Vermittlung kontextbasierter Sichten in mobilen, ubiquitären Systemen

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
Dirk Bade
aus Hamburg

Hamburg 2013

Gutachter:
Prof. Dr. W. Lamersdorf, Universität Hamburg
Prof. Dr. C. Linnhoff-Popien, Ludwig-Maximilians-Universität München

Tag der Disputation: 21. Oktober 2013

Der Kopf muss sich vor dem Herz verneigen.

Fernöstliche Weisheit

Zusammenfassung

Der stetige technologische Fortschritt der vergangenen Jahre brachte eine Reihe von Technologien hervor, die heute als Wegbereiter für Mark Weisers Vision des Ubiquitous Computing gelten. Hierzu gehören insbesondere die Sensornetze (engl. Wireless Sensor Networks), Möglichkeiten zur Identifikation von Objekten mittels Radiowellen (engl. Radio Frequency Identification) sowie Entwicklungen im Bereich der mobilen Datenverarbeitung und Kommunikation. Auf Basis dieser Technologien entstand eine Vielzahl von Geräten, die nicht nur über die Kapazitäten verfügen, größere Mengen an Informationen zu verarbeiten, sondern darüber hinaus auch die Fähigkeit besitzen, zu kommunizieren und neue Daten, insbesondere Kontextdaten, zu erheben.

Eine wesentliche Implikation dieses Entwicklungstrends ist der Umstand, dass die Menge an Daten, die erhoben, verarbeitet und kommuniziert werden, in Zukunft weiter ansteigen wird. Auch ist anzunehmen, dass die Daten verschiedenster Quellen miteinander verknüpft und in Beziehung gesetzt werden, um umfassende Einblicke in Vorgänge der physischen, logischen und sozialen Umwelt zu erhalten. Anwendungen, die derlei Daten verwenden, ordnet man der Klasse der kontextbasierten Anwendungen zu. Diese Klasse zeichnet sich dadurch aus, dass Kontextdaten die Basis für eine adäquate Präsentation von Informationen oder eine Verhaltensadaption der Anwendung selbst darstellen. Grundlage hierfür ist eine detaillierte und vor allem auf bestimmte Aspekte der Umwelt fokussierte Sicht.

Zur Realisierung einer solchen kontextbasierten Sicht bedarf es der Vermittlung von Kontextdaten zwischen Produzenten (z.B. Sensoren) und Konsumenten (z.B. Menschen oder Anwendungen), was neben der reinen Datenerhebung und -übertragung vor allem auch die Datenverarbeitung und -abfrage umfasst. Insbesondere im Rahmen mobiler, ubiquitärer Systeme birgt dies eine ganze Reihe von Herausforderungen: Hierzu gehören nicht nur der Umgang mit der Heterogenität der Daten, der Hard- und Software sowie der vorherrschenden Kommunikationstechnologien, sondern auch die potentielle Mobilität der Produzenten und Konsumenten. Sie haben i.d.R. eingeschränkte Verarbeitungsressourcen und unterschiedliche Anforderungen an den Zugriff und die Verarbeitung von Kontextdaten und sollen letztendlich höherwertige Informationen aus umfangreichen, global verteilten Datenströmen in nahezu Echtzeit ableiten können, was gänzlich neue Anforderungen an die Klasse der kontextbasierten Anwendungen und damit zusammenhängend an die Kontextdatenvermittlung stellt.

Der zentrale Forschungsaspekt dieser Dissertation beschäftigt sich daher mit der Untersuchung und Erarbeitung von Konzepten zur Vermittlung kontextbasierter Sichten zwischen Konsumenten und Produzenten in mobilen, ubiquitären Systemen, welche den Entwurf und die Entwicklung kontextbasierter Anwendungen unterstützen sollen. Als Ergebnis wurden (nicht-) funktionale Anforderungen an einen Vermittlungsprozess erhoben, im Rahmen dessen Kontextdaten nach Maßgabe der Produzenten und Konsumenten in eine für sie geeignete Repräsentation überführt werden können. Unter Berücksichtigung solcher Anforderungen wurden wesentliche Prozessaufgaben identifiziert und die Architektur eines Vermittlers entworfen, welcher alle notwendigen Verarbeitungsschritte durch Ausführung klientenspezifischer Orchestrierungsanweisungen abwickelt. Durch Berücksichtigung etablierter Standards der Sensor Web Enablement-Initiative des Open Geospatial Consortiums sowie der Auto-ID-Standards von EPCglobal bzw. GS1 ist ein Vermittler zudem in der Lage, mit existierenden Diensten aus den Bereichen des Sensor Webs sowie des Internet der Dinge zu interoperieren und höherwertige Aufgaben, wie z.B. das Erstellen kontextbasierter Sichten, für die (mobilen) Klienten, zu übernehmen. Um die prinzipielle Umsetzbarkeit des Vermittlungskonzepts aufzuzeigen, wurde zudem eine exemplarische, prototypische Implementierung entwickelt, welche mittels einer Vielzahl autonomer, aktiver Komponenten den kompletten Vermittlungsprozess unter den Prämissen möglichst hoher Flexibilität, Erweiterbarkeit, Skalierbarkeit, Robustheit etc. realisiert.

