiten zu definieren und unterstützt dadurch eine strukturierte und organisierte Leitstandmodernisierung. Für eine minimalinvasive Einführung neuer Subsysteme in vorhandene Leitstandanwendungen wurden im weiteren Verlauf des Kapitels modulare Ansätze vorgestellt und am Beispiel der Plug-in-Technologie diskutiert. Die Kombination einer (meta-)modellbasierten Gesamtarchitektur mit den Plug-in-Konzepten hilft, eine Anwendung in möglichst autonome und kohärente Plug-ins zu schneiden, die Änderungen und Erweiterungen mit lokalen Auswirkungen unterstützen. Diese Konzepte müssen vor dem Hintergrund langlebiger Leitstandsysteme durch weitere Architekturansätze ergänzt werden, die eine langfristige Evolutionsfähigkeit der neuen Leitstandfunktionalität unterstützen. Dazu wurde in diesem Kapitel mit dem Dekorierer-Muster ein Ansatz zur Wiederverwendung der Werkzeugfunktionalität auf der Präsentationsschicht vorgestellt. Zur Unterstützung der kooperativen Arbeit der Operateure wurde in diesem Kapitel die explizite Modellierung der Anlagenzustände am Beispiel der Alarmierung motiviert und generische Erweiterungskonzepte für Anlagenmodelle auf der funktionalen Schicht konzipiert. Zusammengenommen beantworten diese Teilaspekte die Forschungsfrage, wie neue Leitstandkomponenten mit wenigen Eingriffen und Seiteneffekten in vorhandene Systeme eingeführt werden können, um dadurch die Aufwände und Risiken einer Leitstandmodernisierung zu reduzieren.

Damit die neu eingeführte Leitstandfunktionalität nicht isoliert fortbesteht, sondern komplementär die vorhandene Funktionalität ergänzt, muss sie mit den bestehenden, meist heterogenen Systemen kommunizieren und interagieren. In dieser Arbeit wurden verschiedene Ansätze zur lose gekoppelten Verknüpfung zwischen Werkzeugen auf der Präsentationsschicht und verteilten heterogenen Systemen auf der funktionalen Schicht vorgestellt und mit dem Nachrichtenaustausch ein für beide Schichten geeigneter Ansatz zur Interaktion identifiziert[1]. Messaging-Konzepte bieten, neben einer losen Kopplung zwischen Werkzeugen und Systemen, auch die Möglichkeit,

[1] Da die aktuelle Eclipse-RCP-Version keinen Nachrichtenaustausch unterstützt, wird in dieser Arbeit auf der Präsentationsschicht der Mediator-Ansatz verwendet.

minimalinvasiv an vorhandene Funktionalität anzudocken und mit dieser Informationen auszutauschen. Sie führen aber auch zu einer höheren Architekturkomplexität, die durch eine Datenzugriffsschicht eingedämmt werden kann, um die Einfachheit und Analysierbarkeit eines Leitstandsystems zu erhöhen. Diese Ansätze beantworten zusammengenommen die Forschungsfrage, wie neue und vorhandene Leitstandkomponenten mit wenigen Eingriffen miteinander verknüpft und aufeinander abgestimmt werden können.

Die in diesem Kapitel vorgestellten Ansätze zur minimalinvasiven Einführung neuer Leitstandfunktionalität und ihre Verknüpfung mit der vorhandenen Funktionalität sind in der Abbildung 8.15 im WAM-Dimensionenmodell verortet. Der Schwerpunkt dieses Kapitels liegt auf softwaretechnischen Konstruktionsprinzipien, die der Dimension *verwendete Technik* zuzuordnen sind. Die Auswahl der Konstruktionsprinzipien orientieren sich an dem Vorhaben dieser Arbeit, vorhandene Systeme möglichst zu erhalten und mit wenigen Eingriffen in Systeme und Arbeitsabläufe zu modernisieren. Diese Einflüsse entstammen den Dimensionen *Anwendungsbereich* und *Handhabung & Präsentation*. Sie geben einen Orientierungsrahmen für die technische Realisierung vor.

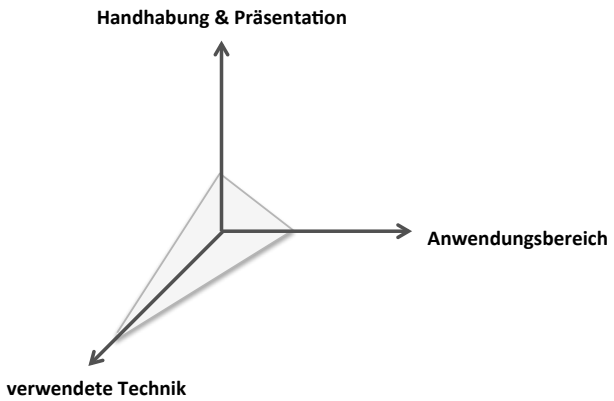


Abbildung 8.15: Einordnung minimalinvasive Leitstandmodernisierung aus technischer Sicht im WAM-Dimensionenmodell (Quelle: eigene Erstellung)

Im nächsten Kapitel wird die anwenderorientierte und softwaretechnische Sichtweise der Leitstandmodernisierung zusammengeführt und die Kernaspekte der vorangegangenen Kapitel zunächst zusammengefasst, bevor ihre Realisierung am Beispiel der *Alarmierung* demonstriert wird.

Kapitel 9

Kurzvorstellung Realisierung

Die Entwurfsmetapher *Modernisierung der Altstadt* gibt die Orientierung für diese Arbeit vor. Statt bestehende Systeme abzulösen oder massiv zu verändern, sollen sie durch minimalinvasive Eingriffe schrittweise modernisiert werden. Dabei dient die anwenderorientierte Sichtweise in dieser Arbeit als Bezugspunkt, um die vielseitigen, oft technologiezentrierten Lösungsalternativen, stets an ihrem Nutzen für die Anwender zu bewerten. Die softwaretechnische Sichtweise ermöglicht eine minimalinvasive Leitstandmodernisierung auf der technischen Ebene und hilft, alternative Realisierungen auf das technisch Machbare einzugrenzen. In diesem Abschnitt werden die anwenderorientierte und softwaretechnische Sicht zusammengeführt, um den Lösungsansatz dieser Arbeit zu verdeutlichen. Dieser Lösungsansatz orientiert sich an den folgenden Kernaspekten:

- In Kapitel 3 wurde die anwenderorientierte Sicht herangezogen, um systematisch aus Sicht der Operateure aktuelle Leitstandprobleme aufzuzeigen und dazu passende Lösungsansätze zu erarbeiten. Diese Lösungsansätze benötigen für ihre Realisierung ein Anlagenmodell, das in Kapitel 4 eingeführt wurde. In Kapitel 5 wurden die anwenderorientierten Lösungsansätze am Beispiel der Leitstandaufgabe *Alarmierung* auf Anlagenmodelle übertragen und anhand eines modellbasierten Alarmbaum-Werkzeugs der Ansatz dieser Arbeit skizziert.

- Die anwenderorientierte Sichtweise dient in dieser Arbeit ferner dazu, Operateure behutsam an neue Leitstandfunktionalität heranzuführen, um ihren Lernaufwand und die damit verbundenen Risiken durch Fehlbedienung der Anlage zu reduzieren. Dazu wurden in Kapitel 7 Konzepte erarbeitet, die neue Leitstandfunktionalität sukzessive einführen, beginnend mit den Bereichen, in denen Operateure den größten Nutzen erwarten. Zu diesem Zweck wird die neue Leitstandfunktionalität im Rahmen einer evolutionären Vorgehensweise als Softwareerweiterung eingeführt. Die Entwurfsmetapher *Modernisierung der Altstadt* verdeutlicht diesen Ansatz.

- Leitstandsysteme bilden oft das Rückgrat für Anlagenbetreiber und konzentrieren ein umfangreiches, über Jahre angesammeltes und bewährtes Wissen. Jegliche Modifikationen dieser Systeme erhöhen das Ausfallrisiko technischer Anlagen. Statt bestehende Leitstandsysteme abzulösen oder tiefgreifend zu verändern, werden sie in dieser Arbeit erhalten und modernisiert. Dazu wurden in Kapitel 8 architektonische und technologische Ansätze zur Einführung neuer Leitstandfunktionalität in vorhandene Systeme und ihre Verknüpfung mit vorhandener Funktionalität diskutiert. Diese Ansätze konkretisieren die Entwurfsmetapher *Modernisierung der Altstadt* auf einer softwaretechnischen Ebene.

Die vorgestellten Aspekte sind in der Abbildung 9.1 auf einer abstrakten Ebene visualisiert. Die neue serverseitige Leitstandfunktionalität ist in der Abbildung 9.1 von den klassischen Servern entkoppelt, um ihre Unantastbarkeit und Souveränität zu wahren. Neue und vorhandene Leitstandfunktionalität sind über das Messaging-System lose gekoppelt und können dadurch Dienste und Daten austauschen. Clientseitig wird die vorhandene Leitstandsoftware systematisch durch neue Werkzeuge für die elementaren Leitstandaufgaben ergänzt, so dass neue und vorhanden Werkzeuge besser auf die Arbeit der Operateure abgestimmt sind. Für Operateure verhält sich die modernisierte Leitstandsoftware wie ein homogenes System mit neuen und verbesserten Werkzeugen. Die technische Fragmentierung aus neuer und vorhandener Funktionalität bleibt für sie verborgen.

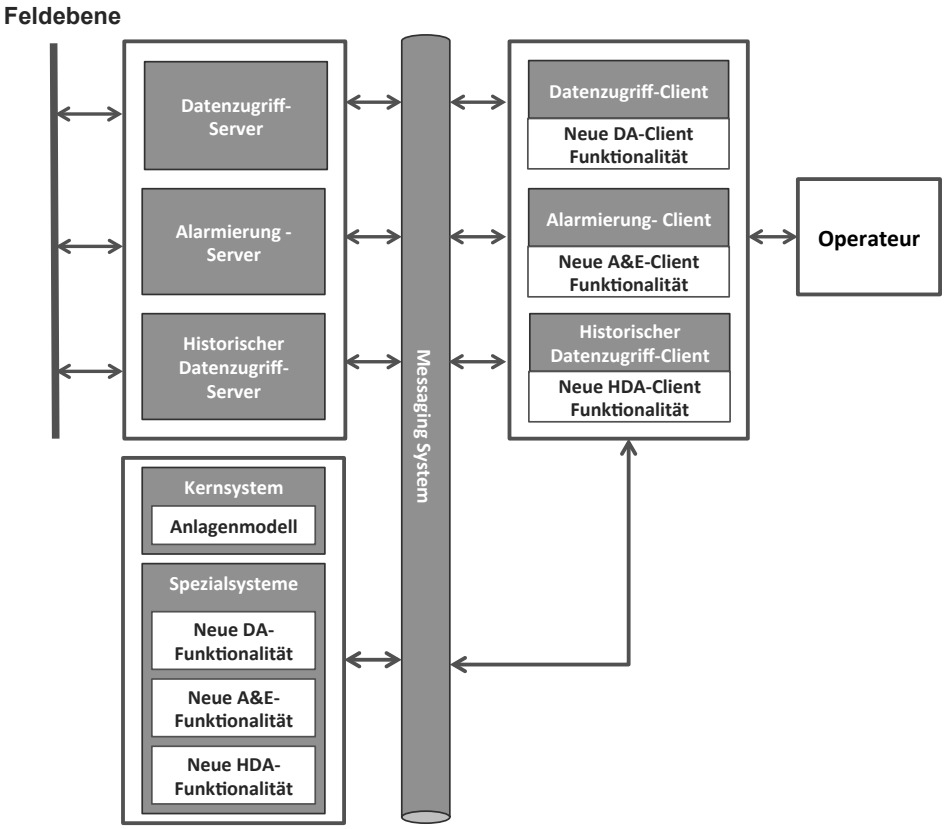


Abbildung 9.1: Minimalinvasive Einführung neuer Leitstandfunktionalität (Quelle: eigene Erstellung)

In den vorangegangenen Kapiteln wurden aus einer anwenderorientierten und technischen Perspektive die Risiken und Aufwände einer Leitstandmodernisierung beleuchtet, Alternativen diskutiert und Vorschläge präsentiert, wie bestehende Systeme nach den Prinzipien der Softwaretechnik erhalten und unter ökonomisch vertretbaren Bedingungen modernisiert werden können. Im folgenden Kapitel werden diese Vorschläge auf das DESY-Leitstandsystem übertragen und am Beispiel der Leitstandaufgabe *Alarmierung* konstruktiv umgesetzt. Dabei wird die duale Sichtweise aus anwenderorientierter und technischer Sicht beibehalten, um weiterhin die Einführung neuer Leitstandfunktionalität aus verschiedenen Perspektiven zu betrachten und Vor- und Nachteile der gewählten Realisierung zu diskutieren.

Kapitel 10

Realisierung modellbasierter Leitstandfunktionalität

In diesem Kapitel wird aufbauend auf der konzeptionellen Vorarbeit der vorangegangenen Kapitel die Realisierung einer anwenderorientierten und minimalinvasiven Leitstandmodernisierung demonstriert. Am Beispiel des Kernsystems Anlagenmodell und des Spezialsystems Alarmierung wird die Metapher *Modernisierung der Altstadt* verdeutlicht und gezeigt, wie bestehende Leitstandsysteme erhalten und gleichzeitig mit neuer Funktionalität ergänzt werden können. Dazu werden die Ansätze dieser Arbeit auf das DESY-Leitstandsystem übertragen und realisiert. Das DESY-System ist nach einer Client/Server-Leitstandarchitektur aufgebaut, wie sie in Abschnitt 2.3 vorgestellt wurde. Im Bereich Alarmierung überwacht ein Alarmserver Grenzwertüberschreitungen auf der Feldebene und versendet diese an ein Alarmmanagementsystem (AMS). Dort werden sie unter Zuhilfenahme komplexer Regelwerke gefiltert, aggregiert und schließlich an Clients verschickt. Diese Zusammenhänge sind in der Abbildung 10.1 visualisiert.

Wie in Abschnitt 3.2 herausgearbeitet, können unnötige Alarme und Alarmfluten trotz eines ausgeklügelten Alarmmanagementsystems nicht gänzlich vermieden werden. Operateure benötigen daher verbesserte Alarm-Werkzeuge, die sie in solchen Situationen unterstützen, die Anlagenkomplexität zu beherrschen. In Abschnitt 5 wurde als Fundament für neue anwenderori-

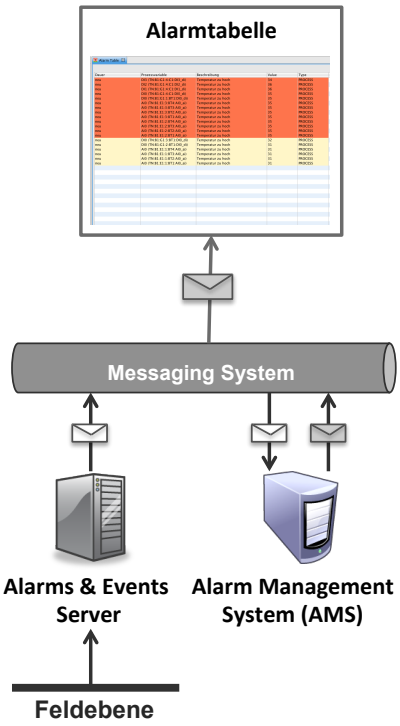


Abbildung 10.1: DESY Alarmsystem-Architektur (Quelle: eigene Erstellung)

entierte Werkzeuge ein explizites Anlagenmodell herausgearbeitet, das in vielen Leitstandsystemen gänzlich fehlt oder nur unzureichende Eigenschaften aufweist. In den nächsten Abschnitten werden daher minimalinvasive Konzepte zur Einführung des Kernsystems *Anlagenmodell* vorgestellt und darauf aufbauend das Spezialsystem *Alarmierung* eingeführt.

10.1 Einführung des Kernsystems Anlagenmodell

Anlagenmodelle bieten Operateuren die Möglichkeit, in der Anlage zu navigieren und beispielsweise die komplexen Zusammenhänge zwischen Anlagenteilen zu verstehen (siehe Abschnitt 3.4). In diesem Abschnitt werden die Architekturkonzepte zur Konstruktion modellbasierter und modularer Anwendungen aus Abschnitt 8.1 auf den Anwendungsfall *Anlagenmodell* übertragen und konkretisiert. Anschließend werden Ansätze zur minimalinvasiven Softwareerweiterung aus Abschnitt 8.2 herangezogen, um die neue Leitstandfunktionalität in ein vorhandenes System einzuführen. Dieses Vorgehen wird zunächst aus einer fachlichen, von den technischen Details abstrahierenden Architektursicht vorgestellt, bevor die entsprechende technische Realisierung diskutiert wird.

10.1.1 Fachliche Architektursicht

In Abschnitt 8.1 wurde eine Gesamtarchitektur für Leitstandanwendungen vorgestellt, die die klassische Drei-Schichten-Architektur um Subsysteme, ihre Beziehungen und fachliche Schnitte erweitert. Bevor diese Architektur auf den Anwendungsfall *Anlagenmodell* übertragen wird, soll dieser Anwendungsfall zunächst aus einer fachlichen Architektursicht betrachtet werden. Die Abbildung 10.2 unterteilt den Anwendungsfall *Anlagenmodell* in Werkzeuge, Materialien und Services:

- **Werkzeug:** Das Werkzeug *Anlagenmodell* visualisiert die hierarchischen Beziehungen zwischen Anlagenteilen und ermöglicht Operateuren, sich in komplexen technischen Anlagen zurechtzufinden.
- **Materialien:** Das Anlagenmodell-Material kapselt fachliche Konzepte wie z. B. die strukturierte Abbildung komplexer Zusammenhänge technischer Anlagen, Suche nach Anlagenteilen und zugehöriger Metainformationen. Das Anlagenmodell-Material greift hierfür auf weitere Materialien wie Anlagenteile und Prozessvariablen zu.

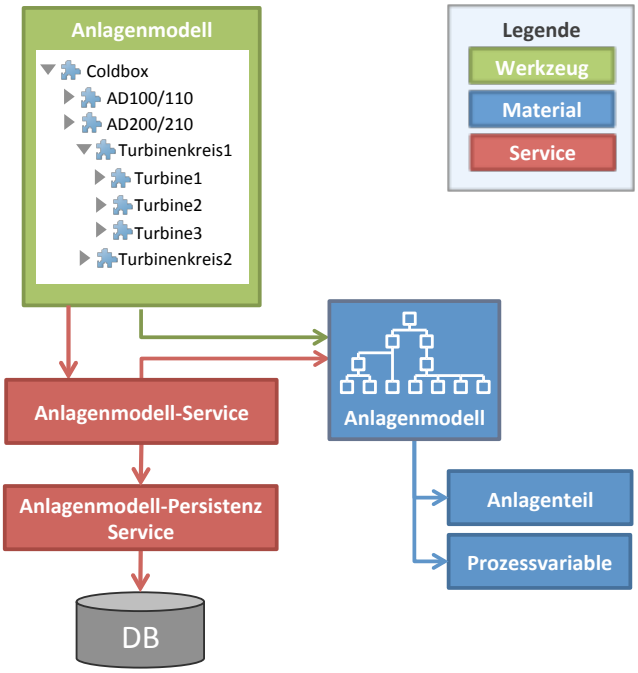


Abbildung 10.2: Fachliche Struktur der Anlagenmodell-Funktionalität (Quelle: eigene Erstellung)

- **Services:** Der Anlagenmodell-Service koordiniert die Erzeugung und Manipulation der Modelle und benachrichtigt Leitstand-Clients bei Modelländerungen. Der Anlagenmodell-Service nutzt einen Persistenz-Service, um Modelle aus der Datenbank zu laden und Modelländerungen zu persistieren.

Für die technische Umsetzung der fachlichen Architektur wird sie in den folgenden Abschnitten entsprechend der Architektur für modulare und modellbasierte Anwendungen aus Abschnitt 8.1 in Subsysteme unterteilt und am Beispiel der Eclipse Plug-in-Technologie aus Abschnitt 8.2 realisiert. Aus Platz- und Übersichtsgründen werden dabei nur die wesentlichen Architekturelemente, ihre Beziehungen und Quellcode-Auszüge präsentiert.

10.1.2 Technische Realisierung auf der Präsentationsschicht

Der Anwendungsfall *Anlagenmodell* bildet einen fachlichen Schnitt entlang aller Schichten einer Schichtenarchitektur. Mit Fokus auf die Präsentations-

schicht kann das Anlagenmodell, entsprechend der Architektur aus Abschnitt 8.1, in folgende Plug-ins aufgeteilt werden (siehe Abbildung 10.3):

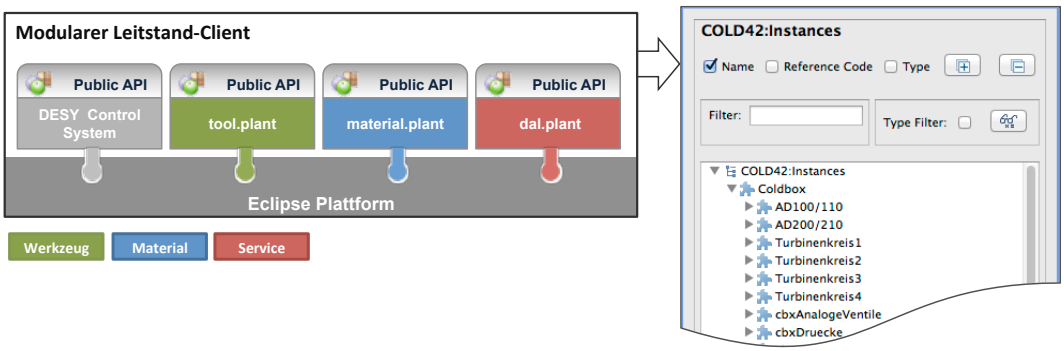


Abbildung 10.3: Plug-in-basiertes Anlagenmodell-Werkzeug (Quelle: eigene Erstellung)

- **DESY Control System:** Repräsentiert aus Platzgründen stellvertretend alle clientseitigen Leitstandkomponenten des DESY-Leitstandsystems (siehe Anhang A.2).
- **de.geneal.desy.tool.plant:** Enthält das Anlagenmodell-Werkzeug und dazugehörige Klassen.
- **de.geneal.desy.material.plant:** Enthält Materialien wie das Anlagenmodell, Anlagenteile und Prozessvariablen sowie dazugehörige Service-Schnittstellen. Werkzeuge und Services nutzen dieses Plug-in als Bibliothek (siehe Abschnitt 8.2).
- **de.geneal.desy.dal.plant:** Wie in Abschnitt 8.3.3 beschrieben, kapselt das DAL-Plug-in die verteilte nachrichtenbasierte Kommunikation mit dem Anlagenmodell-Service.