Unter Einsatz eines solchen Vermittlers können somit Kontextdaten durch Konsumenten einerseits leichter aufgefunden und Daten verschiedenster Quellen miteinander verknüpft werden, und andererseits kann die Verarbeitungslogik und -last in die Infrastruktur verschoben werden, sodass ressourcenschwache Geräte am Rand des Systems entlastet werden.

Abstract

Ongoing technological progress throughout the last years yielded a multitude of new technologies pioneering Mark Weiser's vision of Ubiquitous Computing. Among the most important are wireless sensor networks, object identification by means of radio waves (RFID) as well as new developments in the area of mobile data processing and communication. Based on these technologies an abundance of new devices have been developed which not only posses the capability to process substantial amounts of data, but also have the ability to communicate and to collect new data, most notably context data.

A major implication of this trend is that the amount of data processed and communicated will increase dramatically throughout the next years. Moreover, it is expected that data stemming from diverse sources are linked and related in order to get comprehensive insights into the incidents within the physical, logical and social environment. Applications using this data belong to the class of context-based applications. This class is characterized by the fact that context data is the foundation for adequate presentations of information or adaptations of the application's behaviour. For this purpose, a detailed view, specialized on certain aspects of the environment, is required.

In order to realize such a context-based view, context data must be mediated between producers (e.g. sensors) and consumers (e.g. users or applications) which includes, aside from the sole data collection and dissemination, aspects of data processing and retrieval. Particularly in mobile and ubiquitous systems, several challenges have to be met: In addition to the heterogeneity of data, hard- and software, as well as communication technologies, the potential mobility of producers and consumers has to be taken into account. Such mobile clients often posses limited processing capabilities, have diverse requirements regarding the retrieval and processing of context data, and are ultimately supposed to infer high-level information out of vast, globally distributed data streams in near real time. This yields to entirely new requirements regarding the class of context-based applications and the interrelated mediation of context data.

The central research aspect of this thesis is therefore focused on the analysis and development of concepts for the mediation of context-based views between producers and consumers in mobile and ubiquitous systems, in order to ease and support the design and development of context-based applications. As a result, fundamental (non-) functional requirements for a mediation process that transforms raw context data into individually tailored context-based views for producers and consumers have been analyzed. Based on this requirements catalog the main process tasks have been identified and the architecture of a mediator realizing all processing steps by execution of client-specific orchestration procedures has been developed. By considering existing standards of the Sensor Web Enablement initiative by the Open Geospatial Consortium as well as the Auto-ID standards by EPCglobal or GS1 respectively, such a mediator is also able to interoperate with existing services in the Sensor Web and the Internet of Things in order to fulfill higher-level tasks like the creation of context-based views for (mobile) clients. In order to demonstrate the feasibility of the mediation concept a prototypical example implementation has been developed, realizing the complete mediation process by means of a multitude of autonomous yet cooperating active components to gain a high degree of flexibility, extensibility, scalability, robustness, etc.

By using such a mediator, context data can be discovered by consumers more easily, data stemming from diverse sources can be linked and combined, the processing logic and load can be shifted into the infrastructure and resource-constraint devices at the system's edge can be unburdened.

Danksagung

Während meiner Tätigkeit als wissenschaftlicher Mitarbeiter und insbesondere während der Zeit, in der diese Arbeit niedergeschrieben wurde, haben mich zahlreiche Menschen begleitet und mich und diese Arbeit in positiver Hinsicht beeinflusst. Allen voran möchte ich Herrn Lamersdorf danken. Er gab mir alle Freiheiten in Forschung & Lehre, war immer zur Stelle, wenn ich Rat brauchte, führte mich in die Welt akademischer Gepflogenheiten ein und wurde seiner Rolle als Doktorvater mehr als gerecht.

Mein Dank gilt auch zahlreichen Kollegen, die mich im Laufe der Zeit begleitet haben: Christian und Kristof für die Sorgfalt, Christopher für erfrischende Erkenntnisse, Kai für technische Unterstützung, Wolfgang und Thomas für das Eröffnen neuer Horizonte, Fabian für das Vertrauen und die Abwechslung und Peter und Steffen für die gute Zusammenarbeit in Forschungsprojekten. Ganz großer Dank gilt Alex und Lars, ohne die mein Traum ein eben solcher geblieben wäre und die ein hervorragendes Vorbild waren und bleiben. Ich danke Ante, meinem Doktorbruder, der mir stets das Gefühl gab, nie alleine dazustehen, und Sonja, die eine hervorragende Lehrmeisterin war und mir beibrachte, die Untiefen akademischen Handelns zu umschiffen.