Die Einführung dieser Funktionalität in das vorhandene DESY-Leitstandsystem erfolgt als zusätzliches Werkzeug im Rahmen einer Softwareerweiterung (siehe Abschnitt 7.3). In Abschnitt 8.2 wurden für die minimalinvasive Einführung neuer Werkzeuge die Konzepte Erweiterungen und Erweiterungspunkte vorgestellt. Dieses Konzept wird in Listing 10.1 exemplarisch für das Anlagenmodell-Werkzeug demonstriert. Erweiterungen und Erweiterungspunkte werden in Eclipse über eine XML-Datei, die sog. *Plug-in.xml* zusammengeführt. Im Listing 10.1 definiert das Anlagenmodell-Werkzeug mit

der implementierenden Klasse `PlantModelView` eine Erweiterung für den Standard-Eclipse-Erweiterungspunkt `org.eclipse.ui.views`. Diese Erweiterung wird beim Start der Leitstandanwendung von der Eclipse-Plattform ausgelesen und das Anlagenmodell-Werkzeug in die grafische Darstellung der Anwendung eingebunden. Dafür mussten die DESY-Entwickler während der initialen Entwicklung keine passenden Schnittstellen antizipieren.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?eclipse version="3.4"?>
3  <plugin>
4     <extension
5           point="org.eclipse.ui.views">
6        <view
7              allowMultiple="true"
8              class="de.c1wps.geneal.desy.ui.plant.PlantModelView"
9              icon="images/tree_mode.gif"
10             id="de.c1wps.geneal.desy.ui.plant.PlantModelView"
11             name="Plant Model View"
12             restorable="true">
13       </view>
14     </extension>
15     ...
16 </plugin>
```

Listing 10.1: Anlagenmodell-Werkzeug Erweiterung (Quelle: eigene Erstellung)

Da das Anlagenmodell-Werkzeug im ersten Schritt nicht mit bestehenden Werkzeugen interagieren muss, kann die Werkzeugeinführung auf der Präsentationsschicht ohne Eingriffe in das vorhandene System erfolgen. Das Anlagenmodell-Werkzeug benötigt für die Visualisierung ein Anlagenmodell, das zuvor eine serverseitige Plattform bereitstellen muss. Die Realisierung dieser Plattform wird im nächsten Abschnitt vorgestellt.

10.1.3 Technische Realisierung auf der funktionalen Schicht

Für die technische Realisierung der fachlichen Architektur aus Abbildung 10.2 wird die Gesamtarchitektur aus Abschnitt 8.1 auf die funktionale Schicht übertragen und in folgende Plug-ins unterteilt:

- **de.geneal.desy.service.plant:** Enthält die Implementierung für den Anlagenmodell-Service. Dieser verwaltet den Lebenszyklus der Anlagenmodelle und informiert Clients über ihre Änderungen.

- **de.geneal.desy.material.plant:** In einer Client/Server-Leitstandarchitektur arbeiten sowohl client- als auch serverseitige Anwendungen auf Materialien, wodurch das entsprechende Plug-in in beiden Schichten benötigt wird.
- **de.geneal.desy.dao.plant:** Kapselt den Persistenz-Service, über den Anlagenmodelle für ihre weitere Verwendung aus der Datenbank geladen und nach Änderungen dort wieder persistiert werden.
- **de.geneal.desy.dal.plant:** Stellt Clients fachliche Dienste über eine nachrichtenbasierte Remote-Technologie zur Verfügung.

Während beim DESY die clientseitige Leitstandsoftware auf einer modularen und erweiterbaren Plattform basiert, sind serverseitige Erweiterungen vorhandener klassischer Server deutlich aufwändiger und teilweise gänzlich ausgeschlossen. Dies trifft insbesondere auf Systeme zu, die als Black-Box-Lösung von Fremdanbietern zugekauft wurden (siehe Abschnitt 8.3.2). Statt vorhandene Server zu manipulieren, wird in dieser Arbeit die neue serverseitige Funktionalität auf einer neuen modularen Plattform neben den vorhandenen klassischen Servern eingeführt (siehe Abbildung 10.4). Die neue Anlagenmo-

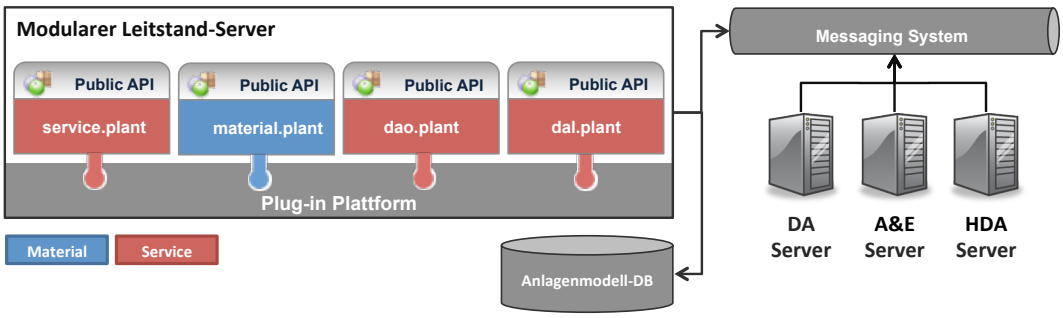


Abbildung 10.4: Plug-in-basierte Anlagenmodell-Funktionalität auf der funktionalen Schicht (Quelle: eigene Erstellung)

dell-Funktionalität greift über das Messaging-System auf Daten und Dienste der klassischen Server zu und stellt neue Dienste bereit, die Clients über die bestehende Messaging-Infrastruktur nutzen können. Über diese Dienste können Clients Anlagenmodelle laden, bearbeiten und anschließend in der Datenbank persistieren. Die modulare Eigenschaft der neuen Plattform bietet

das Potential nicht nur aktuelle, sondern auch zukünftige Anforderungen an neue Leitstandfunktionalität zu unterstützen.

10.1.4 Zusammenfassung

In diesem Abschnitt wurde die (meta-)modellbasierte und modulare Gesamtarchitektur aus Abschnitt 8.1 auf den Anwendungsfall *Anlagenmodell* übertragen und die daraus resultierenden Subsysteme und ihre Beziehungen vorgestellt. Die neuen Subsysteme wurden clientseitig mit Hilfe der Plug-in-Erweiterungsmechanismen aus Abschnitt 8.2 minimalinvasiv in die bestehende DESY-Leitstandanwendung eingeführt. Serverseitig wurde die Anlagenmodell-Funktionalität auf einer neuen modularen Plattform bereitgestellt. Wie in Abschnitt 7.1 diskutiert, bildet das Anlagenmodell ein Fundament für die Entwicklung darauf aufbauender Spezialsysteme. Daher kann die Einführung des Anlagenmodells relativ losgelöst von der bestehenden Leitstandfunktionalität erfolgen. In den nächsten Abschnitten wird am Beispiel des Spezialsystems Alarmierung demonstriert, wie sich die minimalinvasive Vorgehensweise speziell auf die Verknüpfung vorhandener und neuer Leitstandfunktionalität auswirkt.

10.2 Einführung des Spezialsystems Alarmierung

In Kapitel 5 wurde ein Mock-up mit den Werkzeugen Alarmbaum und Prozessbildauflister als Ergänzung zum vorhandenen Alarmtabellen-Werkzeug vorgestellt. In diesem Kapitel werden die Ansätze dieser Arbeit exemplarisch am Beispiel des Alarmbaum-Werkzeugs vorgestellt. Dieses Werkzeug wird mit Hilfe des Dekorierer-Musters aus Abschnitt 8.2.1 realisiert und minimalinvasiv mit den Plug-in-Konzepten aus Abschnitt 8.2 in das vorhandene System eingeführt. Die Alarm-Werkzeuge benötigen ein Alarmmodell, für dessen Realisierung die Modellierungskonzepte aus Abschnitt 8.2.2 aufgegriffen werden. Für die Verknüpfung neuer und vorhandener Alarmfunktionalität werden Ansätze zur losen Kopplung aus Abschnitt 8.3 herangezogen.

10.2.1 Fachliche Architektursicht

Die fachliche Architektur im Kontext der Alarmierung besteht aus den folgenden Werkzeugen, Materialien und Services (siehe Abbildung 10.5).

- **Werkzeuge:** Das Alarmbaum-Werkzeug visualisiert Alarme in einer hierarchischen Darstellung und hilft dadurch Operateuren, den Ort ihrer Entstehung schneller zu erfassen. Das Alarmbaum-Werkzeug ergänzt die Alarmtabelle, die Detailinformationen zu einem Alarm präsentiert. Beide Werkzeuge beziehen den aktuellen Alarmzustand aus dem Alarmmodell.

- **Materialien:** Das Alarmmodell spiegelt alle in einer technischen Anlage aufgetretenen Alarme sowie den Status ihrer Bearbeitung zur Laufzeit wider (siehe Abschnitt 8.2.2).

- **Services:** Der Alarmservice bietet eine Schnittstelle mit sondierenden und manipulierenden Operationen für das Alarmmodell. Er aktualisiert den Zustand des Alarmmodells beim Eintreffen neuer Alarme sowie bei ihrer Aufhebung und informiert darüber alle Leitstand-Clients.

Während das Anlagenmodell client- und serverseitig ohne Veränderungen bestehender Leitstandfunktionalität eingeführt wurde, ist diese Aufgabe bei

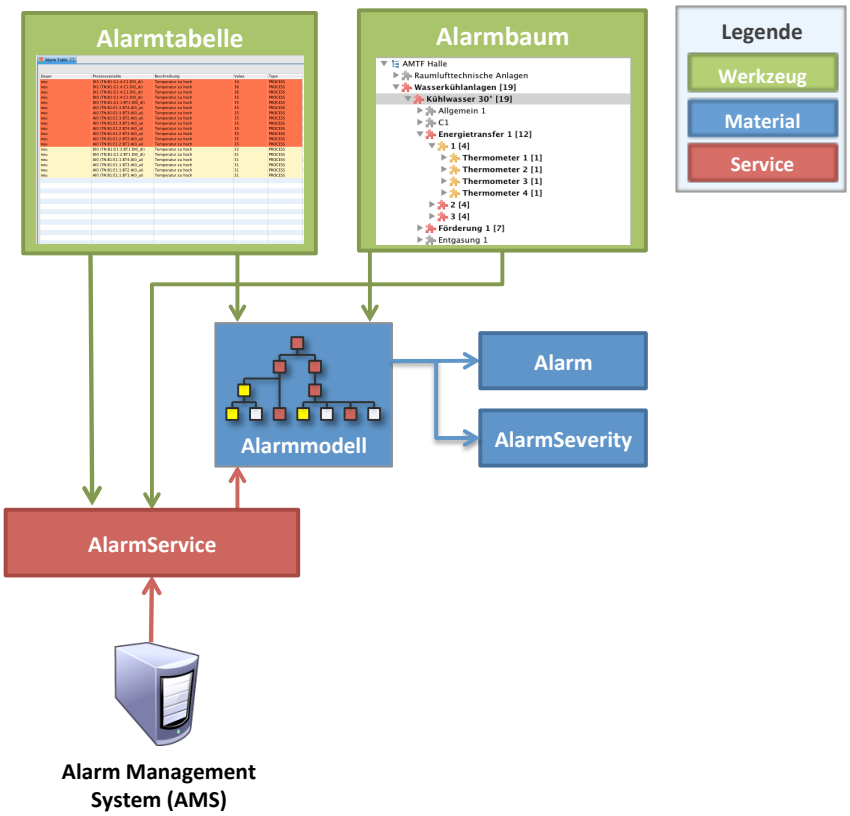


Abbildung 10.5: Fachliche Struktur der Alarm-Funktionalität (Quelle: eigene Erstellung)

der Alarmierung deutlich anspruchsvoller. Dazu müssen bestehende Alarm-Werkzeuge, wie die Alarmtabelle, mit dem Alarmbaum-Werkzeug gekoppelt werden, damit sich diese gegenseitig ergänzen. Serverseitig müssen Alarme über das Messaging-System bezogen und in einem Alarmmodell repräsentiert werden. Schließlich müssen diese Schritte mit möglichst wenigen Eingriffen in bestehende Systeme erfolgen. Die Umsetzung dieser Schritte wird in den folgenden Abschnitten vorgestellt.

10.2.2 Technische Realisierung auf der Präsentationsschicht

Die Übertragung der Gesamtarchitektur aus Abschnitt 8.1 auf den Anwendungsfall *Alarmierung* ist in der Abbildung 10.6 dargestellt. Die Einführung dieser Plug-ins in die vorhandene DESY-Leitstandanwendung erfolgt mit den gleichen Plug-in-Erweiterungskonzepten, wie in Abschnitt 10.1.2 am Beispiel

des Kernsystems *Anlagenmodell* vorgestellt. Aus Platzgründen werden die zuvor eingeführten Anlagenmodell-Plug-ins zusammengefasst:

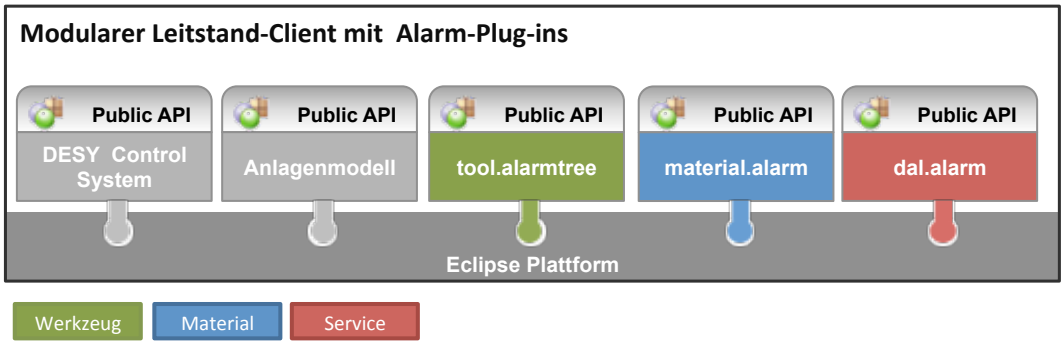


Abbildung 10.6: Plug-in-basiertes Alarm-Werkzeug (Quelle: eigene Erstellung)

- **de.geneal.desy.tool.alarmtree:** Enthält das Alarmbaum-Werkzeug und den Alarm-Dekorierer, der das Anlagenmodell-Werkzeug einschließt und mit alarmspezifischer Darstellung ergänzt.
- **de.geneal.desy.material.alarm:** Enthält Materialien wie Alarmmodell und Alarm.
- **de.geneal.desy.dal.alarm:** Kapselt die Kommunikation mit dem Messaging-System, um beispielsweise Alarme zu empfangen oder Anlagenteile stumm zu schalten.

Die Visualisierung der Alarme in einer hierarchischen Anlagenstruktur und die alarmspezifischen Berechnungen zu jedem Anlagenteil werden softwaretechnisch wie folgt realisiert:

- **Alarme im Alarmbaum-Werkzeug visualisieren:** Für die Visualisierung der Alarme im Alarmbaum-Werkzeug verwendet dieses das Anlagenmodell-Werkzeug wieder und dekoriert die grafische Darstellung mit Hilfe des Dekorierer-Musters aus Abschnitt 8.2.1. Der `AlarmDecorator` bietet Operationen wie `decorateImage()` und `decorateText()`, die für jedes Anlagenteil die Anzahl darin aufgetretener Alarme, ihren höchsten Schweregrad und stummgeschaltete Anlagenteile visualisieren. Für diese Funktionalität greift der Alarm-Dekorierer auf alarmspezifische Funktionalität des Alarmmodells und dazugehöriger Alarm-Knoten zu (siehe Listing 10.2).

```
1  public class AlarmDecorator implements ILabelDecorator{
2     @Override
3     public Image decorateImage(Image image, Object element) {
4        AlarmNode node = (AlarmNode) element;
5        if (node.isInMaintenance()) {
6           return createDecoratedImage(image, AlarmImages.MUTE);
7        } else if (node.hasAlarms()) {
8           return createAlarmImage(node);
9        } else {
10          return createDecoratedImage(image, null);
11       }
12    }
13    @Override
14    public String decorateText(String text, Object element) {
15       AlarmNode node = (AlarmNode) element;
16       if (node.hasAlarms() && !node.isInMaintenance()) {
17          // decorate original plant unit name bold font + alarm quantity
18          text = String.format("%s [%d]", text, node.getAlarmQuantity());
19       }
20       return text;
21    }
22 }
```

Listing 10.2: Alarm-Dekorierer (Quelle: eigene Erstellung)

- **Alarme empfangen und Darstellung aktualisieren:** Um eingehende Alarme zu empfangen, registriert sich das Alarmbaum-Werkzeug als Alarm-Beobachter in der lokalen Service-Registratur, die Beobachter und beobachtbare Objekte verwaltet (siehe Abschnitt 8.2). In der Service-Registratur kann die Datenzugriffsschicht aus Abschnitt 8.3.3 Alarm-Beobachter auffinden und sie unabhängig von der verwendeten Remote-Technologie über eingehende Alarme informieren. Der Umgang des Alarmbaum-Werkzeugs mit Alarmen ist im Listing 10.3 dargestellt.

```
1  public class  AlarmTreeView extends ViewPart {
2     ...
3     private void handleAlarmEvent(final IAlarm alarm) {
4        IAlarmNode node = getAlarmModel()
5              .findNodeById(alarm.getSourceNode().getId());
6        node.addAlarm(alarm);
7        refreshTreeNode(node);
8     }
9  }
```

Listing 10.3: Aktualisierung der Alarmdarstellung im Alarmbaum-Werkzeug (Quelle: eigene Erstellung)

Der `AlarmTreeView` ist die implementierende Klasse des Alarmbaum-Werkzeugs. In seiner `handleAlarmEvent`-Methode sucht das Werkzeug zunächst nach dem zugehörigen Alarm-Knoten im Alarmmodell und fügt diesem den Alarm hinzu. Nach dem gleichen Prinzip reagiert die `AlarmTreeView` auf Anlagenteile, die Operateure für Wartungs- oder Testzwecke stumm schalten.

- **Alarmbaum und Alarmtabelle verknüpfen:** Für die Verknüpfung neuer und vorhandener Werkzeuge wurden in Abschnitt 8.3.1 verschiedene Varianten zur Werkzeugkopplung vorgestellt und mit dem Mediator-Ansatz ein Kandidat gewählt, der in der aktuellen Eclipse-Plattform realisiert ist. In Eclipse übernimmt der sog. *Selection Service* die Rolle des Mediators. An diesem können sich Werkzeuge als Auslöser von Ereignissen (engl. Selection Provider) und/oder als Beobachter von Ereignissen (engl. Selection Listener) an- und abmelden.

  Wie im Abschnitt 8.3.1 herausgearbeitet, werden Werkzeuge meist ohne die Absicht entwickelt, sie später mit neuen Werkzeugen zu verknüpfen. In dem Werkzeug *Alarmtabelle* fehlt beispielsweise ein Beobachter, der auf Selektionen im Alarmbaum-Werkzeug reagiert und Alarme in der Tabelle entsprechend filtert. Dieser Beobachter muss daher über einen Eingriff in den Quellcode der Alarmtabelle hinzugefügt werden.

```
1  public class AlarmTableView extends ViewPart {
2     ...
3     private void registerListener() {
4        getSelectionService().addSelectionListener(
5            new ISelectionListener(ALARMTREE_ID, new ISelectionListener() {
6           public void selectionChanged(IWorkbenchPart part,
7                 ISelection selection) {
8              IAlarmNode selectedNode = getSelectedNode(selection);
9              if (selectedNode != null) {
10                filterPlantNode(selectedNode);
11             }
12          ...
13    }
14    private void filterPlantNode(IAlarmNode node) {
15       final PlantNodeFilter filter = new PlantNodeFilter(node);
16       alarmTableViewer.addFilter(filter);
17    }
18 }
```

Listing 10.4: Ergänzungen zur Werkzeuginteraktion im Alarmtabellen-Werkzeug (Quelle: eigene Implementation)

Im Listing 10.4 registriert das Alarmtabellen-Werkzeug zunächst den fehlenden Beobachter für Selektionen im Alarmbaum-Werkzeug (Zeile 3-13). Anschließend delegiert der Beobachter Selektionen im Alarmbaum-Werkzeug an einen `PlantNodeFilter`. Dieser filtert Alarme in der Alarmtabelle entsprechen dem selektierten Anlagenteil. Die Interaktion im Alarmbaum-Werkzeug geschieht auf die gleiche Weise, über die Anmeldung eines Beobachters am Mediator. Dieser erkennt Selektionen im Alarmtabellen-Werkzeug und veranlasst den Alarmbaum, sich bis zum selektierten Anlagenteil zu entfalten.

All diese Fragmente tragen in ihrer Gesamtheit zur Entwicklung der in Abbildung 10.7 dargestellten Alarm-Werkzeuge bei.

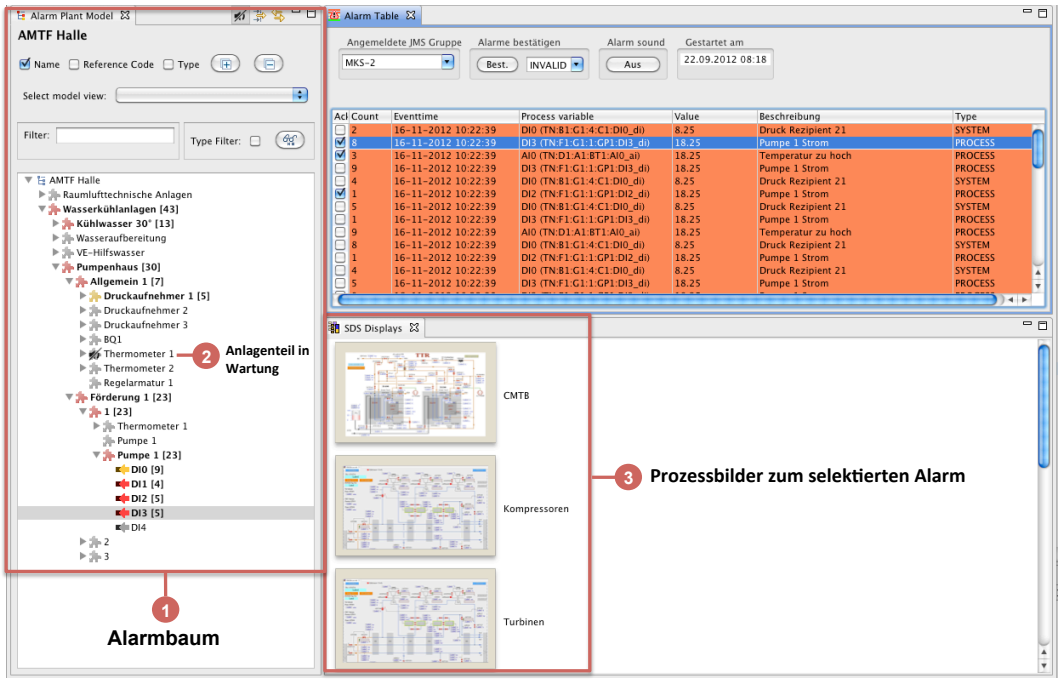


Abbildung 10.7: Kombination neuer und vorhandener Alarm-Werkzeuge (Quelle: eigene Erstellung aufbauend auf Pappert, 2012)

Das *Dekorierer-Muster* ermöglicht die Wiederverwendung des Anlagenmodell-Werkzeugs, um das Alarmbaum-Werkzeug mit alarmspezifischer Darstellung zu ergänzen (Punkt (1) der Abbildung 10.7). Das Alarmbaum-Werkzeug visualisiert ein Alarmmodell und präsentiert zu jedem Anlagenteil die Anzahl und den Schweregrad darin aufgetretener Alarme. Mit dem Alarmbaum-Werkzeug

können Operateure Anlagenteile in Wartung oder Testbetrieb stumm schalten, die daraufhin entsprechend gekennzeichnet werden (2). Das Prozessbildauflister-Werkzeug (3) zeigt alle zu einem selektierten Alarm zugehörigen Prozessbilder, die Operateure direkt aufrufen und dadurch Alarme im Kontext der Anlage analysieren können.

Die vorgestellten Alarm-Werkzeuge unterstützen Operateure, Alarme strukturiert zu betrachten und Anlagenteile in Wartung oder Testbetrieb stumm zu schalten. Die Interaktion zwischen dem Alarmbaum-Werkzeug und dem Alarmtabellen-Werkzeug ermöglicht zudem, Alarme auf einen Ausschnitt der Anlage einzugrenzen und erleichtert die Analyse komplexer Alarmsituationen. Diese Funktionalität muss durch eine entsprechende serverseitige Anwendungslogik unterstützt werden, dessen Einführung im folgenden Abschnitt vorgestellt wird.

10.2.3 Technische Realisierung auf der funktionalen Schicht

Für die Einführung der serverseitigen Alarm-Funktionalität wird die in Abschnitt 10.1 eingeführte modulare Plattform um alarmspezifische Plug-ins erweitert. Diese sind in der Abbildung 10.8 dargestellt, wobei zur besseren Übersicht die Anlagenmodell-Plug-ins nicht einzeln aufgeführt sind:

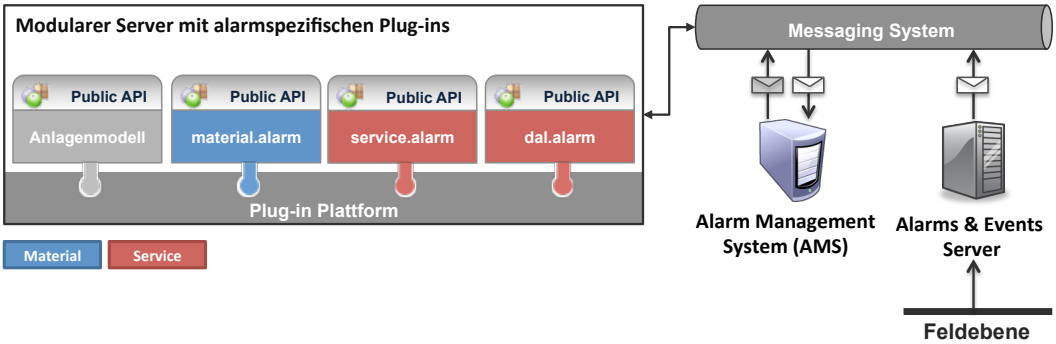


Abbildung 10.8: Serverseitige Plug-ins für die Alarmierung (Quelle: eigene Erstellung)

- **de.c1wps.geneal.desy.material.alarm:** Enthält das zum Anlagenmodell strukturähnliche Alarmmodell, das auf weitere Materialien wie z. B. Alarme zugreift.

- **de.c1wps.geneal.desy.service.alarm:** Der Alarmservice aktualisiert das Alarmmodell beim Eintreffen neuer Alarme und hilft dadurch, die reale Situation in der Anlage mit ihrem softwaretechnischen Modell zu synchronisieren.

- **de.c1wps.geneal.desy.dal.alarm:** Kapselt die konkrete Remote-Technologie für den Nachrichtenaustausch über das Messaging-System.

Die grundlegende Idee für die minimalinvasive Einführung der neuen serverseitigen Alarm-Funktionalität besteht darin, den vorhandenen A&E-Server und das Alarmmanagementsystem nicht zu verändern, sondern die von ihnen versendeten Alarme über das Messaging-System abzugreifen und in einem Alarmmodell zu hinterlegen. Clients können auf das Alarmmodell über den *Alarmservice* zugreifen, der das Alarmmodell bei jeder Änderung in der Anlage aktualisiert und Clients informiert. Folgende Schritte sind dazu nötig:

- **Alarme empfangen:** Damit der Alarmservice Alarme empfangen kann, muss er sich als Alarm-Beobachter (`IAlarmEventListener`) in der Service-Registratur anmelden. Dort kann er von der Datenzugriffsschicht aus Abschnitt 8.3.3 gefunden und technologieunabhängig über eingehende Alarme informiert werden. Der Alarmservice aktualisiert daraufhin das Alarmmodell und informiert Clients, vorausgesetzt das zum Alarm zugehörige Anlagenteil befindet sich nicht in Wartung oder Testbetrieb (siehe Listing 10.5).

```
1  public class AlarmServiceImpl implements IAlarmService,
       IAlarmEventListener{
2     private IAlarmModel alarmModel;
3     @Override
4     public void alarmEmerged(IAlarm alarm) {
5        alarmModel.addAlarm(alarm);
6        IAlarmNode node = alarm.getSourceNode();
7        if (!node.isInMaintenance()) {
8           informAlarmClients(alarm);
9        }
10    ...
11 }
```

Listing 10.5: Alarmbehandlung im Alarmservice (Quelle: eigene Implementation)

- **Alarmmodell aktualisieren:** Das Alarmmodell erweitert das generische Modell aus Abschnitt 8.2.2 mit alarmspezifischer Funktionalität. Dazu gehört beispielsweise die Zuordnung eingehender Alarme zum entsprechenden Anlagenteil (Listing 10.6). Hierfür extrahiert das Alarmmodell die Prozessvariable aus dem eingehenden Alarm, sucht nach dem Anlagenteil zur Prozessvariable und fügt dem Anlagenteil den Alarm hinzu.

```
1  public class AlarmModel extends GenericModel<IAlarmNode> implements
       IAlarmModel {
2     ...
3     public void addAlarm(IAlarm alarm) {
4        String pvName = alarm.getPVName();
5        IAlarmNode node = findNodeByPvName(pvName);
6        node.addAlarm(alarm);
7        alarm.setSourceNode(node);
8     }
9  }
```

Listing 10.6: Alarmmodell (Quelle: eigene Implementation)

- **Kapselung der Komplexität in Messaging-Systemen:** Die Datenzugriffsschicht kapselt die Verbindung mit dem Messaging-System und den verteilten Zugriff auf Alarme (siehe Abschnitt 8.3.3). Zusätzlich konvertiert sie technologiespezifische Alarm-Nachrichten in fachliche Alarm-Materialien und benachrichtigt Alarm-Beobachter, wie den Alarmservice, über neue Alarme (Listing 10.7).

```
1  public class AlarmDALImpl implements EventHandler {
2     private List<IAlarmEventListener> alarmEventListener;
3     ...
4     @Override
5     public synchronized void handleEvent(Event event) {
6        if (event.getTopic().equals("geneal/ioc-alarms")) {
7           IAlarm alarm = convertEventToAlarm(event);
8           informListener(alarm);
9        }
10       ...
11    }
```

Listing 10.7: Alarm-Datenzugriffsschicht (Quelle: eigene Implementation)

Die Realisierung der serverseitigen Alarm-Funktionalität konkretisiert die Metapher *Modernisierung der Altstadt*. Der jahrelang erprobte Alarms&Events-

Server übernimmt weiterhin die hardwarenahe Überwachung der Schwellwertüberschreitungen. Das bewährte Alarmmanagementsystem prüft diese Überschreitungen gegen ein ausgeklügeltes Regelwerk und filtert dadurch viele, jedoch nicht alle unnötigen Alarme heraus. Die neue Alarm-Funktionalität ergänzt diese Systeme, in dem sie ein explizites Alarmmodell einführt. Dieses vermittelt Operateuren ein einheitliches Bild der gegenwärtigen Alarmsituation, strukturiert Alarme und hilft bei Alarmfluten, die Übersicht zu behalten. Zusätzlich reduziert die neue Alarm-Funktionalität unnötige Alarme, die von Anlagenteilen in Wartung oder Testbetrieb verursacht werden.

10.3 Zusammenfassung

In diesem Kapitel wurde aus einer anwenderorientierten Sicht gezeigt, wie neue und vorhandene Werkzeuge kombiniert werden können, um die Arbeitssituationen der Operateure in Leitständen zu verbessern. Mit dem Anlagenmodell und dem Alarmmodell wurden in diesem Kapitel zwei explizite Modelle eingeführt, die Operateuren helfen, ein einheitliches Bild von der gegenwärtigen Situation in der Anlage zu entwickeln, ihre Arbeit besser zu koordinieren und die mentale Belastung zu reduzieren. Die inkrementelle Einführung des Kernsystems *Anlagenmodell* und des Spezialsystems *Alarmierung* hat zudem gezeigt, wie neue Leitstandfunktionalität schrittweise eingeführt werden kann, beginnend mit den Bereichen, in denen Operateure den größten Handlungsbedarf sehen. Dies stellt den Nutzen neuer Leitstandfunktionalität in den Vordergrund und hilft, die Skepsis der Operateure bei einer Leitstandmodernisierung zu überwinden.

Auf der technischen Ebene wurde in diesem Kapitel die modellbasierte und modulare Konstruktion neuer Leitstandfunktionalität auf der Präsentations- und funktionalen Schicht demonstriert und ihre minimalinvasive Einführung in eine bestehende Systemlandschaft vorgestellt. Damit die neue und vorhandene Leitstandfunktionalität nicht isoliert nebeneinander besteht, wurden auf der Präsentationsschicht Werkzeuge minimalinvasiv verknüpft und ihre Interaktion ermöglicht. Auf der funktionalen Schicht wurde die neue Leitstandfunktionalität über das Messaging-System an eine vorhandene System-

landschaft angeschlossen und ihr Datenaustausch ermöglicht. Die in diesem Kapitel vorgestellten Lösungsansätze sollten nicht nur den aktuellen Anforderungen an Leitstandfunktionalität genügen, sondern das Potential aufweisen, auch zukünftige Anforderungen zu unterstützen. Die Architekturkonzepte dieser Arbeit und die modulare Plug-in-Technologie bieten gemeinsam die dafür nötige Flexibilität und Erweiterbarkeit.

Damit wurde in diesem Kapitel die anwenderorientierte und softwaretechnische Sichtweise auf die Leitstandmodernisierung zusammengeführt, konstruktiv im DESY-System umgesetzt und die Einsatzfähigkeit der konzipierten Lösung demonstriert. Im nächsten Kapitel wird der Ansatz dieser Arbeit evaluiert und auf die Leitstandaufgaben *Datenzugriff* sowie *Historischer Datenzugriff* übertragen.

Kapitel 11

Evaluation

In dieser Arbeit wird die Forschungsmethodik *Design Research* verwendet. Ihre Ergebnisse können in Konzepte, Modelle, Methoden und Exemplare oder Ausprägungen unterteilt werden (vgl. Abschnitt 1.4). March und Smith ergänzen diese Ergebnisse um die Forschungsaktivitäten Konstruieren, Evaluieren, Theorien aufstellen und Beweisen. *Konstruieren* demonstriert die Machbarkeit eines Artefaktes. In der *Evaluation* werden Bewertungskriterien aufgestellt, um die Güte des konstruierten Artefaktes zu beurteilen. *Theorien* erklären wie oder warum ein Artefakt in seiner Umgebung funktioniert. *Beweise* bestätigen oder widerlegen Theorien [March und Smith, 1995, S. 258]. March und Smith stellen Forschungsergebnisse und Forschungsaktivitäten in einer Matrix gegenüber, die sie als Forschungs-Framework bezeichnen [March und Smith, 1995, S. 255]. Jede Zelle in dieser Matrix präsentiert unterschiedliche Forschungsergebnisse, die durch verschiedene Forschungsaktivitäten überprüft werden können. Die Forschungsergebnisse und Aktivitäten dieser Arbeit sind in der Abbildung 11.1 im Framework eingeordnet. Die konstruierten Artefakte dieser Arbeit sind vornehmlich Konzepte, Methoden und Exemplare, die in der DESY-Infrastruktur realisiert und evaluiert wurden. Da eine umfassende Evaluation über mehrere Industriezweige und Unternehmen im Rahmen dieser Arbeit nicht möglich ist, ist die Spalte *Evaluation* in der Abbildung 11.1 nicht vollständig ausgefüllt. Die Einordnung der Ergebnisse und Aktivitäten dieser Arbeit in das Forschungs-Framework wird in den folgenden Abschnitten erläutert.

Forschungsaktivitäten

Forschungs-ergebnisse	Konstruieren	Evaluieren	Theorien aufstellen	Beweisen
Konzepte				
Modelle				
Methoden				
Exemplare				

Legende: = im Fokus der Arbeit

Abbildung 11.1: Research Framework (Quelle: March und Smith, 1995)

11.1 Evaluation der Konzepte, Modelle, Methoden und Exemplare

Die in dieser Arbeit konstruierten Artefakte können nach der ISO/IEC 9126-Norm für Softwarequalität hinsichtlich ihrer Funktionalität, Änderbarkeit, Effizienz, Zuverlässigkeit, Übertragbarkeit, Gebrauchstauglichkeit oder anderer relevanter Eigenschaften evaluiert werden. Die Bewertung dieser Eigenschaften in Softwaresystemen hängt stark vom betrachteten Artefakt und der verwendeten Bewertungskriterien oder Metriken ab. Während beispielsweise die Effizienz der ablauffähigen Exemplare durch Messungen der Performanz oder der zyklomatischen Komplexität quantitativ erfassbar und vergleichbar ist, ist die Bewertung der Merkmale Funktionalität, Gebrauchstauglichkeit oder Änderbarkeit immer subjektiv und von der menschlichen Wahrnehmung und individueller Präferenzen abhängig [vgl. Sommerville 2001, S. 558; Hevner et al. 2004, S. 86]. Um die Bewertung der Artefakte dieser Arbeit nicht zu stark an einzelnen Individuen auszurichten, erfolgte ihre Evaluation im Austausch mit mehreren Institutionen, die in den folgenden Abschnitten detailliert vorgestellt werden, sowie mit zwei Abteilungen beim DESY:

- **MKS-2:** Die Abteilung Maschine-Kryogenik-Supraleitung[1] (MKS-2) stellt die Helium-Kälteversorgung für die supraleitenden Komponenten der DESY-Beschleuniger sicher und hat einen speziellen Fokus auf die Prozesskontrolle der Kälteanlagen.

[1]MKS-2 Abteilung: `http://www-mks2.desy.de/content/index_ger.html`

- **MKK:** Die Abteilung Maschine-Kraft-Kühlung[2] (MKK) ist für die Energieversorgung der Gebäude, Beschleuniger und Experimente bei DESY zuständig.

Beide Abteilungen setzen die gleiche Leitstandsoftware *Control System Studio*[3] (CSS) ein. Auf Basis dieses Systems wurden die Konzepte, Modelle, Methoden und Exemplare dieser Arbeit konstruiert und evaluiert. Die Evaluation dieser Artefakte wird detailliert in den folgenden Abschnitten beschrieben.

11.1.1 Konzepte

Konzepte beeinflussen die Wahrnehmung des Problemfeldes und bieten ein Vokabular, um Aufgaben, Probleme und Lösungen zu beschreiben (siehe Abschnitt 1.4). Zu Beginn der Forschungsarbeit wurde zunächst das Problemfeld einer rein technologiezentrierten und ingenieurwissenschaftlichen Sichtweise in der Automatisierungstechnik herausgearbeitet. Dies erfolgte im Rahmen einer Literaturrecherche im Bereich der Arbeitspsychologie. Workshops mit Prof. Friedhelm Nachreiner von der Gesellschaft für Arbeits-, Wirtschafts- und Organisationspsychologischer Forschung e. V., sowie mit Prof. Leon Urbas vom Institut für Automatisierungstechnik der technischen Universität Dresden und Interviews mit Vertr. Prof. Dr. Martin Christof Kindsmüller vom Fachbereich Mensch-Computer-Interaktion der Universität Hamburg halfen, die erarbeiteten Konzepte zu evaluieren. Weitere Evaluation des Problemfeldes fand im Rahmen von Leitstand-Besichtigungen namhafter Industrieunternehmen statt. Dazu gehören die GeneAL-Pilotpartner DESY, HPA und HHLA sowie die Unternehmen Hamburger Hochbahn AG, Tunnelbetrieb Hamburg, Deutsche Post DHL (Hamburg), Kühne + Nagel (Hamburg), ABB (Ladenburg), BTC (Berlin), die Müllverbrennungsanlage Bielefeld und Siemens Hamburg. Der Austausch mit diesen Unternehmen bestätigte die immer noch vorherrschende technologiezentrierte Sichtweise in Leitständen, ein damit einhergehendes technologiezentriertes Design der Leitstandsysteme und die daraus resultierenden Probleme für Operateure.

[2] MKK Abteilung: `http://mkk.desy.de`
[3] siehe Anhang A.2

Ausgehend von der formulierten Problemstellung half eine intensive Literaturarbeit im Bereich der Software- und Automatisierungstechnik bei der Eingrenzung des Lösungsraumes. Dazu hat auch der Austausch mit den Instituten für Automatisierungstechnik an der Helmut-Schmidt-Universität Hamburg, an der Otto-von-Guericke-Universität Magdeburg sowie dem Institut für Wirtschaftsinformatik an der Johannes-Kepler-Universität Linz und dem Institut für Software Systems Engineering an der Universität Duisburg-Essen beigetragen. Aus der formulierten Problemstellung und dazu passenden Lösungsansätzen entstanden Konzepte, die mit den vorgestellten Instituten und Pilotpartnern in Interviews und Workshops evaluiert wurden.

11.1.2 Modelle

Modelle nutzen Konzepte, um eine reale Situation inklusive des Design-Problems und Lösungsraums darzustellen (siehe Abschnitt 1.4). Modelle können nach March und Smith auch Beschreibungen sein, wie Dinge funktionieren [March und Smith, 1995, S. 256]. Nach Hevner et al. repräsentieren Modelle auch Verbindungen zwischen Komponenten des Problem- und Lösungsraums, um Design-Entscheidungen und ihre Auswirkungen auf die reale Welt zu erforschen [Hevner et al., 2004, S. 79]. Diese unscharfe Auffassung des Modell-Begriffs erschwert eine Zuordnung der Forschungsergebnisse zu diesem Artefakt. In dieser Arbeit wird daher der Modell-Begriff nach Stachowiak als ein Abbild seines Originals mit Verkürzungsmerkmalen verstanden (siehe Abschnitt 4.2). Als zentrales Modell dieser Arbeit wurde das Anlagenmodell in Abschnitt 4 herausgearbeitet, auf dessen Fundament die Leitstandmodernisierung fußt. Statt ein weiteres eigenentwickeltes Anlagenmodell einzuführen, wurden zu Beginn der Forschungsarbeit Anforderungen an Anlagenmodelle formuliert, mit vorhandenen Standards verglichen und mit dem OPC-UA-Standard ein geeigneter Modell-Kandidat gewählt.

Die Evaluation OPC-UA-basierter Modelle hat gezeigt, dass diese eine geeignete Metamodellierungssprache bieten, die nicht nur aktuelle, sondern auch zukünftige Anforderungen an Anlagenmodelle erfüllen kann. In dieser Arbeit wurde aber auch gezeigt, dass die OPC-UA-Technologie Anlagenbetreibern nur wenig Orientierung bei der Verwendung, Einführung, kontinuierlicher War-

tung und Erweiterung der Anlagenmodelle bietet. Zu Beginn der Forschungsarbeit wurde zunächst der Nutzen der Anlagenmodelle aus anwenderorientierter Sicht durch Literaturrecherche und Interviews mit Operateuren herausgearbeitet und ihr Potential zur Verbesserung der elementaren Leitstandaufgaben der Operateure aufgezeigt. Für die Realisierung der identifizierten Lösungsansätze wurde für das Anlagenmodell eine hohe Strukturähnlichkeit zwischen den realen Anlagenteilen und den Modellelementen gefordert. Auf der technischen Ebene wurde die Langlebigkeit der Anlagenmodelle hervorgehoben und die damit einhergehende Problematik herausgearbeitet, dass Anlagenmodelle mit fachlichen und baulichen Veränderungen in der Anlage anwachsen und langfristig zu Monolithen zu degenerieren drohen.

Als Lösungsansatz für dieses Problem wurde in Abschnitt 8.2.2 ein generisches Modell mit Spezialmodellen eingeführt und am Beispiel des Anlagenmodells und des Alarmmodells evaluiert. Das generische Modell kapselt die wiederverwendbare Modellfunktionalität des Anlagen- und Alarmmodells, reduziert somit die Code-Duplikation und stellt über die Typisierung sicher, dass nur kompatible Knoten eines Modells miteinander kombiniert werden dürfen. Die Spezialmodelle beinhalten kontextspezifische Leitstandfunktionalität und können dadurch weitgehend unabhängig voneinander gewartet und erweitert werden, ohne ihre Kompatibilität einzubüßen. Diese Konzepte präsentieren ergänzende Modellierungsansätze, die dem Artefakt *Modelle* in der Abbildung 11.1 zugeordnet werden können. Da die Modellierungsansätze dieser Arbeit auf den OPC-UA-Konzepten aufbauen, sind die entsprechenden Zellen in der Abbildung 11.1 nur zum Teil ausgefüllt.

11.1.3 Methoden

Methoden basieren auf Konzepten und Modellen und bieten eine Anleitung, wie Probleme gelöst werden können (siehe Abschnitt 1.4). Zu Beginn der Forschungsarbeit wurden wiederkehrende Leitstandprobleme auf ein technologiezentriertes Design der Leitstandsysteme zurückgeführt. Als Lösungsansatz für die aufgeführten Probleme wurde die *anwenderorientierte Sichtweise* eingeführt, die das technologiezentrierte Design ergänzt (siehe Kapitel 3). Damit ist eine Methodik verknüpft, Lösungen für Probleme im Anwendungsbereich

stets an ihrem Nutzen für die Anwender auszurichten und die technische Realisierung nachfolgend zu eruieren. Mit dieser Sichtweise wurden im weiteren Verlauf der Arbeit anwenderorientierte Lösungen konzipiert und die Notwendigkeit herausgearbeitet, diese Lösungen in vorhandene Leitstandsysteme einführen zu müssen. In zahlreichen Interviews und Workshops mit Operateuren und Ingenieuren wurde dabei deutlich, dass diese Aufgabe mit technischen und organisatorischen Veränderungen und Risiken für den Anlagenbetrieb verbunden ist. Zur Eingrenzung dieser Risiken wurde auf der technischen Ebene die Problematik herausgearbeitet, neue Leitstandfunktionalität mit wenigen Eingriffen und Seiteneffekten einführen zu müssen, um den Anlagenbetrieb möglichst wenig zu stören. Bei den organisatorischen Herausforderungen wurde die Skepsis der Operateure gegenüber Veränderungen hervorgehoben. Diese resultiert aus ihrer Erfahrung mit vergangenen Technologiewechseln und oft ausbleibenden Verbesserungen für ihre Arbeitssituationen.

Als Lösungsansatz für die aufgezeigten Probleme und als zentralen Beitrag dieser Arbeit wurde die *minimalinvasive Vorgehensweise* eingeführt und auf der technischen und Anwenderebene evaluiert. Auf der Anwenderebene wurden im Rahmen einer Literaturrecherche evolutionäre Vorgehensmodelle vorgestellt, mit den Konzepten Kernsystem und Spezialsysteme sowie Softwareerweiterung ergänzt und in der Entwurfsmetapher *Modernisierung der Altstadt* vereint (siehe Kapitel 7). Diese Entwurfsmetapher und das damit verbundene Vorgehen resultieren aus dem Wunsch der Operateure und Ingenieure, vorhandene Systeme und Wissen zu erhalten und die Leitstandmodernisierung mit wenigen Eingriffen und Auswirkungen auf den Anlagenbetrieb durchzuführen. Dazu wurde auf der technischen Ebene eine Gesamtarchitektur zur Konstruktion (meta-)modellbasierter Anwendungen herausgearbeitet, mit Mikro-Architekturkonzepten zur minimalinvasiven Einführung neuer Leitstandfunktionalität ergänzt und Ansätze zur losen Kopplung neuer und vorhandener Leitstandfunktionalität vorgestellt (siehe Kapitel 8). In dieser Arbeit wurde die anwenderorientierte und minimalinvasive Vorgehensweise zusammengeführt und als Methodik für die Leitstandmodernisierung vorgeschlagen. Diese Methodik wurde in Diskussionen und Workshops mit dem Fachbereich Softwaretechnik der Universität Hamburg und dem Instituten für

Automatisierungstechnik an der Helmut-Schmidt-Universität Hamburg sowie mit DESY Operateuren und Ingenieuren evaluiert.

11.1.4 Exemplare

Im Design Research müssen die konzipierten Lösungsansätze in einer technischen Infrastruktur eingesetzt und gegen die Anforderung aus dem Anwendungskontext geprüft werden, um ihre Einsatzfähigkeit und Tragfähigkeit zu demonstrieren (siehe Abschnitt 1.4). Dafür wurden im Verlauf der Forschungsarbeit Softwareartefakte entwickelt, in der technischen Infrastruktur des Pilotpartners DESY eingesetzt und evaluiert. Da die Erprobung der entwickelten Artefakte in der realen Anlage und am Beispiel realer Situationen zu viele Risiken für den Anlagenbetrieb barg, wurden im Rahmen der Forschungsarbeit Daten aus dem realen System extrahiert und in einer Testumgebung mit Simulatoren für Echtzeitdaten, Alarme und historische Daten bereitgestellt. Dies ermöglichte eine praxisnahe Evaluation realer Situationen mit neuer Leitstandfunktionalität, ohne den Anlagenbetrieb zu stören.

Die Entwicklung der neuen Leitstandfunktionalität verlief entsprechend der evolutionären Vorgehensweise zyklisch und iterativ, beginnend mit der Umsetzung des Kernsystems (siehe Abschnitt 7.1). Darauf aufbauend erfolgte die Realisierung der Spezialsysteme für die elementaren Leitstandaufgaben. Zu Beginn jeder Realisierung wurden zunächst typische Arbeitsabläufe der Operateure analysiert und mit einem Werkzeug zur Geschäftsprozessmodellierung dokumentiert. In anschließenden Workshops mit Operateuren wurden besonders kritische Arbeitsabläufe identifiziert, dazu passende Lösungsalternativen diskutiert und ihre Umsetzung priorisiert. Um Operateuren eine grobe Vorstellung von der zu entwickelnden Leitstandfunktionalität zu vermitteln, wurden Mock-ups eingesetzt (wie z. B. in Abschnitt 5). Diese unterstützen die Evaluation des Designs und erlauben Operateuren das Ergebnis früh zu beeinflussen.

Die Konstruktion der Software-Artefakte erfolgte in dreiwöchigen Iterationen. Das Ergebnis jeder Iteration bestand aus einer ablauffähigen Software, in der Teile der Anforderungen realisiert, von ausgewählten Operateuren erprobt

und bewertet wurden. Das Feedback der Operateure wurde in der anschließenden Iteration eingearbeitet, in der auch neue Anforderungen umgesetzt wurden. Im Anschluss an die Realisierung der jeweiligen Leitstandfunktionalität wurde diese in Workshops mit weiteren Operateuren der DESY-Abteilungen MKS-2 und MKK diskutiert und evaluiert.

Um die Tragfähigkeit der anwenderorientierten und minimalinvasiven Leitstandmodernisierung zu festigen, wurde der Ansatz dieser Arbeit auf die Leitstandaufgaben Datenzugriff und historischer Datenzugriff übertragen, konstruktiv umgesetzt und evaluiert.

11.2 Evaluation am Beispiel Datenzugriff

In Abschnitt 3.1 wurde als größte Herausforderung für Operateure, im Bereich Datenzugriff, die manuelle und fehleranfällige Erstellung, Konfiguration und Pflege der Prozessbilder herausgearbeitet. Anwenderorientierte Ansätze adressieren diese Problematik, in dem sie besonders fehleranfällige Aufgaben softwaretechnisch unterstützen. Dazu wurden automatische und halbautomatische Projektierungsansätze vorgestellt, von denen in dieser Arbeit das halbautomatische Objekt-Typ-Konzept exemplarisch herangezogen wird (siehe Abschnitt 3.1). Für die minimalinvasive Einführung dieses Ansatzes wird die Metapher *Modernisierung der Altstadt* aufgegriffen und die neue Funktionalität als Softwareerweiterung zu den vorhandenen Projektierungs-Werkzeugen eingeführt. Diese Erweiterung kann im Rahmen einer inkrementellen Vorgehensweise als Spezialsystem *Datenzugriff* aufgefasst werden und sich beispielsweise an die Einführung des Spezialsystems *Alarmierung* anschließen.

11.2.1 Benutzungsmodell

Für die halbautomatische Projektierung wird ein Modell der Anlage benötigt, in dem Anlagenteile ihrem Typ zugeordnet sind (siehe Abschnitt 3.1). Das in Abschnitt 4.5 eingeführte Anlagenmodell erfüllt diese Anforderung und realisiert dazu das Konzept einer Typ-Bibliothek. Diese verwaltet alle im Anlagenmodell verfügbaren Anlagenteil-Typen. Auf dieser Grundlage können neue

Werkzeuge entwickelt werden, die das bestehende Projektierungs-Werkzeug (Punkt (1) der Abbildung 11.2) ergänzen.

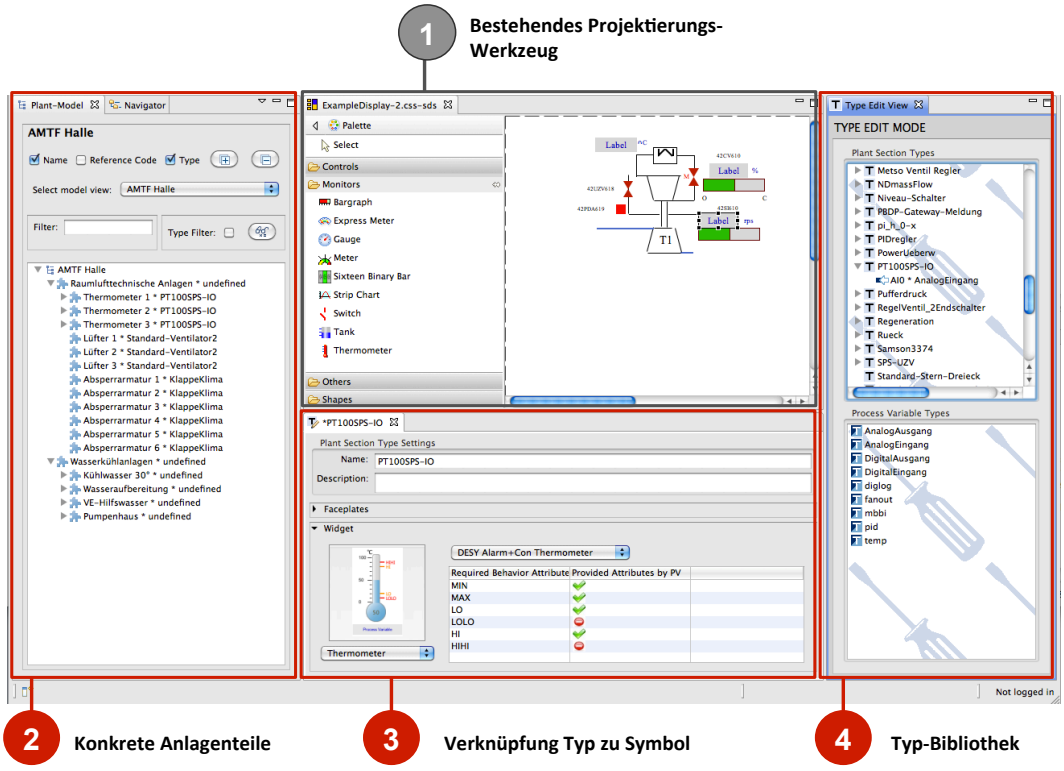


Abbildung 11.2: Modellbasiertes Werkzeug für die Projektierung (Quelle: eigene Erstellung)

Zu den neuen Werkzeugen gehört das Typ-Bibliothek-Werkzeug (4), das sowohl einfache Anlagenteil-Typen wie Magnetventile oder Temperaturfühler, als auch komplexe Typen wie Motoren oder Turbinen visualisiert. Anlagenteil-Typen können im entsprechenden Typ-Editor-Werkzeug (3) konfiguriert und beispielsweise mit dem dazugehörigen Symbol und seiner Darstellung in Alarmsituationen verknüpft werden. Sobald diese Konfiguration abgeschlossen ist, können Projekteure konkrete Anlagenteile per Drag&Drop aus dem Anlagenmodell (2) auf das Prozessbild ziehen, wo sie vorkonfiguriert mit dem dazugehörigen grafischen Symbol erscheinen. Nachträgliche Änderungen jeglicher Konfiguration können auf der Typ-Ebene vorgenommen und auf alle konkreten Anlagenteile und bereits erstellte Prozessbilder automatisch übertragen werden. Das vereinfacht Projekteuren erheblich die manuelle Rekonfiguration der Anlagenteile.

11.2.2 Technische Realisierung

Die minimalinvasive Einführung der halbautomatischen Projektierung in die bestehende Anwendung erfolgt über den in Abschnitt 8.2 vorgestellten Plug-in-Erweiterungsmechanismus. Die Basis für die neue Funktionalität bildet das Anlagenmodell mit Typ-Informationen zu Anlagenteilen. Den Zugriff auf Typ-Informationen kapselt eine `PlantTypeLibrary`, deren Baumstruktur in der Abbildung 11.2 unter **(4)** dargestellt ist. Die Konfiguration der Anlagenteil-Typen mit ihrem Symbol und Verhalten in Alarmsituationen erfolgt mit Hilfe des Typ-Editor-Werkzeugs **(3)**. Diese Konfiguration ist unabhängig von der Anlagenstruktur oder der Struktur der Typ-Bibliothek. Sie benötigt lediglich eine eindeutige Abbildung von einem Anlagenteil-Typ auf seine UI-spezifische Konfiguration. Diese Konfiguration wird daher mit Hilfe eines fachlichen Services realisiert (siehe Abschnitt 8.2.2). In Listing 11.1 persistiert der `PlantWidgetService` eine `WidgetConfiguration`, in der alle zu einem Anlagenteil-Typ zugehörige Informationen, wie Symbol und Alarmverhalten, hinterlegt sind. Der `PlantWidgetService` entkoppelt das Anlagenmodell von der UI-spezifischen Konfiguration der Anlagenteil-Typen, wodurch Modell und Konfigurationen weitgehend unabhängig voneinander gewartet und erweitert werden können.

```
1  public class PlantTypeEditor extends ViewPart{
2     private IPlantWidgetService widgetService;
3     private IPlantUnitType plantUnitType;
4     ...
5     public void saveTypeConfiguration() {
6        WidgetConfiguration widgetConf = new WidgetConfiguration(plantUnitType);
7        widgetConf.setWidget(getSelectedWidget());
8        widgetConf.setAlarmBehaviour(getSelectedAlarmBehavior());
9        widgetService.saveWidgetConfiguration(widgetConf);
10    }
```

Listing 11.1: Typ-basierte Konfiguration (Quelle: eigene Implementation)

Während der Projektierung muss aus dem Anlagenteil, das Operateure per Drag&Drop auf das Prozessbild ziehen, der zugehörige Typ extrahiert, die zum Typ passende Widget-Konfiguration aus dem `PlantWidgetService` geladen und schließlich das entsprechende Widget auf dem Prozessbild visualisiert werden. Das Prozessbild-Werkzeug unterstützt jedoch nur bekannte Drag&Drop-Operationen für Zeichenelemente wie Linien, Kreise oder Rechte-

cke. Die neue Funktionalität kann prinzipiell direkt im Prozessbild-Werkzeug ergänzt werden, wodurch dieses Wissen darüber erlangt, wie zu Anlagenteilen passende Widgets bezogen werden. Dies führt zu einer Abhängigkeit zwischen der neuen Funktionalität und dem Prozessbild-Werkzeug, was sich negativ auf die Wart- und Erweiterbarkeit des Werkzeugs auswirkt. Statt die neue Drag&Drop-Funktionalität invasiv im Prozessbild-Werkzeug zu implementieren, bietet sich dafür das *Command-Pattern* nach [Gamma et al., 2009] an, das im DESY-Prozessbild-Werkzeug bereits realisiert ist. Dieses Muster erlaubt, neue Funktionalität in einem Befehl (engl. Command) zu kapseln. Das Prozessbild-Werkzeug führt diesen Befehl über Operationen, wie z. B. `zeichneWidget()` aus, ohne die konkrete Implementierung der Operation zu kennen [vgl. Gamma et al., 2009, S. 73]. Das Prozessbild-Werkzeug kann dadurch mit wenigen Eingriffen mit neuer Funktionalität ergänzt und seine Erweiterbarkeit sowie Evolutionsfähigkeit erhöht werden.

11.3 Evaluation am Beispiel historischer Datenzugriff

In Abschnitt 3.3 wurde die Analyse historischer Daten mit Kurvenbildern vorgestellt und die Problematik aufgezeigt, dass Kurvenbilder nur wenige, höchstens 10 Messgrößen gleichzeitig darstellen können. Als Lösungsansatz wurde die Idee skizziert, Kurvenbilder in Kombination mit Prozessbildern zu verwenden. Diese erlauben, historische Daten im Kontext der Anlage zu analysieren. Auch für dieses Vorhaben wird die Metapher *Modernisierung der Altstadt* aufgegriffen und vorhandene Werkzeuge im Rahmen einer Softwareerweiterung um ein zusätzliches Werkzeug ergänzt.

11.3.1 Benutzungsmodell

Für die Analyse historischer Daten wird das vorhandene Prozessbild- und Kurvenbild-Werkzeug (1 und 2) mit einem neuen *Archiv-Stepper-Werkzeug* (3) ergänzt (siehe Abbildung 11.3). Dieses erlaubt Operateuren, den Zeitraum

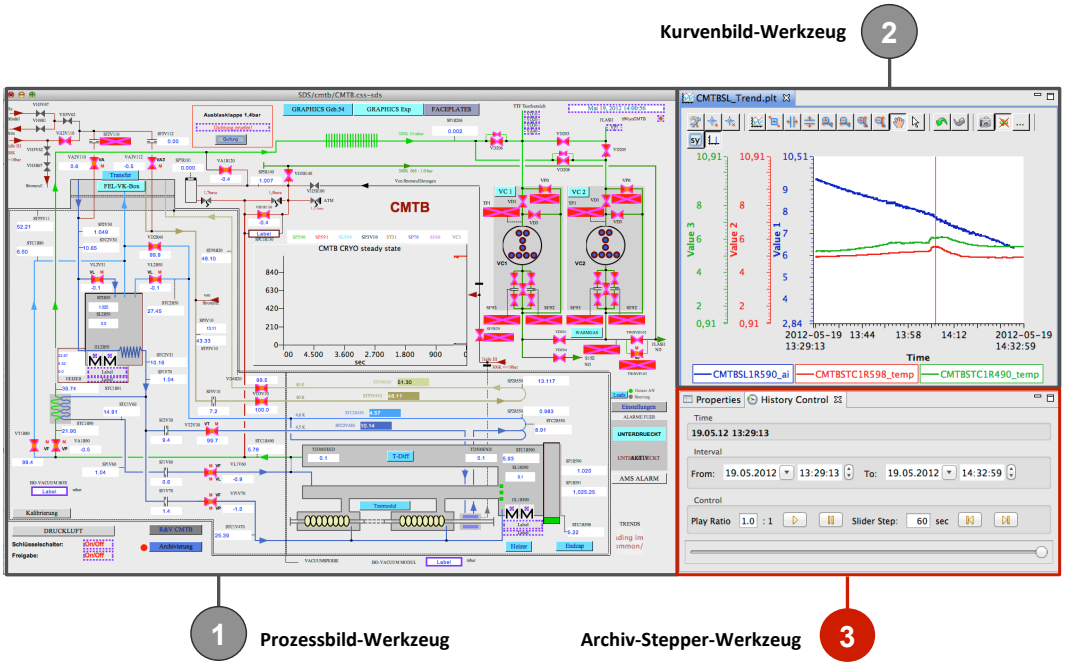


Abbildung 11.3: Modellbasiertes Werkzeug für die historische Datenanalyse (Quelle: Mein, 2013)

für die historische Datenanalyse auszuwählen und innerhalb der gewählten Zeitreihe schrittweise vergangene Situationen nachzufahren. Das Archiv-Stepper-Werkzeug löst mit jedem Schritt entlang der historischen Zeitreihe einen Impuls aus, der das Prozessbild- und Kurvenbild-Werkzeug veranlasst, ihre Darstellung zu aktualisieren. Es bietet zudem eine Abspiel-Funktion, die eine vergangene Situation in einem festgelegten Intervall automatisch wiedergibt. Das Archiv-Stepper-Werkzeug kombiniert die Vorteile des Prozessbild- und Kurvenbild-Werkzeugs. Während das Kurvenbild-Werkzeug eine aggregierte Übersicht über einen längeren Zeitraum historischer Werte für eine oder einige wenige Prozessvariablen bietet, erlaubt das Prozessbild-Werkzeug, eine historische Situation detailliert im Kontext der Anlage zu analysieren.

In dem Prozessbild-Werkzeug der Abbildung 11.3 sind einige Anlagenteile Magenta eingefärbt. Für diese Anlagenteile existieren keine historischen Daten im ausgewählten Zeitraum. Die Ursache hierfür konnte darauf zurückgeführt werden, dass beim DESY Ingenieure Hardware-Änderungen in der Anlage vornehmen und Softwareentwickler diese Änderungen manuell mit dem Archiv für historische Daten synchronisieren müssen. Diese Aufgabe ist sehr

fehleranfällig, da Softwareentwickler häufig gar nicht oder sehr spät über Änderungen in der Anlage informiert werden. Dieses Problem ist beim DESY bekannt, eine passende Lösung lag jedoch zum Zeitpunkt der Entstehung dieser Arbeit nicht vor.

11.3.2 Technische Realisierung

Für die Visualisierung historischer Daten in Prozessbildern müssen diese, neben dem historischen Messwert, auch dazugehörige Metadaten wie z. B. Alarmgrenzen enthalten, um ihre Überschreitung zu visualisieren. Wie in Abschnitt 3.3 beschrieben, werden historische Daten häufig ohne Metadaten gespeichert, um die Datenmenge in der Datenbank zu reduzieren. Dies erschwert nicht nur die Visualisierung von Alarmzuständen, sondern auch Füllstände für z. B. Tanks, für die minimale und maximale Obergrenzen benötigt werden. Ohne diese Angaben ist eine korrekte Initialisierung dazu passender grafischer Symbole nicht möglich. Die fehlenden Metadaten zu Messwerten können im Anlagenmodell hinterlegt und während der historischen Datenanalyse mit den entsprechenden Messwerten verknüpft werden. Die Anreicherung historischer Messwerte mit Metadaten übernimmt der fachliche `HistoryDataService`. An diesem melden sich das Prozessbild-und Kurvenbild-Werkzeug als Beobachter für Messwerte an, woraufhin der Service passende Zeitreihen zusammenstellt, die Werkzeuge aktualisiert und synchronisiert.

Das Prozessbild- und das Kurvenbild-Werkzeug beziehen historische Daten über eine Datenzugriffsschicht (DAL) (siehe Abschnitt 8.3.3). Während das Prozessbild-Werkzeug auf die DAL-Schnittstelle für Echtzeitdaten zugreift, bezieht das Kurvenbild-Werkzeug Daten aus der DAL-Schnittstelle für historische Daten, die in einer Datenbank hinterlegt sind. Für die Analyse historischer Daten im Prozessbild- und Kurvenbild-Werkzeug ist es erforderlich, den Datenzugriff beider Werkzeuge auf den `HistoryDataService` umzuleiten. Der Austausch der DAL-Schnittstellen vorhandener Werkzeuge ist prinzipiell gänzlich ohne Modifikationen möglich, vorausgesetzt der Zugriff auf die Schnittstellen ist über das *Dependency Injection Pattern* nach [Fowler, 2004] realisiert. Bei diesem Muster bekommen Werkzeuge die zu benutzende

Schnittstelle von außen injiziert, was den Austausch der Schnittstelle vereinfacht. Die Aufgabe der Werkzeuge reduziert sich auf die Visualisierung der Daten aus der übergebenen Schnittstelle, ohne ihre Herkunft zu kennen. Die DESY-Werkzeuge wurden jedoch nicht nach dem *Dependency Injection Pattern* entwickelt und müssen entsprechend überarbeitet werden. Die Einführung des Archiv-Stepper-Werkzeugs erfolgte hingegen ohne Modifikationen der vorhandenen Anwendung über den Plug-in-Erweiterungsmechanismus. Für die Synchronisation zwischen bestehenden und neuem Werkzeug, mussten diese mit Beobachtern und dazugehöriger Logik ergänzt werden.

11.4 Zusammenfassung

In diesem Abschnitt wurden die Forschungsergebnisse dieser Arbeit den Artefakten Konzepte, Modelle, Methoden und Exemplare zugeordnet und anhand der Forschungsaktivitäten Konstruieren und Evaluieren in einer realen Infrastruktur beim DESY eingesetzt und überprüft. Ausgewählte Artefakte dieser Arbeit wurden zusätzlich bei den Pilotpartnern HPA und HHLA sowie auf den Konferenzen Automation 2012 [Reiswich, 2012a], INDIN 2012 [Reiswich und Fay, 2012] sowie der Fachtagung Modellierung 2012 [Reiswich, 2012b] vorgestellt und mit dem Fachpublikum diskutiert. Architekturkonzepte für Plug-in-basierte Systeme wurden auf der Konferenz ICSE 2010 [Kornstädt und Reiswich, 2010b] sowie in der Fachzeitschrift IEEE Software [Kornstädt und Reiswich, 2010a] vorgestellt. Der Workshop „Informationsmodellierung in technischen Anlagen“ auf der SWA 2011 half zusätzlich, die Konzepte, Modelle und Methoden dieser Arbeit zu überprüfen. Im nächsten Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und ein Ausblick auf mögliche anschließende Themengebiete gegeben.

Kapitel 12

Zusammenfassung und Ausblick

Die Entwicklung von Leitstandsystemen wurde in der Vergangenheit überwiegend aus einer ingenieurwissenschaftlichen Sichtweise vorangetrieben und hat zu einem hohen Automatisierungsgrad technischer Anlagen geführt. Eine vollständige Automatisierung aller prozesstechnischer Aufgaben ist bislang nicht möglich, ihre Steuerung und Überwachung erfordern nach wie vor gezielte Eingriffe durch Operateure. Diesen fehlen gerade in kritischen Situationen geeignete Werkzeuge, die Operateure dabei unterstützen, den gegenwärtigen Zustand der Anlage schnell zu erfassen und Maßnahmen einzuleiten, die den Ausfall der Anlage verhindern. Bei der *human-centered automation* rücken Operateure deutlich mehr in den Mittelpunkt des Prozessgeschehens und übernehmen eine aktive Rolle. Dabei ist bislang nicht eindeutig geklärt, nach welchen Prinzipien die Aufgabenverteilung zwischen Mensch und Maschine erfolgen soll. Die Lösungsansätze reichen von rein technologiezentriert, mit Schwerpunkt auf Verbesserungen der Konfiguration und Integrität der Hardware, bis anwenderorientiert mit Schwerpunkt auf Verbesserungen der Gebrauchsqualität der Leitstandsoftware.

Die Prozesskomplexität in großen Industrieanlagen führt dazu, dass technologiezentrierte Ansätze gerade in kritischen Situationen oft versagen oder nicht den Kern der Leitstandprobleme adressieren. Anlagenbetreiber suchen daher nach Möglichkeiten, ihre Leitstandsysteme besser auf die Bedürfnisse der Operateure anzupassen und dadurch die Anlagenverfügbarkeit zu erhöhen. Jegliche Bemühungen, die eingesetzten Leitstandsysteme zu verbessern,

führen zu Veränderungen in der Software und in den Arbeitsabläufen der Operateure. Dies erhöht das Unfallrisiko durch Fehlbedienung und Fehlfunktion, infolgedessen Anlagenausfälle drohen. Die Modernisierung vorhandener Leitstandsysteme hat folglich Auswirkungen auf die Anwenderebene und die eingesetzten Systeme. In dieser Arbeit wurde die Frage aufgegriffen, wie vorhandene Leitstandsysteme besser auf die Arbeitssituationen der Operateure angepasst werden können, ohne dabei bewährte Arbeitsabläufe und Systeme massiv zu verändern. Diesem Ziel nähert sich die vorliegende Arbeit wie folgt:

- Anhand einer detaillierten fachlichen Analyse nach WAM und einer Literaturrecherche wurden zu Beginn dieser Arbeit wiederkehrende Leitstandprobleme aufgezeigt. Ihre Ursache wurde mit Hilfe der Arbeitspsychologie auf ein technologiezentriertes und für Operateure ungeeignetes Design der Leitstandsoftware zurückgeführt. In dieser Arbeit wurde die technologiezentrierte Sichtweise durch eine anwenderorientierte Sichtweise ergänzt. Diese berücksichtigt stärker die kognitiven Fähigkeiten und das Expertenwissen der Operateure. Daraus resultieren Leitstand-Werkzeuge, die Operateuren die Beherrschung der Komplexität in technischen Anlagen erleichtern. Diese Aspekte beantworten die erste Forschungsfrage dieser Arbeit. Sie zeigen den Beitrag auf, den eine anwenderorientierte Sichtweise in der Automatisierungstechnik leisten kann, um systematisch relevante Leitstandprobleme zu erkennen und dazu passende Lösungen zu erarbeiten.

- Die in dieser Arbeit aufgezeigten anwenderorientierten Lösungen benötigen ein strukturähnliches Anlagenmodell. In dieser Arbeit wurden Anforderungen an Anlagenmodelle herausgearbeitet und mit aktuellen Standards verglichen. Mit dem OPC-UA-Standard wurde ein für diese Arbeit geeigneter Kandidat identifiziert. Anhand des Dimensionenmodells nach WAM wurde verdeutlicht, dass für diese Arbeit nicht der gewählte Standard relevant ist. Entscheidend ist, dass dieser ein flexibles und erweiterbares Metamodell mit einer Modellierungssprache bietet, die es ermöglicht, Anlagenmodelle nicht nur an aktuelle, sondern auch an zukünftige Anforderungen anzupassen. Damit wurden softwaretechnische Aspekte der zweiten Forschungsfrage beantwortet und mit

expliziten Anlagenmodellen eine architektonische Anforderung an eine anwenderorientierte Leitstandmodernisierung formuliert.

- Leitstandsysteme werden aufgrund wechselnder technologischer und wirtschaftlicher Anforderungen häufig angepasst. Sie drohen dadurch, zu Legacy-Systemen zu verfallen, in denen Änderungen mit hohem Aufwand verbunden und teilweise gänzlich ausgeschlossen sind. Vor einer Modernisierung müssen daher Leitstandsysteme ihre Fähigkeit zur Evolution wiedererlangen. Dazu wurden in dieser Arbeit Eigenschaften evolutionsfähiger Systeme benannt und ausgewählte Reengineering-Maßnahmen vorgestellt, wie Legacy-Leitstandsysteme ihre Evolutionsfähigkeit wiedererlangen können. Auf dieser Basis wurden aktuelle Vorschläge der OPC-Foundation zur evolutionären Einführung neuer Leitstandfunktionalität untersucht. Diese Vorschläge wurden der Softwaremigration zugeordnet und für das Vorhaben dieser Arbeit als ungeeignet ermessen, da sie nur unzureichend bereits getätigte Investitionen der Anlagenbetreiber in Leitstandsysteme, Unternehmensdaten und Ausbildung ihrer Mitarbeiter berücksichtigen. Um diese Aspekte stärker zu berücksichtigen, wurden in dieser Arbeit technische, ökonomische und organisatorische Einflussfaktoren auf Softwaresysteme untersucht, auf die Leitstandmodernisierung übertragen und daraus Minimalanforderungen an eine Leitstandmodernisierung abgeleitet.

  Leitstandsysteme sind aus ökonomischer Sicht probate Modernisierungs-Kandidaten, müssen jedoch auf technischer Ebene geeignete Schnittstellen aufweisen, über die sie mit anderen Systemen Daten und Dienste austauschen können. Sie müssen zudem analysierbar, erweiterbar und änderbar sein, damit Modernisierungsmaßnahmen technisch möglich sind. Die Modernisierung vorhandener Systeme muss schließlich die begrenzte Offenheit der Anlagenbetreiber und Operateure für Veränderungen berücksichtigen und bewährte Handlungsmuster und Arbeitsabläufe weitgehend erhalten. Dazu wurde in diesem Kapitel das minimalinvasive Vorgehen eingeführt und die damit verbundenen Chancen für Anlagenbetreiber aus anwenderorientierter und technischer Sicht herausgearbeitet. Damit wurde in dieser Arbeit abschließend die zweite

Forschungsfrage beantwortet, welche softwaretechnischen, ökonomischen und organisatorischen Anforderungen Anlagenbetreiber erfüllen müssen, um eine anwenderorientierte Leitstandmodernisierung zu ermöglichen.

- Die Einführung neuer Leitstandfunktionalität impliziert meist eine Veränderung vorhandener Systeme und routinierter Arbeitsabläufe der Operateure. Statt ganze Leitstandsysteme oder große Teile davon auszutauschen oder tiefgreifend zu verändern, wurden in dieser Arbeit evolutionäre Ansätze vorgestellt. Diese führen neue Leitstandfunktionalität inkrementell ein und berücksichtigen stärker den Lernprozess der Operateure mit neuer Software. Dieser Lernprozess wurde zusätzlich durch die Softwareerweiterung unterstützt, bei der neue Leitstand-Werkzeuge vorhandene ergänzen. Zur Verdeutlichung dieser Vorgehensweise wurde die Entwurfsmetapher *Modernisierung der Altstadt* eingeführt. Diese Metapher skizziert das minimalinvasive Vorgehen dieser Arbeit, bei dem vorhandene Leitstandsysteme behutsam und mit wenigen Eingriffen modernisiert werden. Diese methodischen Ansätze beantworten die dritte Forschungsfrage, wie neue Leitstandfuktionalität in vorhandene Systeme eingeführt werden kann, ohne dabei bewährte Arbeitsabläufe der Operateure massiv zu verändern.

- Das minimalinvasive Vorgehen auf der anwenderorientierten Ebene bedarf auf der softwaretechnischen Ebene geeigneter Konzepte und einer passenden Architektur. Dazu wurde die klassische Drei-Schichten-Architektur zur Konstruktion modellbasierter und modularer Anwendungen vorgestellt und um fachliche Schnitte und Subsysteme erweitert. Für die Einführung neuer Funktionalität wurden Plug-in-basierte Erweiterungsmechanismen vorgestellt, die Änderungen und Erweiterungen mit lokalen Auswirkungen unterstützen. Die Plug-in-Ansätze müssen durch weitere Architekturkonzepte begleitet werden, um eine langfristige Evolutionsfähigkeit der neuen Leitstandfunktionalität zu ermöglichen. Dazu wurden auf der Präsentationsschicht das Dekorierer-Muster und auf der funktionalen Schicht ein generisches Modell mit Spezialmodellen erarbeitet. Zusammengenommen beantworten diese Aspekte die vierte

Forschungsfrage und präsentieren softwaretechnische Konzepte, die neue Leitstandkomponenten mit wenigen Eingriffen und Seiteneffekten in vorhandene Systeme einführen.

Damit die neue und vorhandene Leitstandfunktionalität interagieren und sich gegenseitig ergänzen kann, wurden auf der Präsentationsschicht der Mediator-Ansatz und auf der funktionalen Schicht Messaging-Konzepte zur losen Kopplung (heterogener) Systeme vorgestellt. Die mit Messaging-Konzepten einhergehende Komplexität wurde durch eine Datenzugriffsschicht eingegrenzt. Damit wurde die letzte Forschungsfrage beantwortet und Ansätze vorgestellt, die neue und vorhandene Leitstandkomponenten mit wenigen Eingriffen miteinander verknüpfen, damit diese sich gegenseitig ergänzen können.

- Für die Realisierung der konzeptionellen Lösungsansätze dieser Arbeit wurde die anwenderorientierte und technische Sichtweise der Leitstandmodernisierung zusammengeführt. Am Beispiel des Kernsystems *Anlagenmodell* und des Spezialsystems *Alarmierung* wurde die Angemessenheit dieser Ansätze für ihren Verwendungszweck demonstriert. Zusätzlich wurden diese Ansätze auf die Leitstandaufgaben *Datenzugriff* und *Historischer Datenzugriff* übertragen und evaluiert. Dabei wurde gezeigt, dass der minimalinvasive Modernisierungsansatz dieser Arbeit an den entscheidenden Schlüsselstellen ausreichend flexible und erweiterbare Konzepte bietet. Diese weisen das Potential auf nicht nur aktuelle, sondern auch zukünftige Anforderungen an neue Leitstandfunktionalität zu unterstützen.

Die minimalinvasive und anwenderorientierte Leitstandmodernisierung resultiert aus dem Wunsch der Anlagenbetreiber und Operateure, die Modernisierung ihrer Systeme mit wenigen Eingriffen und negativen Auswirkungen auf den Anlagenbetrieb durchzuführen. Die Innovation dieser Arbeit besteht darin, dass sie die langjährigen Investitionen der Anlagenbetreiber in Systeme, Unternehmensdaten und Ausbildung ihrer Mitarbeiter berücksichtigt. Diese Systeme und das Know-How der Mitarbeiter sind für Anlagenbetreiber wichtig, jegliche Veränderungen der Systeme und Arbeitsabläufe bergen Risiken und gefährden den Anlagenbetrieb. In dieser Arbeit wurden methodische und

softwaretechnische Konstruktionsprinzipien aufgezeigt, wie neue IT-Lösungen in vorhandene Systeme eingeführt werden können, ohne den Anlagenbetrieb massiv zu stören. Der Ansatz dieser Arbeit bietet somit für Anlagenbetreiber eine kleinschrittige und risikoreduzierte Alternative zu invasiven Ansätzen wie Neuentwicklung oder dem Wechsel zum Standardsystem.

Die Realisierung einer minimalinvasiven Vorgehensweise ist mit einigen Nachteilen verbunden, die im folgenden vorgestellt und dahingehend diskutiert werden, wie weitere Forschungsarbeit diese Nachteile reduzieren kann:

- **Parallelbetrieb:** Vorhandene und neue Leitstandfunktionalität implementieren häufig unterschiedliche Schnittstellen und Modelle. Die minimalinvasive Vorgehensweise führt oft zu Kompromissen, in denen alte und neue Konzepte parallel betrieben werden müssen. So bedingt beispielsweise der Erhalt bestehender Server-Funktionalität die Entwicklung einer weiteren Plattform, in der die neue Leitstandfunktionalität enthalten ist. Entwickler müssen sich dadurch wiederkehrend für Wartungs- und Erweiterungsarbeiten in unterschiedliche Technologien und Konzepte einarbeiten. Dies erhöht den Aufwand und reduziert die Verständlichkeit der Software. Entwickler können dieser Problematik prinzipiell durch umfangreiche Reengineering-Maßnahmen begegnen und vorhandene Systeme ganzheitlich auf die neuen Konzepte umstellen. Dazu müssten sie vorhandene Leitstandsysteme massiv überarbeiten, was nicht konform zum in dieser Arbeit vorgeschlagenen Vorgehen ist.

- **Werkzeugvielfalt:** In dieser Arbeit wurden neue Werkzeuge im Rahmen einer Softwareerweiterung zusätzlich zu den bestehenden Werkzeugen vorgestellt. Bei der Evaluation dieses Ansatzes hat sich gezeigt, dass mit jedem neuen Werkzeug die Übersicht der Operateure kontinuierlich abnahm. Sie waren in bestimmten Arbeitssituationen verunsichert, ob und welche unterstützenden Werkzeuge ihnen zur Verfügung stehen. Ceaparu et al. haben bei ihren Untersuchungen festgestellt, dass fehlende, oder in Menüs und Untermenüs versteckte und dadurch schwer zu findende Funktionalität, die dritthäufigste Ursache für Frustration der Anwender ist [Ceaparu et al., 2004, S. 342ff]. Um diese Probleme zu reduzieren, existieren Ansätze, die Werkzeuge entsprechend den

Aufgaben der Anwender zusammenstellen und visualisieren [vgl. z. B. Metzger und Polaków 2008; Reiswich 2012a]. Wechselt der Anwender seine Aufgabe, wechseln auch die dafür benötigten Werkzeuge. Eine nähere Untersuchung dieser Ansätze kann signifikant zur Verbesserung der Werkzeug-Übersicht beitragen.

- **Erstellung und Pflege der Anlagenmodelle:** Anlagenmodelle bilden die Basis für eine anwenderorientierte Leitstandmodernisierung. In vielen Unternehmen fehlen Anlagenmodelle, wodurch sie zunächst erstellt und anschließend gepflegt werden müssen. In der Literatur existieren Vorschläge, wie diese Modelle auf Basis vorhandener Unternehmensdaten entstehen können [vgl. z. B. Urbas und Doherr 2011; Schleipen 2008; Reiswich und Fay 2012]. Diese Unternehmensdaten wurden ursprünglich ohne die Absicht erfasst, für Anlagenmodelle wiederverwendet zu werden. Sie müssen daher für neue Anwendungsfälle in einer manuellen Prozedur semantisch angereichert werden, was aufwändig und fehleranfällig ist. Bei dem sog. *Internet der Dinge* melden sich Geräte per Plug&Play selbständig im Kommunikationsverbund an und stellen ihre Dienste sowie Metainformationen anderen Teilnehmern zur Verfügung [vlg. Elger und Haußner 2010; Mattern 2005; Günthner et al. 2010]. Dadurch kann die Erstellung und Pflege der Anlagenmodelle zukünftig automatisiert und somit wesentlich vereinfacht werden.

- **Minimalinvasives Vorgehen nicht in allen Dimensionen möglich:** In dieser Arbeit wurden softwaretechnische Einflüsse auf eine Leitstandmodernisierung anhand des Dimensionenmodells nach WAM mit den Dimensionen Anwendungsbereich, Handhabung & Präsentation und verwendete Technik diskutiert. Diese Dimensionen wurden mit ökonomischen, organisatorischen und technischen Einflussfaktoren auf eine Softwaremodernisierung ergänzt. Die Dimensionen und Einflussfaktoren konkurrieren bei einer Leitstandmodernisierung miteinander. Beispielsweise kann ein minimalinvasives Vorgehen in der Dimension Handhabung & Präsentation, bei der Leitstandwerkzeuge der Operateure nur wenig angepasst werden, softwaretechnisch sehr herausfordernd sein und sich entsprechend auf das Projektbudget auswirken. Ökono-

misch günstig erscheinende Alternativen, wie Neuentwicklung oder der Wechsel zum Standardsystem, können in einer gesamtwirtschaftlichen Betrachtung wiederum unrentabler ausfallen und eine Leitstandmodernisierung auch in einem stark erodierten System rechtfertigen. Für Anlagenbetreiber entsteht daraus das Dilemma, die konkurrierenden Dimensionen und Einflussfaktoren auf eine Leitstandmodernisierung in eine Balance zu bringen, die den Anlagenbetrieb minimal beeinträchtigt.

Neben den vorgestellten Ansätzen, die speziell auf die Lösungen dieser Arbeit und damit einhergehende Herausforderungen abgestimmt sind, existieren auch weiterführende Themen, die sich nahtlos an das Vorhaben dieser Arbeit anschließen können:

- **Datenorientierung:** In Industrieanlagen werden permanent Daten gemessen und persistiert. Ein Großteil der gesammelten Daten liegt die meiste Zeit brach. Sie werden beispielsweise in Auszügen zur Analyse kritischer Situationen herangezogen. Neuere Ansätze nutzen das zuvor erwähnte *Internet der Dinge*, um fortwährend den Datenbestand algorithmisch auszuwerten und beispielsweise Wartungsempfehlungen auszusprechen, noch bevor Anlagen ausfallen [Heise, 2012]. Eine kontinuierliche und systematische Datenanalyse bietet Anlagenbetreibern die Möglichkeit, präventiv einzugreifen und kritische Situationen vorzubeugen und somit die Risiken und Kosten zu senken. Damit eröffnen diese Ansätze zusätzliche Optionen, die mechanische Schadensfrüherkennung durch z. B. Schwingungsüberwachung [vgl. z. B. Schmidt, 1988] auf einer softwaretechnischen Ebene zu ergänzen und dadurch zur Erhöhung der Anlagenverfügbarkeit beizutragen.

- **Übergeordnete Leitstandfunktionen:** In der Automatisierungstechnik existiert aktuell die Tendenz, die Logik zur Steuerung und Überwachung von Anlagenteilen tiefer in die Feldebene, direkt an den Ort des Prozessgeschehens zu verlagern. Dadurch sollen aktuelle monolithische Automatisierungssysteme dezentralisiert und die Entscheidungskompetenz zur Steuerung der Anlage an autonome Geräte übertragen werden. Dies erhöht einerseits die Flexibilität und adressiert gleichzeitig die

wachsende Komplexität der Automatisierungssysteme [vgl. Feldhorst und Libert 2010; Günthner et al. 2010]. Durch die Verlagerung der Steuerungsaufgaben einzelner Anlagenteile auf die Feldebene können sie auf der Leitebene zu höherwertigen Leitstandfunktionen orchestriert werden. Dadurch haben Operateure gerade in kritischen Situationen die Möglichkeit, nicht nur einzelne Geräte, sondern komplexe Prozessabläufe zu beeinflussen und mit wenigen Handgriffen Gefahrensituationen zu entschärfen. Die höherwertigen Leitstandfunktionen bieten aber auch die Möglichkeit, feingranulare Informationen der Feld- und Leitebene für die Managementebene zu aggregieren und Anlagenbetreiber bei strategischen Entscheidungen zu unterstützen.

Diese Dissertation ist im Kontext des GeneAL-Forschungsprojektes und aus der Anforderung entstanden, die darin erarbeiteten Ansätze in bestehende Leitstandsysteme einzuführen. Für Anlagenbetreiber sind jegliche Modernisierungsvorhaben mit vielen Risiken und Aufwänden verbunden. Gleichzeitig muss die Leitstandsoftware während ihres Lebenszyklus gewartet und erweitert werden, um ihren Nutzen für das Unternehmen zu erhalten. Diese Kontroverse hat zur Entstehung dieser Arbeit geführt, mit dem Anspruch, neue Leitstandfunktionalität mit minimaler Beeinträchtigung bestehender Systeme, unternehmenskritischer Prozesse und Arbeitsabläufe der Operateure einzuführen. In dieser Dissertation wurden die wesentlichen, sich aus der Aufgabenstellung ergebenden Fragestellungen vorgestellt und auf einer konzeptionellen und methodischen Ebene diskutiert. Auf einer konstruktiven Ebene wurde die Tragfähigkeit und Angemessenheit der vorgeschlagenen Konzepte demonstriert und evaluiert. Das Ergebnis dieser Evaluation, die Diskussionen auf wissenschaftlichen Konferenzen und mit Anlagenbetreibern sowie das Feedback der Operateure lassen die Hoffnung zu, dass die Konzepte dieser Arbeit Anlagenbetreibern einen anwenderorientierten und minimalinvasiven Ansatz zur ökonomischen und risikoreduzierten Leitstandmodernisierung bieten.

Literaturverzeichnis

[Allen und Frost 1998] ALLEN, Paul ; FROST, Stuart: *Component-Based Development for Enterprise Systems*. Cambridge University Press, 1998

[ANSI/ISA-18.2 2009] ANSI/ISA-18.2: *Management of Alarm Systems for the Process Industries*. 2009

[Bäumer et al. 1996] BÄUMER, Dirk; BISCHOFBERGER, Walter R.; LICHTER, Horst ; ZÜLLIGHOVEN, Heinz: User interface prototyping — concepts, tools, and experience. In: *Proceedings of the 18th international conference on Software engineering*. Washington, DC, USA : IEEE Computer Society, 1996 (ICSE '96), S. 532–541

[Beach 1992] BEACH, Brian W.: Connecting software components with declarative glue. In: *Proceedings of the 14th international conference on Software engineering*. New York, NY, USA : ACM, 1992 (ICSE '92), S. 120–137

[Beck und Andres 2004] BECK, Kent ; ANDRES, Cynthia: *Extreme Programming Explained: Embrace Change*. 2. Edition. Addison-Wesley Professional, 2004

[Beer et al. 2013] BEER, Wolfgang; STUMMER, Heinz ; WINTERER, Mario: Maschinen wie eine App bedienen - Anforderungen an Multitouch-Bedienung in der Industrie. In: *Objekt Spektrum* (2013), Nr. 2, S. 66–70

[Bennett 1995] BENNETT, Keith H.: Legacy systems: coping with stress. In: *IEEE Software* 12 (1995), Nr. 1, S. 19–23

[Bennett und Rajlich 2000] BENNETT, Keith H. ; RAJLICH, Václav T.: Software maintenance and evolution: a roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA : ACM, 2000 (ICSE '00), S. 73–87

[Bézivin 2004] BÉZIVIN, Jean: In Search of a Basic Principle for Model Driven Engineering. In: *Upgrade* vol. 5 (2004), Nr. 2, S. 21–24

[Birsan 2005] BIRSAN, Dorian: On Plug-ins and Extensible Architectures. In: *Queue* 3 (2005), Nr. 2, S. 40–46

[Brand et al. 2004] BRAND, Klaus-Peter; BRUNNER, Christoph ; HAUDE, Jochen: Objektmodelle. In: SCHWARZ, Karlheinz (Hrsg.): *Offene Kommunikation nach IEC 61850 für die Schutz-und Stationsleittechnik*. VDE Verlag, 2004, Kapitel 5, S. 45–62

[Bratthall et al. 2002] BRATTHALL, L. G.; GEEST, R. van der; HOFMANN, H.; JELLUM, E.; KORENDO, Z.; MARTINEZ, R.; ORKISZ, M.; ZEIDLER, C. ; ANDERSSON, J. S.: Integrating Hundred's of Products through One Architecture-The Industrial IT architecture. In: *Proceedings of the 24th International Conference on Software Engineering (ICSE)* (2002)

[Breivold et al. 2007] BREIVOLD, Hongyu P.; CRNKOVIC, Ivica ; ERIKSSON, Peter: Evaluating Software Evolvability. In: *Proceedings of the 7th Conference on Software Engineering Research and Practice in Sweden*, 2007, S. 96–103

[Bäumer 1998] BÄUMER, Dirk: *Softwarestrukturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme*, Universität Hamburg, Diss., 1998

[Carroll und Carrithers 1984] CARROLL, John M. ; CARRITHERS, Caroline: Training wheels in a user interface. In: *Communications of the ACM* 27 (1984), Nr. 8, S. 800–806

[CAS 2010] CAS: *OPC Unified Architecture e-book*. `http://www.commsvr.com/uamodeldesigner/html/7cd6ebfb-3582-4ae7-8b1c-89e038d9bb66.htm`, 2010. – [Zugriff 13.12.2012]

[Ceaparu et al. 2004] CEAPARU, Irina; LAZAR, Jonathan; BESSIERE, Katie; ROBINSON, John ; SHNEIDERMAN, Ben: Determining Causes and Severity of End-User Frustration. In: *International Journal of Human-Computer Interaction* 17 (2004), Nr. 3, S. 333–356

[Chapin et al. 2001] CHAPIN, Ned; HALE, Joanne E.; KHAN, Khaled M.; RAMIL, Juan F. ; TAN, Wui-Gee: Types of software evolution and software maintenance. In: *Journal of Software Maintenance and Evolution: Research and Practice* 13 (2001), Nr. 1, S. 3–30

[CMMI Product Team 2002] CMMI PRODUCT TEAM: *Capability Maturity Model, Integration (CMMI)*. Technical Report CMU/SEI-2002-TR-011, Carnegie Mellon Software Engineering Institute, Pittsburgh, 2002. – Version 1.1

[Colazzo et al. 2008] COLAZZO, Luigi; MOLINARI, Andrea ; TOMASINI, Sara: Is new necessarily good? Testing usability of the new Office 2007 user interface. In: LUCA, Joseph (Hrsg.) ; WEIPPL, Edgar R. (Hrsg.): *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*. Vienna, Austria : AACE, 2008, S. 1371–1379

[Comella-Dorda et al. 2000] COMELLA-DORDA, S.; WALLNAU, K.; SEACORD, R.C. ; ROBERT, J.: A survey of black-box modernization approaches for information systems. In: *Software Maintenance, 2000. Proceedings. International Conference on*, 2000, S. 173–183

[Cupek et al. 2009] CUPEK, R; FOJCIK, M ; SANDE, O: Object Oriented Vertical Communication in Distributed Industrial Systems. In: *Computer Networks* (2009), S. 72–78

[Czaja und Nair 2012] CZAJA, Sara J. ; NAIR, Sankaran N.: Human factors engineering and systems design. In: SALVENDY, Gavriel (Hrsg.): *Handbook of Human Factors and Ergonomics*, John Wiley & Sons, Inc., 2012, S. 38–56

[Dalesio et al. 1994] DALESIO, Leo R.; HILL, Jeffrey O.; KRAIMER, Martin; LEWIS, Stephen; MURRAY, Douglas; HUNT, Stephan; WATSON, William; CLAUSEN, Matthias ; DALESIO, John: The experimental physics and industrial control system architecture: past, present, and future. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 352 (1994), S. 179 – 184

[Daum 2007] DAUM, Berthold: *Rich-Client-Entwicklung mit Eclipse 3.3. Anwendungen entwickeln mit Eclipse RCP, SWT, Forms, GEF, BIRT, JPA u.a.m.* dpunkt.verlag, 2007

[Davis 1993] DAVIS, Fred D.: User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. In: *International Journal of Man-Machine Studies* 38 (1993), Nr. 3, S. 475–487

[Demeyer et al. 2008] DEMEYER, Serge; DUCASSE, Stéphane ; NIERSTRASZ, Oscar: *Object-Oriented Reengineering Patterns, 1. Open-Source Edition*. Square Bracket Associates, Switzerland, 2008

[Diedrich und Mühlhause 2011] DIEDRICH, Christian ; MÜHLHAUSE, Mathias: Modellansätze für die digitale Fabrik. In: *Automatisierungstechnik 1/2011* (2011)

[DIN 19235] *Meldung von Betriebszuständen*

[Drath et al. 2008] DRATH, R; LUDER, A; PESCHKE, J ; HUNDT, L: AutomationML-the glue for seamless automation engineering. In: *Emerging Technologies and Factory Automation* (2008)

[Drath 2010] DRATH, Rainer: *Datenaustausch in der Anlagenplanung mit AutomationML*. Springer, 2010

[Drath et al. 2011] DRATH, Rainer; FAY, Alexander ; BARTH, Mike: Interoperabilität von Engineering-Werkzeugen. In: *at - Automatisierungstechnik* 59 (2011), Nr. 7, S. 451–460

[Eclipse-Foundation 2012] ECLIPSE-FOUNDATION: *E4/Event Processing*. `http://wiki.eclipse.org/E4/Event_Processing`, 2012. – [Zugriff 27.09.2012]

[EEMUA 2007] EEMUA: *Alarm Systems - A Guide to Design, Management and Procurement*. The engineering equipment and materials users association, 2007

[ELCON 2003] *The Economic Impacts of the August 2003 Blackout*. Electricity Consumers Resource Council. `http://www.elcon.org/Documents/EconomicImpactsOfAugust2003Blackout.pdf`. Version: 2003. – [Zugriff 12.06.2013]

[Elger und Haußner 2010] ELGER, Jürgen ; HAUSSNER, Carolin: Entwicklungen in der Automatisierungstechnik. In: **[Günthner und ten Hompel, 2010]**, S. 23–27

[Emigholz 1996] EMIGHOLZ, Kenneth F.: Improving the operator's capabilities during abnormal operations; observations from the control house. In: *Process Safety Progress* 15 (1996), Nr. 3, S. 154–158

[Endsley und Connors 2008] ENDSLEY, M.R. ; CONNORS, E.S.: Situation awareness: State of the art. In: *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, 2008, S. 1–4

[Enste und Mahnke 2011] ENSTE, Udo ; MAHNKE, Wolfgang: OPC Unified Architecture - Die nächste Stufe der Interoperabilität. In: *at - Automatisierungstechnik* 7 (2011), S. 397–404

[Epple 2003] EPPLE, Ullrich: Austausch von Anlagenplanungsdaten auf der Grundlage von Metamodellen. In: *atp, Automatisierungstechnische Praxis 7 (45), 61.70* (2003)

[Epple 2011] EPPLE, Ulrich: Merkmale als Grundlage der Interoperabilität technischer Systeme. In: *at - Automatisierungstechnik* 59 (2011), Nr. 7, S. 440–450

[Favre 2005] FAVRE, Jean-Marie: Foundations of Meta-Pyramids: Languages vs. Metamodels – Episode II: Story of Thotus the Baboon. In: BEZIVIN, Jean (Hrsg.) ; HECKEL, Reiko (Hrsg.): *Language Engineering for Model-Driven Software Development*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005 (Dagstuhl Seminar Proceedings 04101)

[Fay et al. 2009] FAY, Alexander; SCHLEIPEN, Miriam ; MÜHLHAUSE, Mathias: Wie kann man den Engineering-Prozess systematisch verbessern? In: *ATP-Automatisierungstechnische Praxis* 1-2 (2009)

[Feldhorst und Libert 2010] FELDHORST, Sascha ; LIBERT, Sergey: Software-Methoden für die Automatisierung. In: **[Günthner und ten Hompel, 2010]**, S. 29–37

[Fildebrandt 2012] FILDEBRANDT, Ulf: *Software modular bauen - Architektur von langlebigen Softwaresystemen- Grundlagen und Anwendungen mit OSGi und Java.* dpunkt.verlag GmbH, 2012

[Floyd 1984] FLOYD, Christiane: A systematic look at prototyping. In: *Approaches to Prototyping*. Springer-Verlag, 1984, S. 1–18

[Floyd et al. 1997] FLOYD, Christiane; KRABBEL, Anita; RATUSKI, Sabine ; WETZEL, Ingrid: Zur Evolution der evolutionären Systementwicklung: Erfahrungen aus einem Krankenhausprojekt. In: *Informatik-Spektrum* 20 (1997), S. 13–20

[Floyd et al. 1989] FLOYD, Christiane; REISIN, Fanny-Michaela ; SCHMIDT, Gerhard: STEPS to software development with users. In: GHEZZI, C. (Hrsg.) ; MCDERMID, J.A. (Hrsg.): *ESEC '89* Bd. 387. Springer Berlin Heidelberg, 1989, S. 48–64

[Fowler 2004] FOWLER, Martin: *Inversion of Control Containers and the Dependency Injection pattern*. `http://www.martinfowler.com/articles/injection.html`, 2004. – [Zugriff 14.03.2013]

[Freund 2010] FREUND, Curt: Die Instandhaltung im Wandel. In: SCHENK, Michael (Hrsg.): *Instandhaltung technischer Systeme*. Springer Berlin Heidelberg, 2010, S. 1–22

[Fröhlich et al. 2002a] FRÖHLICH, Peter; HU, Zaijun ; SCHOELZKE, Manfred: Imposing Modeling Rules on Industrial Applications through Metamodeling. In: ARISAWA, Hiroshi (Hrsg.); KAMBAYASHI, Yahiko (Hrsg.); KUMAR, Vijay (Hrsg.); MAYR, Heinrich (Hrsg.) ; HUNT, Ingrid (Hrsg.): *Conceptual Modeling for New Information Systems Technologies* Bd. 2465. Springer Berlin / Heidelberg, 2002, S. 166–182

[Fröhlich et al. 2002b] FRÖHLICH, Peter; HU, Zaijun ; SCHOELZKE, Manfred: Using UML for Information Modeling in Industrial Systems with Multiple Hierarchies. In: JÉZÉQUEL, Jean-Marc (Hrsg.); HUSSMANN, Heinrich (Hrsg.) ; COOK, Stephen (Hrsg.): *«UML» 2002 — The Unified Modeling Language* Bd. 2460. Springer Berlin / Heidelberg, 2002, S. 327–351

[Früh et al. 2009] FRÜH, K. F.; MAIER, U. ; SCHAUDEL, D.: *Handbuch der Prozessautomatisierung*. 4. Auflage. Oldenbourg Industrieverlag, 2009

[Gamma et al. 2009] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph ; VLISSIDES, John: *Entwurfsmuster*. Addison-Wesley, 2009

[García et al. 2004] GARCÍA, R; GELLE, E ; STROHMEIER, A: The Concepts of IEC 61346 Applied to a Software Architecture for Automation. In: *en.scientificcommons.org* (2004)

[García und Gelle 2006] GARCÍA, R.G. ; GELLE, E.: Applying and adapting the IEC 61346 standard to industrial automation applications. In: *Industrial Informatics, IEEE Transactions on* 2 (2006), Nr. 3, S. 185 –191

[García und Gelle 2004] GARCÍA, Rodrigo G. ; GELLE, Esther: Modelling Industrial Plant Automation Systems Using IEC 61346. In: *Proceedings of the International Conference on Intelligent Knowledge Systems, Robotics and Automation, August 16-20, Assos, Turkey*, 2004

[Gilb 1985] GILB, Tom: Evolutionary Delivery versus the "waterfall model". In: *SIGSOFT Softw. Eng. Notes* 10 (1985), Nr. 3, S. 49–61

[Gîrbea et al. 2010] GÎRBEA, A.; NECHIFOR, S.; SISAK, F. ; PERNIU, L.: Design and implementation of an OLE for process control unified architecture aggregating server for a group of flexible manufacturing systems. In: *IEEE The Institution of Engineering and Technology* 5 (2010), Nr. 4, S. 406 – 414

[Goll 2011] GOLL, Joachim: *Methoden und Architekturen der Softwaretechnik*. Vieweg+Teubner Verlag, 2011

[Gryczan 1996] GRYCZAN, Guido: *Prozeßmuster zur Unterstützung kooperativer Tätigkeit*, Universität Hamburg, Diss., 1996

[Günthner und ten Hompel 2010] GÜNTHNER, Willibald (Hrsg.) ; HOMPEL, Michael ten (Hrsg.): *Internet der Dinge in der Intralogistik*. Springer Berlin Heidelberg, 2010 (VDI-Buch)

[Günthner et al. 2010] GÜNTHNER, Willibald A.; CHISU, Razvan ; KUZMANY, Florian: Die Vision vom Internet der Dinge. In: **[Günthner und ten Hompel, 2010]**, S. 41–44

[Güttel und Fay 2008] GÜTTEL, Knut ; FAY, Alexander: Title. In: *Automatisierungstechnische Praxis* 50 (2008), Nr. 1, S. 34–39

[Hannelius et al. 2008] HANNELIUS, T; SALMENPERA, M ; KUIKKA, Seppo: Roadmap to adopting OPC UA. In: *IEEE International Conference on Industrial Informatics (INDIN 2008, DCC, Daejeon, Korea July 13-16)* (2008)

[Harel und Rumpe 2004] HAREL, David ; RUMPE, Bernhard: Modeling languages: Syntax, semantics and all that stuff. In: *IEEE Software* (2004)

[Hasselbring 2006] HASSELBRING, Wilhelm: Software-Architektur. In: *Informatik-Spektrum* 29 (2006), Nr. 1, S. 48–52

[Hatje et al. 2007] HATJE, Jan; CLAUSEN, Matthias; GERKE, Christian; MOELLER, Markus ; RICKENS, Helge: Control System Studio (CSS). In: *International Conference on Accelerator & Large Experimental Physics Control Systems* (2007)

[Haußner et al. 2010] HAUSSNER, Carolin; ELGER, Jürgen ; KUZMANY, Florian: Der Lebenszyklus heutiger Materialflusssysteme – eine Übersicht. In: **[Günthner und ten Hompel, 2010]**, S. 173–186

[Heise 2012] HEISE: *GE setzt auf Internet der Dinge*. `http://m.heise.de/newsticker/meldung/GE-setzt-auf-Internet-der-Dinge-1765486.html`, 2012. – [Zugriff 13.12.2012]

[Herczeg 2008] HERCZEG, Michael: Vom Werkzeug zum Medium: Mensch-Maschine-Paradigmen in der Prozessführung. In: *M. Grandt und A. Bauch (Hrsg.), Beiträge der Ergonomie zur Mensch-System-Integration, DGLR-Bericht* 04 (2008), S. 1–11

[Hesse und Mayr 2008] HESSE, W ; MAYR, Heinrich C.: Modellierung in der Softwaretechnik: eine Bestandsaufnahme. In: *Informatik-Spektrum* (2008)

[Hevner et al. 2004] HEVNER, Alan R.; MARCH, Salvatore T.; PARK, Jinsoo ; RAM, Sudha: Design Science in Information Systems Research. In: *MIS Quarterly* 28 (2004), Nr. 1, S. 75–105

[Higgins et al. 2011] HIGGINS, N.; VYATKIN, V.; NAIR, N.-K.C. ; SCHWARZ, K.: Distributed Power System Automation With IEC 61850, IEC 61499, and Intelligent Control. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 41 (2011), Nr. 1, S. 81 –92

[Hoc et al. 1995] HOC, Jean-Michel; CACCIABUE, Pietro C.; HOLLNAGEL, Erik ; CACCIABUE, P. C.: *Expertise and Technology: Cognition & Human-Computer Cooperation*. Lawrence Erlbaum Associates, Inc., 1995

[Hohpe und Woolf 2004] HOHPE, Gregor ; WOOLF, Bobby: *Enterprise Integration Patterns*. Addison-Wesley, 2004

[Hollender 2009] HOLLENDER, Martin: *Collaborative Process Automation Systems*. ISA, 2009

[Hollifield und Habibi 2007] HOLLIFIELD, Bill ; HABIBI, Eddie: *Alarm Management: Seven Effective Methods for Optimum Performance.* ISA, 2007

[Hollnagel et al. 1995] HOLLNAGEL, Erik; CACCIABUE, Pietro C. ; HOC, Jean-Michel: Work with technology: Some fundamental issues. In: *Expertise and technology: Cognition & human-computer cooperation*, 1995, S. 1 – 15

[Hu 2003] HU, Zaijun: Using Ontology to Bind Web Services to the Data Model of Automation Systems. In: CHAUDHRI, AkmalB. (Hrsg.); JECKLE, Mario (Hrsg.); RAHM, Erhard (Hrsg.) ; UNLAND, Rainer (Hrsg.): *Web, Web-Services, and Database Systems* Bd. 2593. Springer Berlin Heidelberg, 2003, S. 154–168

[IEEE 1219-1998] *IEEE Standard for Software Maintenance*. – 1998

[ISO 15926] *Industrial automation systems and integration—Integration of life-cycle data for process plants including oil and gas production facilities*. – Geneva, Switzerland, 2003

[ISO 9000] *ISO 9000 - Quality management*. `http://www.iso.org/iso/iso_9000`, . – [Zugriff: 16.08.2013]

[ISO/IEC 9126] *Software engineering — Product quality*. – 2001

[Johansson et al. 2006] JOHANSSON, O.; POP, A. ; FRITZSON, P.: Engineering Design Tool Standards and Interfacing Possibilities to Modelica Simulation Tools. In: *5th International Modelica Conference* (2006)

[Karagiannis und Höfferer 2006] KARAGIANNIS, D. ; HÖFFERER, P.: Metamodels in Action: An Overview. In: *J. Filipe, B. Shishkov and M. Helfert (Hrsg.): ICSOFT 2006 - First International Conference on Software and Data Technologies*, Insticc Press, Setúbal, 2006, S. IS–27 – IS–36

[Karagiannis und Kuhn 2002] KARAGIANNIS, D ; KUHN, H: Metamodelling platforms. In: *Lecture Notes in Computer Science* (2002)

[Karwowski 2012] KARWOWSKI, Waldemar: The discipline of human factors and ergonomics. In: SALVENDY, Gavriel (Hrsg.): *Handbook of Human Factors and Ergonomics*, John Wiley & Sons, Inc., 2012, S. 3–37

[Kay 1977] KAY, Alan: Microelectronics and the personal computer. In: *Scientific American* (1977), S. 231–244

[Keller 2000] KELLER, Wolfgang: The bridge to the new town — a legacy system migration pattern. In: *Proceedings of EuroPLoP*, 2000

[Kindsmüller 2006] KINDSMÜLLER, Martin C.: *Trend-Literacy – zur Interpretation von Kurvendarstellungen in der Prozessführung*, Humboldt-Universität zu Berlin, Diss., 2006

[KKS 1978] *Kraftwerk-Kennzeichensystem: KKS-Richtlinien*. 1978

[Kornstädt 2002] KORNSTÄDT, Andreas: *Ein Einschubsystem für die musikwissenschaftliche Analyse*, Universität Hamburg, Diss., 2002

[Kornstädt und Reiswich 2010a] KORNSTÄDT, Andreas ; REISWICH, Eugen: Composing Systems with Eclipse Rich Client Platform Plug-Ins. In: *IEEE SOFTWARE* (2010), S. 78–81

[Kornstädt und Reiswich 2010b] KORNSTÄDT, Andreas ; REISWICH, Eugen: Staying afloat in an expanding sea of choices: emerging best practices for eclipse rich client platform development. In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE)*, ACM, 2010, S. 59–67

[Kurz 2008] KURZ, Hans: Alarmmanagement-Ziele, Erfahrungen, Nutzen. In: *atp – Automatisierungstechnische Praxis* (2008), Nr. 2, S. 86–90

[Lange et al. 2010] LANGE, Jürgen; IWANITZ, Frank ; BURKE, Thomas J.: *OPC von Data Access bis Unified Architecture*. 4. Auflage. VDE Verlag, 2010

[Lehman 1984] LEHMAN, M. M.: Program Evolution. In: *Information Processing and Management* 20 (1984), Nr. 1, S. 19–36

[Lehman und Ramil 2002] LEHMAN, Meir M. ; RAMIL, Juan F.: Software Evolution and Software Evolution Processes. In: *Annals of Software Engineering* 14 (2002), S. 275–309

[Lehnhoff et al. 2011] LEHNHOFF, Sebastian; MAHNKE, Wolfgang ; ROHJANS, Sebastian: IEC 61850 based OPC UA Communication-The Future of Smart Grid Automation. In: *17th Power Systems Computation Conference* (2011)

[Lilienthal 2008] LILIENTHAL, Carola: *Komplexität von Softwarearchitekturen*, Universität Hamburg, Diss., 2008

[Liu et al. 2005] LIU, J.; LIM, K.W.; HO, W.K.; TAN, K.C.; TAY, A. ; SRINIVASAN, R.: Using the OPC standard for real-time process monitoring and control. In: *Software, IEEE* 22 (2005), Nr. 6, S. 54 – 59

[Liu et al. 2003] LIU, Jun; LIM, Khiang W.; HO, Weng K.; TAN, Kay C.; SRINIVASAN, R. ; TAY, A.: The intelligent alarm management system. In: *Software, IEEE* 20 (2003), Nr. 2, S. 66 – 71

[Ludewig 2003] LUDEWIG, J: Models in software engineering-an introduction. In: *Software and Systems Modeling* (2003)

[Lunze 2003] LUNZE, Jan: *Automatisierungstechnik*. Oldenbourg Verlag München Wien, 2003

[Mackiewicz 2006] MACKIEWICZ, R.E.: Overview of IEC 61850 and Benefits. In: *Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES*, 2006, S. 623 –630

[Magnussen 2012] MAGNUSSEN, Sönke: *Qualitätssicherung bei der Restrukturierung eines 14 Millionen LOC ABAP-Systems*. C1 WPS WAM-Workshop, 2012

[Mahnke 2011] MAHNKE, Wolfgang: *Persönliches Interview am 20 Juli 2011 auf der Software-Architektur-Tagung, Hamburg*. 2011

[Mahnke et al. 2011] MAHNKE, Wolfgang; GOSSLING, Andreas; GRAUBE, Markus ; URBAS, Leon: Information modeling for middleware in automation. In: *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, 2011, S. 1–7

[Mahnke et al. 2009] MAHNKE, Wolfgang; LEITNER, Stefan-Helmut ; DAMM, Matthias: *OPC Unified Architecture*. Springer, 2009

[March und Smith 1995] MARCH, Salvatore T. ; SMITH, Gerald F.: Design and natural science research on information technology. In: *Decision Support Systems* 15 (1995), Nr. 4, S. 251–266

[Marples und Kriens 2001] MARPLES, D. ; KRIENS, P.: The Open Services Gateway Initiative: an introductory overview. In: *Communications Magazine, IEEE* 39 (2001), Nr. 12, S. 110 –114

[Mattern 2005] MATTERN, Friedemann: Die technische Basis für das Internet der Dinge. In: FLEISCH, Elgar (Hrsg.) ; MATTERN, Friedemann (Hrsg.): *Das Internet der Dinge*. Springer Berlin Heidelberg, 2005, S. 39–66

[McAffer und Lemieux 2005] MCAFFER, Jeff ; LEMIEUX, Jean-Michel: *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications*. Addison-Wesley Longman, 2005

[McAffer et al. 2010] MCAFFER, Jeff; VANDERLEI, Paul ; ARCHER, Simon: *OSGi and Equinox: Creating Highly Modular Java Systems*. Addison Wesley Pub Co Inc, 2010

[Mein 2013] MEIN, Christian: *Vereinfachung der Analyse von historischen Daten in Softwareleitständen*, Universität Hamburg, Diplomarbeit, 2013. – noch nicht abgeschlossene Arbeit

[Metzger und Polaków 2008] METZGER, M ; POLAKÓW, G: A study on appropriate plant diagram synthesis for user-suited HMI in operating control. In: *Engineering Interactive Systems* (2008)

[Meyer 1986] MEYER, Bertrand: Genericity versus Inheritance. In: *OOPSLA Proceedings* (1986)

[Meyer 1997] MEYER, Betrand: *Object-oriented software construction, 2nd Edition*. Prentice Hall Inc., 1997

[Milliner et al. 1995] MILLINER, Steven; BOUGUETTAYA, Athman ; PAPAZOGLOU, Mike: A scalable architecture for autonomous heterogeneous database interactions. In: *Proceedings of the International Conference on Very Large Data Bases* (1995)

[Mumaw et al. 2000] MUMAW, Randall J.; ROTH, Emilie M.; VICENTE, Kim J. ; BURNS, Catherine M.: There Is More to Monitoring a Nuclear Power Plant than Meets the Eye. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 42 (2000), Nr. 1, S. 36–55

[Nachreiner et al. 2006] NACHREINER, Friedhelm; NICKEL, Peter ; MEYER, Inga: Human factors in process control systems: The design of human machine interfaces. In: *Safety Science* 44 (2006), Nr. 1, S. 5–26

[Nagel und Westfechtel 2003] NAGEL, Manfred ; WESTFECHTEL, Bernhard: *Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen: Symposium (Forschungsberichte DFG)*. Wiley-VCH, 2003

[NAMUR 2003] NAMUR: *Arbeitsblatt NA 102: Alarm Management*. 2003

[Nock 2004] NOCK, Clifton: *Data Access Patterns*. Addison-Wesley, 2004

[Obst et al. 2012] OBST, M.; DOHERR, F.; URBAS, L; DRUMM, O. ; BAUER, C.: Integriertes HMI-Engineering. In: *VDI-Berichte 2171*, 2012, S. 227 –230

[OMG 2003] OMG: *MDA Guide Version 1.0.1*. Object Management Group, 2003

[Otto und Schuler 2000] OTTO, Michael ; SCHULER, Norbert: *Fachliche Services: Geschäftslogik als Dienstleistung für verschiedene*

Benutzungsschnittstellen-Typen, Universität Hamburg, Diplomarbeit, 2000

[Pappert 2012] PAPPERT, Sebastian: *Ein Werkzeug zur Beherrschung von Alarmen in einer anlagenmodellbasierten Leitstandsoftware*, Universität Hamburg, Diplomarbeit, 2012

[Parasuraman et al. 2000] PARASURAMAN, R.; SHERIDAN, T.B. ; WICKENS, C.D.: A model for types and levels of human interaction with automation. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 30 (2000), Nr. 3, S. 286 –297

[Parasuraman und Riley 1997] PARASURAMAN, Raja ; RILEY, Victor: Humans and Automation: Use, Misuse, Disuse, Abuse. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 39 (1997), Nr. 2, S. 230–253

[Perleth 2007] PERLETH, M.: Minimal-invasive Chirurgie. In: *Der Chirurg* 78 (2007), S. 511–518

[Polke 1994] POLKE, M.: *Prozeßleittechnik*. 2. Auflage. R.Oldenbourg Verlag München Wien, 1994

[PSOTF 2003] *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*. U.S.-Canada Power System Outage Task Force. `https://reports.energy.gov/B-F-Web-Part1.pdf`. Version: 2003. – [Zugriff 12.06.2012]

[Pätzold 2011] PÄTZOLD, Torsten: *Persönliches Interview mit DESY-Operateur, geführt vom Verfasser.* Hamburg, 20 Januar 2011

[Rajlich und Bennett 2000] RAJLICH, V.T. ; BENNETT, Keith H.: A staged model for the software life cycle. In: *Computer* 33 (2000), Nr. 7, S. 66 –71

[Ransom et al. 1998] RANSOM, J.; SOMERVILLE, I. ; WARREN, I.: A method for assessing legacy systems for evolution. In: *Software Maintenance and Reengineering, 1998. Proceedings of the Second Euromicro Conference on*, 1998, S. 128–134

[Rausch 2000] RAUSCH, Andreas: Software evolution in componentware - a practical approach. In: *Proceedings of the Australian Software Engineering Conference*, 2000, S. 13 –21

[Reiswich 2012a] REISWICH, Eugen: Unterstützung von Leitstandarbeitsplätzen durch Anlageninformationsmodelle. In: *VDI-Berichte 2171*, 2012, S. 303 – 307

[Reiswich 2012b] REISWICH, Eugen: Verbesserung der Arbeitssituationen im Leitstand durch Anlageninformationsmodelle. In: *Fachtagung Modellierung 2012, Bamberg* (2012)

[Reiswich und Fay 2012] REISWICH, Eugen ; FAY, Alexander: Strategy for the amendment of plant information models by means of OPC UA. In: *IEEE International Conference on Industrial Informatics* (2012)

[Riebisch und Bode 2009] RIEBISCH, Matthias ; BODE, Stephan: Software-Evolvability. In: *Informatik-Spektrum* 32 (2009), S. 339–343

[Riehle 1997] RIEHLE, Dirk: Composite design patterns. In: *Proceedings of the 12th ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications*, ACM, 1997 (OOPSLA '97), S. 218–228

[Riehle und Züllighoven 1995] RIEHLE, Dirk ; ZÜLLIGHOVEN, Heinz: A pattern language for tool construction and integration based on the tools and materials metaphor. In: *Pattern Languages of Program Design* (1995), S. 9–42

[Runde und Fay 2008] RUNDE, S. ; FAY, A.: A data exchange format for the engineering of building automation systems. In: *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, 2008, S. 303 –310

[Salvendy 2012] SALVENDY, Gavriel (Hrsg.): *Handbook of Human Factors and Ergonomics, 4. Edition*. John Wiley & Sons, Inc., 2012

[Sauer und Ebel 2007a] SAUER, O. ; EBEL, M.: Engineering of production monitoring and control systems. In: *Technische Universität/Fakultät für*

Informatik und Automatisierung : Computer science meets automation, Vol. 1, 52 (2007), S. 237–244

[Sauer 2013] SAUER, Olaf: lnformationstechnik für die Fabrik der Zukunft. In: SCHOLZ-REITER, Bernd (Hrsg.): *Industrie Management 1/2013: Vierte industrielle Revolution*. GITO Verlag, 2013, S. 11–14

[Sauer und Ebel 2007b] SAUER, Olaf ; EBEL, Miriam: Plug-and-work von Produktionsanlagen und übergeordneter Software. In: KOSCHKE, Rainer (Hrsg.); HERZOG, Otthein (Hrsg.); RÖDIGER, Karl-Heinz (Hrsg.) ; RONTHALER, Marc (Hrsg.): *GI Jahrestagung (2)* Bd. 110, GI, 2007 (LNI), S. 331–338

[Sauter 2005] SAUTER, T.: Integration aspects in automation - a technology survey. In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* Bd. 2, 2005

[Sauter 2007] SAUTER, T.: The continuing evolution of integration in manufacturing automation. In: *Industrial Electronics Magazine, IEEE* 1 (2007), Nr. 1, S. 10 –19

[Schleburg et al. 2012] SCHLEBURG, Markus; CHRISTIANSEN, Lars; FAY, Alexander ; THORNHILL, Nina F.: Reduction of alerts in automated systems based on a combined analysis of process connectivity and alarm logs. In: BOGLE, Ian David L. (Hrsg.) ; FAIRWEATHER, Michael (Hrsg.): *Proceedings of the 22nd European Symposium on Computer Aided Process Engineering*, 2012

[Schleipen 2008] SCHLEIPEN, M: OPC UA supporting the automated engineering of production monitoring and control systems. In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (2008)

[Schlick et al. 2013] SCHLICK, Jochen; STEPHAN, Peter ; ZÜHLKE, Detlef: Produktion 2020. Auf dem Weg zur 4. industriellen Revolution. In: SCHEER, August-Wilhelm (Hrsg.): *Industrie 4.0*. IMC AG, 2013

[Schmidt 1988] SCHMIDT, Reinhard: *Die Anwendung von Zustandsbeobachtern zur Schwingungsüberwachung und Schadensfrüherkennung auf mechanische Konstruktion*, Institut für Mechanik der Ruhr-Universität Bochum, Diss., 1988

[Schmitz und Epple 2007] SCHMITZ, S. ; EPPLE, U.: Automatisierte Projektierung von HMI-Oberflächen. In: *VDI-Berichte Nr. 1980*, 2007, S. 1–12

[Schwarz und Börcsök 2000] SCHWARZ, M. H. ; BÖRCSÖK, J.: An Investigation and Survey on the Future Direction of OLE for Process Control (OPC). In: *International Journal of Computers* 1 (2000), Nr. 4

[Schwarz et al. 2012] SCHWARZ, Tobias; HEILIG, Mathias; BUTSCHER, Simon; MÜLLER, Jens ; REITERER, Harald: Die Gestaltung der Arbeitsumgebung von Operatoren (k-)ein Platz für Innovationen? In: *Automation 2012* (2012). – VDI Verlag

[Schwarzer und Krcmar 2004] SCHWARZER, Bettina ; KRCMAR, Helmut: *Wirtschaftsinformatik: Grundzüge der betrieblichen Datenverarbeitung*. 3. Auflage. Schäffler-Poeschel Verlag Stuttgart, 2004

[Schöner 2010] SCHÖNER, Michael: *Persönliches Interviews mit DESY-Operateur, geführt vom Verfasser.* Hamburg, 24 März 2010

[Seidewitz 2003] SEIDEWITZ, E.: What models mean. In: *Software, IEEE* 20 (2003), Nr. 5, S. 26 – 32

[Sheridan 1996] SHERIDAN, T.B.: Allocating functions among humans and machines. In: BEEVIS, D. (Hrsg.); ESSENS, P. (Hrsg.) ; SCHUFFEL, H. (Hrsg.): *Improving Function Allocation for Integrated Systems Design*, Wright–Patterson Airforce Base, 1996, S. 7–27

[Sheridan 2012] SHERIDAN, Thomas B.: Human Supervisory Control. In: SALVENDY, Gavriel (Hrsg.): *Handbook of Human Factors and Ergonomics*, John Wiley & Sons, Inc., 2012, S. 990 – 1015

[Shneiderman 2003] SHNEIDERMAN, Ben: Promoting universal usability with multi-layer interface design. In: *Proceedings of the 2003 conference on Universal usability*. New York, NY, USA : ACM, 2003 (CUU '03), S. 1–8

[Siltanen und Pärnänen 2006] SILTANEN, Pekka ; PÄRNÄNEN, Antti: Comparison of data models for plant lifecycle information management. In: *Proceedings of the 13th ISPE International Conference on Concurrent Engineering: Leading the Web in Concurrent Engineering*, 2006, S. 346–353

[Smolander 2002] SMOLANDER, K.: Four metaphors of architecture in software organizations: finding out the meaning of architecture in practice. In: *International Symposium on Empirical Software Engineering, (ISESE)*, 2002, S. 211–221

[Sneed et al. 2010] SNEED, Harry M.; WOLF, Ellen ; HEILMANN, Heidi: *Softwaremigration in der Praxis*. 1. Auflage. dpunkt.verlag, 2010

[Sneed und Sneed 2003] SNEED, Harry M. ; SNEED, Stephan H.: *Web-basierte Systemintegration: So überführen sie bestehende Anwendungssysteme in eine moderne Webarchitektur*. Vieweg + Teubner, 2003

[Sommerville 2001] SOMMERVILLE, Ian: *Software Engineering*. 6. Auflage. Addison-Wesley, 2001

[Son und Yi 2010] SON, Mai ; YI, Myeong-Jae: A study on OPC specifications: Perspective and challenges. In: *Strategic Technology (IFOST), 2010 International Forum on*, 2010, S. 193 –197

[Stachowiak 1973] STACHOWIAK, Herbert: *Allgemeine Modelltheorie*. Springer-Verlag, Wien, 1973

[Strasser et al. 2005] STRASSER, Thomas; ZOITL, Alois; AUINGER, Franz ; SÜNDER, Christoph: Towards Engineering Methods for Reconfiguration of Distributed Real-Time Control Systems Based on the Reference Model of IEC 61499. In: MAŘÍK, Vladimír (Hrsg.); BRENNAN, Robert (Hrsg.) ; PĚCHOUČEK, Michal (Hrsg.): *Holonic and Multi-Agent Systems for Manufacturing* Bd. 3593. Springer Berlin Heidelberg, 2005, S. 165–175

[Strube et al. 2012] STRUBE, Martin; KÜHL, Ingmar; HOLM, Thomas; FAY, Alexander; MÜHLFED, Reinhard ; FIGALIST, Helmut: Modellierung von Kommunikationsschnittstellen bestehender Automatisierungslösungen in Moderniesierungsprojekten auf Basis von Signallisten. In: *VDI-Berichte 2171*, 2012, S. 95 –98

[Sullivan und Notkin 1992] SULLIVAN, Kevin J. ; NOTKIN, David: Reconciling environment integration and software evolution. In: *ACM Trans. Softw. Eng. Methodol.* 1 (1992), Nr. 3, S. 229–268

[Szyperski 2002] SZYPERSKI, Clemens: *Component Software: Beyond Object-Oriented Programming, 2nd ed.* Addison-Wesley, 2002

[Szyperski 2003] SZYPERSKI, Clemens: Component technology: what, where, and how? In: *Proceedings of the 25th International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2003 (ICSE '03), S. 684–693

[Theng et al. 2010] THENG, Yin-Leng; TING, Eng ; TAO, Xuehong: From Microsoft Word 2003 to Microsoft Word 2007: Design Heuristics, Design Flaws and Lessons Learnt. In: FORBRIG, Peter (Hrsg.); PATERNÓ, Fabio (Hrsg.) ; MARK PEJTERSEN, Annelise (Hrsg.): *Human-Computer Interaction* Bd. 332. Springer Boston, 2010, S. 289–292

[Thiel et al. 2010] THIEL, Klaus; MEYER, Heiko ; FUCHS, Franz: *MES - Grundlage der Produktion von morgen*. 2. überarbeitete Auflage. Oldenbourg Verlag München, 2010

[TU Hamburg-Harburg 2012] TU HAMBURG-HARBURG: *Einweihung des Hauptgebäudes der TU Hamburg – weithin sichtbares Wahrzeichen mit großer Strahlkraft*. `http://intranet.tuhh.de/aktuell/pressemittelung_einzeln.php?id=8422`, 2012. – Pressemitteilung, 24.08.2012

[Urbas und Doherr 2011] URBAS, L ; DOHERR, F: autoHMI: a model driven software engineering approach for HMIs in process industries. In: *IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (2011), S. 627 – 631

[Urbas et al. 2011] URBAS, Leon; HENNIG, Stefan; HAGER, Henning; DOHERR, Falk ; BRAUNE, Annerose: Towards context adaptive HMIs in process industries. In: *IEEE International Conference on Industrial Informatics (INDIN)* (2011), S. 244 – 249

[Urbas et al. 2012] URBAS, Leon; ZIEGLER, Jens ; DOHERR, Falk: Produktergonomie in der Prozessautomatisierung. In: *Zeitschrift für Arbeitswissenschaft* (2012)

[Van Tan et al. 2009] VAN TAN, Vu; YOO, Dae-Seung ; YI, Myeong-Jae: Device Integration Approach to OPC UA-Based Process Automation Systems

with FDT/DTM and EDDL. In: HUANG, De-Shuang (Hrsg.); JO, Kang-Hyun (Hrsg.); LEE, Hong-Hee (Hrsg.); KANG, Hee-Jun (Hrsg.) ; BEVILACQUA, Vito-antonio (Hrsg.): *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence* Bd. 5755. Springer Berlin / Heidelberg, 2009, S. 1001–1012

[Vicente et al. 2001] VICENTE, Kim J.; ROTH, Emilie M. ; MUMAW, Randall J.: How do operators monitor a complex, dynamic work domain? The impact of control room technology. In: *International Journal of Human-Computer Studies* 54 (2001), Nr. 6, S. 831 – 856

[Weber 2006] WEBER, Klaus H.: *Inbetriebnahme verfahrenstechnischer Anlagen: Praxishandbuch mit Checklisten und Beispielen*. 3. Auflage. Springer, 2006

[Wickens und Hollands 2000] WICKENS, Christopher D. ; HOLLANDS, Justin G.: *Engineering Psychology and Human Performance*. 3. Edition. Prentice Hall Inc., 2000

[Wiendahl et al. 2007] WIENDAHL, H.-P.; ELMARAGHY, H.A.; NYHUIS, P.; ZÄH, M.F.; WIENDAHL, H.-H.; DUFFIE, N. ; BRIEKE, M.: Changeable Manufacturing - Classification, Design and Operation. In: *CIRP Annals - Manufacturing Technology* 56 (2007), Nr. 2, S. 783–809

[Winter 2008] WINTER, R: Design science research in Europe. In: *European Journal of Information Systems* (2008)

[Wütherich et al. 2008] WÜTHERICH, Gerd; HARTMANN, Nils; KOLB, Bernd ; LÜBKEN, Matthias: *Die OSGi Service Platform*. Dpunkt Verlag, 2008

[Züllighoven 2005] ZÜLLIGHOVEN, Heinz: *Object-Oriented Construction Handbook*. Dpunkt Verlag, 2005

[Züllighoven et al. 1998] ZÜLLIGHOVEN, Heinz; BÄUMER, Dirk; BLEEK, Wolf-Gideon ; LILIENTHAL, Carola: *Das objektorientierte Konstruktionshandbuch*. Dpunkt Verlag, 1998. – der Printausgabe beigelegete elektronische Version

Abbildungsverzeichnis

1.1 Vereinfachtes Modell der objektorientierten Softwareentwicklung nach WAM 14

2.1 Lebenszyklus technischer Anlagen 24

2.2 Automatisierungspyramide 26

2.3 Ein-Sensor-eine-Anzeige-Prinzip 28

2.4 Vermischung analoger und digitaler Anzeigen in Leitständen . 29

2.5 Leitstand Hamburger Hochbahn 30

2.6 Prozessleitsystem 30

2.7 CMTB Prozessbild 32

2.8 Alarmtabelle 34

2.9 Kurvenbild 35

4.1 Meta-Modellierung Ebenenmodelle 63

4.2 Dimensionenmodell der Softwareentwicklung nach WAM . . . 72

4.3 Abstrakte Darstellung des OPC-UA-Metamodells 75

4.4 Ausschnitt eines OPC-UA-basierten Anlagenmodells 76

4.5 Visualisierung eines Kryogenik-Anlagenmodells 77

5.1 Modellbasiertes Alarm-Werkzeug 82

5.2 Einordnung der Ergebnisse im WAM-Dimensionenmodell . . . 86

6.1 Architekturerosion in Softwaresystemen 88
6.2 Phasenmodell des Softwarelebenszyklus 91
6.3 OPC UA Wrapper 99
6.4 OPC UA Proxy 100
6.5 Bewertung von Legacy-Systemen anhand technischer und Business-Faktoren 103

7.1 Softwareredefinition vs. Softwareerweiterung 117
7.2 Entwurfsmetapher *Modernisierung der Altstadt* 120
7.3 Einordnung minimalinvasive Leitstandmodernisierung aus anwenderorientierter Sicht im WAM-Dimensionenmodell 123

8.1 Generische Architektur für eine Metamodellierungs-Plattform 127
8.2 Erweiterte Drei-Schichten-Architektur 128
8.3 Einführung des Kernsystems *Anlagenmodell* 131
8.4 Historische Entwicklung modularer Systeme 132
8.5 Interaktionsmöglichkeiten zwischen Plug-ins 132
8.6 Anlagenmodell- und Alarmbaum-Werkzeuge 135
8.7 Mögliche Alternativen bei der Erweiterung eines Anlagenmodells mit neuer Funktionalität 138
8.8 Konzept generisches Modell mit Spezialmodellen 141
8.9 Generisches Modell mit Spezialmodellen 143
8.10 Neue und vorhandene Leitstandfunktionalität 146
8.11 Mechanismen zur Werkzeugkopplung 148
8.12 Verteilte Variante des Nachrichten-Bus-Ansatzes 152
8.13 Message Endpoints 154
8.14 Messaging Gateway 155

8.15 Einordnung minimalinvasive Leitstandmodernisierung aus technischer Sicht im WAM-Dimensionenmodell 157

9.1 Minimalinvasive Einführung neuer Leitstandfunktionalität . . 161

10.1 DESY Alarmsystem-Architektur 164

10.2 Fachliche Struktur der Anlagenmodell-Funktionalität 166

10.3 Plug-in-basiertes Anlagenmodell-Werkzeug 167

10.4 Plug-in-basierte Anlagenmodell-Funktionalität auf der funktionalen Schicht 169

10.5 Fachliche Struktur der Alarm-Funktionalität 172

10.6 Plug-in-basiertes Alarm-Werkzeug 173

10.7 Kombination neuer und vorhandener Alarm-Werkzeuge 176

10.8 Serverseitige Plug-ins für die Alarmierung 177

11.1 Research Framework 184

11.2 Modellbasiertes Werkzeug für die Projektierung 191

11.3 Modellbasiertes Werkzeug für die historische Datenanalyse . . 194

A.1 Die Eclipse Platform 235

A.2 Control System Studio Architektur 237

Listings

10.1 Anlagenmodell-Werkzeug Erweiterung 168
10.2 Alarm-Dekorierer 174
10.3 Aktualisierung der Alarmdarstellung im Alarmbaum-Werkzeug 174
10.4 Ergänzungen zur Werkzeuginteraktion im Alarmtabellen-Werkzeug 175
10.5 Alarmbehandlung im Alarmservice 178
10.6 Alarmmodell 179
10.7 Alarm-Datenzugriffsschicht 179
11.1 Typ-basierte Konfiguration 192

Anhang A

Verwendete Technologien

A.1 Rich Clients

In dieser Arbeit wird die *Eclipse Rich Client Platform* als technologische Entwicklungsplattform eingesetzt. Bevor die damit verbundenen Konzepte erläutert werden, ist es zuvor hilfreich die Konzepte *Rich-Client*, *Fat-Client* und *Thin-Client* zu differenzieren.

Bei *Rich-Clients* steht die benutzerfreundliche Handhabung sowie eine reichhaltige Funktionalität einer Anwendung im Vordergrund. Hierzu gehören ein hoher Grad an Interaktionsmöglichkeiten mit der Oberfläche wie z. B. die Unterstützung von Drag&Drop, Tastaturkürzeln und Möglichkeiten zur Anpassung der Oberfläche an eigene Bedürfnisse. Darüber hinaus eignen sich Rich-Clients für die Verarbeitung großer Datenmengen, den Einsatz auf mobilen Geräten und für die Zusammenarbeit zwischen den Anwendern [McAffer und Lemieux, 2005, S. 1ff]. Diese und weitere Vorteile werden deutlich, wenn man sich die Alternativen zu Rich-Clients ansieht, die Fat-Clients und Thin-Clients.

Ein *Fat-Client* besteht aus den Elementen API (Application Programming Interface), der Hardwareabstraktionsschicht (engl. Hardware Abstraction Layer, HAL) sowie der Hardware selbst. Fat-Clients verfügen über eigene Speicher- und Verarbeitungskapazitäten, daher wird der Hauptteil der Aufgabenverarbeitung auf dem Client ausgeführt. Auf dem Server werden in der Regel nur

die Zugriffe auf die Datenbanken durchgeführt. Fat-Clients übernehmen somit vollständig die Berechnung der Daten, ihre Aufbereitung und die grafische Präsentation. Zu den Vorteilen von Fat-Clients gehören die hohe Performanz bei der Verarbeitung großer Datenmengen, gute Benutzerführung sowie die Möglichkeiten zum Offline-Betrieb. Zu den Nachteilen zählen die Plattformabhängigkeit sowie der hohe Aufwand bei der Softwareverteilung und Wartung [Sneed und Sneed, 2003, S. 162ff].

Bei einem *Thin-Client* werden alle Daten von einem Server generiert und vom Client lediglich dargestellt. Die Daten werden dabei meist in Form von HTML Dateien über das Intra- bzw. Internet übertragen. Thin-Clients präsentieren ihre Inhalte grundsätzlich statisch. Unter Verwendung zusätzlicher Komponenten wie z. B. Java-Applets oder Active-X-Komponenten kann die Funktionalität der Thin-Clients um dynamische Prozesse ergänzt werden. Zu den Vorteilen von Thin-Clients gehören die Plattformunabhängigkeit, einfache Bedienung sowie der geringe Aufwand bei der Softwareverteilung und Wartung. Sie sind von jedem Rechner mit einer Internetverbindung und Browser zu bedienen und bieten dadurch zusätzlich eine geografische Freiheit. Ihre Grenzen erreichen Thin-Clients bei der Verarbeitung großer Datenmengen. Diese Grenzen sind durch eine beschränkte Datentransferrate sowie Serverrechenkapazität gesetzt. Thin-Clients sind zusätzlich an eine permanente Serververbindung gebunden und eignen sich nicht für einen Offline-Betrieb. Schließlich ist auch die Benutzerführung für komplexe Anwendungen eingeschränkt [Sneed und Sneed, 2003, S. 167-169].

Rich-Clients übernehmen die Vorteile der Fat-Clients, große Datenmengen zu verarbeiten, ermöglichen eine gute Benutzerführung sowie einen Offline-Betrieb. Gleichzeitig bieten sie mit der Plattformunabhängigkeit und der verbesserten Softwareverteilung und Wartung die Vorteile von Thin-Clients [Daum 2007, S. 2-5; McAffer und Lemieux 2005, S. 1ff].

A.1.1 Eclipse Rich Client Platform

Bis zur Version 2.1 wurde Eclipse hauptsächlich als reine Entwicklungsumgebung (engl. Integrated Development Environment, IDE) für Java benutzt. Das

Komponentenmodell von Eclipse hatte jedoch das Potenzial, zu einer allgemeinen Plattform für Desktop-Anwendungen zu werden. Daraufhin wurde die Eclipse Architektur überarbeitet. Seit der Version 3.0 kann die Eclipse Plattform für Rich-Client-Anwendungen verwendet werden. Die Rich-Client-Platform bildet dabei einen kleinen Kern, der dafür benötigt wird, Plug-ins auszuführen. Die ursprüngliche Eclipse-IDE wurde vom Kern gelöst und ist jetzt eine spezielle Rich-Client-Anwendung, die auf der Eclipse RCP aufsetzt.

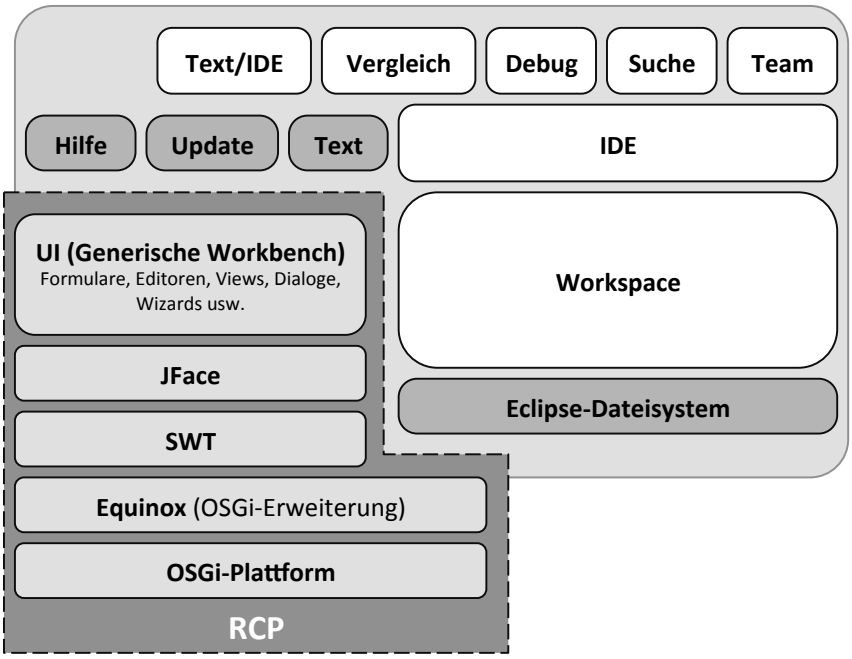


Abbildung A.1: Die Eclipse Plattform (Quelle: Daum, 2007, S. 12)

Der größte Teil der Eclipse RCP (dunkelgrauer Bereich in Abbildung A.1) besteht aus Komponenten für die Benutzeroberfläche. SWT (Standard Widget Toolkit) und JFace stellen die Basisfunktionalität für grafische Oberflächen (GUI) zur Verfügung. SWT sorgt zusätzlich dafür, dass alle Rich-Client-Applikationen auf allen gängigen Betriebssystemen eine native Darstellung haben. Auf den GUI-Bibliotheken baut die generische Workbench auf. Diese verwaltet u.a. Menü- und Werkzeugleisten, Editoren, Views und Wizards. Generisch bedeutet hierbei, dass diese Funktionalität in jede Rich-Client-Anwendung eingebunden werden kann. Seit Eclipse 3.0 ist auch die OSGi-Platform[1] Bestandteil der Eclipse RCP (siehe A.1.2).

[1]OSGi = Open Service Gateway Initiative

Im hellgrauen Bereich der Abbildung A.1 sind optionale Komponenten dargestellt, die auch in eigenen Rich-Client-Applikationen verwendet werden können. Die Eclipse-IDE Funktionalität wird jedoch in der Regel nicht für eigene Applikationen benötigt [Daum 2007, S. 11-13; McAffer und Lemieux 2005, S. 13ff].

A.1.2 Eclipse und OSGi

Die *Open Service Gateway Initiative Alliance* (OSGi Alliance) ist ein unabhängiges Konsortium von Technologieunternehmen mit dem Ziel der Standardisierung von Diensten, die auf lokalen Netzwerken und eingebetteten Geräten zum Einsatz kommen. Das OSGi-Framework ergänzt die Rich Client Platform um eine serviceorientierte und komponentenbasierte Infrastruktur. Diese Infrastruktur gewährleistet die Interoperabilität von Anwendungen und Diensten, die die OSGi-Spezifikation erfüllen. OSGi-Dienste können auf verschiedenen OSGi-konformen Servern ausgeführt werden. In Eclipse werden OSGi-Dienste auf der Open-Source-Implementierung *Equinox* ausgeführt [Wütherich et al., 2008]. Mit OSGi können Plug-ins anderen Plug-ins Dienste in einer zentralen Service-Registratur anbieten und nutzen. Die Abhängigkeit zwischen Dienstanbieter und Nutzer kann dabei minimal und lediglich auf ein Interface beschränkt sein. Im OSGi-Kontext wird statt dem Begriff Plug-in, der Begriff Bundle verwendet, obwohl zwischen diesen Begriffen nur ein geringer Unterschied besteht [McAffer und Lemieux, 2005, S. 18]. Neben diesen Konzepten führt OSGi in RCP-Anwendungen das sog. *Hot Plugging* ein. Dadurch können Plug-ins prinzipiell im laufenden Betrieb installiert, aktualisiert und ausgetauscht werden, ohne die Anwendung neu zu starten [vgl. Wütherich et al. 2008; McAffer et al. 2010].

A.2 Control System Studio

Das Control System Studio[2] (CSS) ist eine beim DESY entwickelte Leitstandsoftware, die auf der Eclipse RCP-Technologie basiert und in dieser Arbeit für

[2]CSS Website: `http://css.desy.de/content/index_eng.html`

die Realisierung und Evaluation der erarbeiteten Konzepte verwendet wurde. Die grundlegende Idee beim CSS besteht darin, eine modulare Plattform für Leitstandsoftware zu entwickeln, die Unternehmen an eigene Bedürfnisse und Systeme anpassen oder für neue Anwendungsfälle erweitern können [Hatje et al., 2007]. Dazu abstrahiert CSS von konkreten Datenquellen und Formaten und bietet entsprechende Schnittstellen, über die Unternehmen ihre Daten anbinden können.

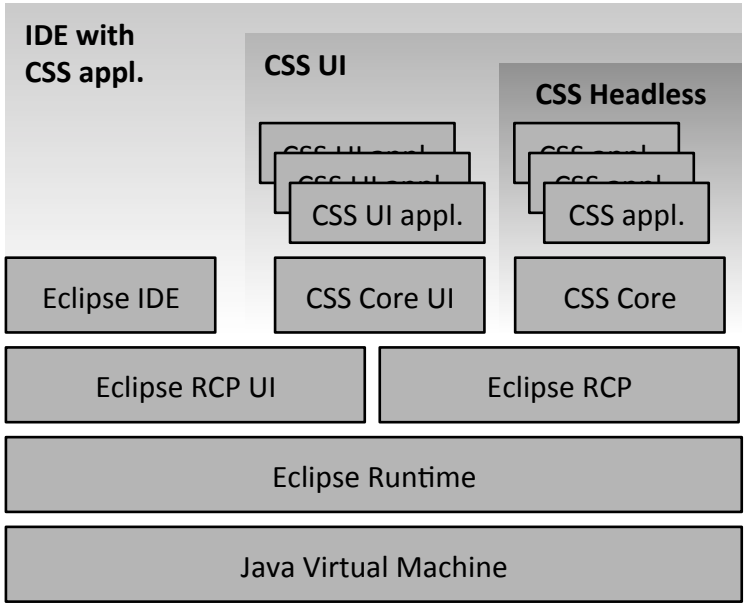


Abbildung A.2: Control System Studio Architektur (Quelle: Hatje et al., 2007)

CSS besteht konzeptionell aus den Bestandteilen *CSS Core*, mit Plug-ins für gemeinsame Leitstandfunktionen wie z. B. Echtzeitdatenübertragung, und den *CSS-Anwendungen*, die individuell entwickelt und in die Plattform installiert werden können (Abbildung A.2). Zu den generischen und in anderen Unternehmen wiederverwendbaren Anwendungen gehören *Probe*, *Data Browser* und *PV-Table*, die zur Datenvisualisierung dienen. Darüber hinaus existiert mit dem Synoptic Display Studio (SDS) ein Projektierungs-Werkzeug für die Erstellung und Pflege der Prozessbilder [Hatje et al., 2007].

Anhang B

Abbildungstabelle der OPC-UA-Metamodell-Elemente

OPC UA	**Klassendiagramm Abbildung 4.4**
BaseNode	PlantNode
Object	PlantSection
ObjectType	PlantSectionType
Variable	ProcessVariable
VariableType	ProcessVariableType
Attribute	ProcessVariableAttribute
InformationModel	PlantModel
BrowseService	PlantModelService

Tabelle B.1: Gegenüberstellung der OPC-UA-Konzepte und ihre Realisierung im Klassendiagramm der Abbildung 4.4 (Quelle: eigene Erstellung)