Insbesondere möchte ich auch meinem Kollegen Volker danken, einerseits für schöne Diskussionen und interessante Einblicke in aktuelle (universitäre) Geschehnisse, andererseits auch für die technische Unterstützung in all den Jahren. Der liebste Dank gebührt Anne. Nicht nur, weil sie alle Minenfelder und Abkürzungen kennt, sondern vor allem weil sie jeden Tag den Mühlenrädern Schwung geben kann. Auch möchte ich den zahlreichen Studenten danken, deren wissenschaftliche Arbeiten ich begleiten durfte, allen voran Jan, Hai-Minh, Julian und Daniel.

In all den Jahren gab es auch eine Reihe von Menschen außerhalb meines universitären Kontextes, die mir und meiner Arbeit in der ein oder anderen Weise beigestanden haben. Dank Telse, Bruno, Tanja, Silke, Anne und ganz besonders Sarih ist die Bibliothek auch im Herzen ein zweites Zuhause geworden. Kappe, Wolf und Sarah sei neben vielen anderen Dingen für offene Ohren und Rettungsanker gedankt und Katharina für das Gefühl nicht gänzlich verschollen zu sein. Mayya danke ich für die passende Untermalung und die Hilfe, den richtigen Rhythmus zu finden. All meinen Begleitern in all den Jahren auf dem Weg nach Santiago sei für andere Perspektiven gedankt. Und ich danke Iris, sie hat mir schlussendlich wieder gezeigt, was im Leben wirklich wichtig ist und wofür es sich zu leben lohnt.

Und natürlich gebührt mein Dank auch meiner Familie: Jürgen und Regine für das Verständnis und die Geduld, Mel, Kerstin, Jan und insbesondere Frank für einen Ort des Rückzugs und Juliana, Philipp, Finn, Nils, Konsti und Max für Dinge, die ich schon lange nicht mehr gemacht hatte. Ilse für den Willen und das Herz und Jutta, zu der ich immerzu aufblicke, für alles.

Dirk Bade

Inhaltsverzeichnis

1. Einleitung

Der durchschnittliche Wahrnehmungs-, Interessens- und Wirkungsbereich der Menschen, insbesondere in industrialisierten Ländern, hat sich in den letzten Jahrzehnten immens erweitert. Waren bis Ende der 80er Jahre Zeitungen, Funk und Fernsehen mit redaktionell gepflegten Inhalten die primären Informationsquellen, wandelte sich mit der Verbreitung des Internets Ende der 90er Jahre die Rolle des Menschen vom reinen Medien-Konsumenten hin zu einer eher gestalterischen Form. Es war für Jedermann nicht nur ein Leichtes nahezu beliebige Informationen, die genau den eigenen Interessen entsprachen, weltweit sekundenschnell zuzugreifen, sondern selbst Informationen einer breiten Öffentlichkeit in Form privater Homepages, Blogs, Foren, Galerien, sozialer Netzwerke etc. zugänglich zu machen. Auf diese Weise entstand neben professionellen Anbietern von Inhalten ein weites Feld von Informationsquellen aus privater Hand.

Auch heute noch wird ein Großteil der öffentlich verfügbaren Informationen im Internet von Menschen generiert und durch Menschen konsumiert [IDC12]. Doch der stetige technologische Fortschritt brachte in den vergangenen Jahren bereits eine Reihe von Entwicklungen hervor, die es nunmehr auch Dingen erlauben, als aktive Teilnehmer im Internet zu agieren, Informationen zu publizieren und zu konsumieren. Im einfachsten Fall werden Dinge zunächst mit einer digitalen Identität sowie mit der Fähigkeit diese zu kommunizieren, versehen. Einen Schritt weiter gehen Dinge, die zusätzlich ihre Umwelt mittels Sensoren wahrnehmen und diese Beobachtungen periodisch veröffentlichen können. Den nächsten logischen Evolutionsschritt beschreiten dann die „intelligenten Dinge“, die eine digitale Identität besitzen, ihre Umwelt wahrnehmen, mit anderen Dingen kommunizieren sowie ggf. eigene Schlussfolgerungen aus ihren Beobachtungen ziehen und auf die Umwelt einwirken können, zum Beispiel um sich den Bedürfnissen ihrer Benutzer anzupassen [BCL+04].

Durch die Fähigkeit, den eigenen Kontext wahrzunehmen und zu publizieren, ergeben sich gänzlich neue Anwendungsmöglichkeiten für derlei Dinge. Im öffentlichen Bereich werden sie bereits heute eingesetzt, um frühzeitig vor drohenden Tsunamis, Erdbeben, Waldbränden oder Lawinen zu warnen, Verkehrsflüsse zu steuern oder Mikroklima-Phänomene zu untersuchen. Im Wirtschaftssektor dienen sie zum Beispiel der Überwachung von Lieferketten, der kontinuierlichen Inventarisierung sowie der Kontrolle der Betriebsfähigkeit von Maschinen. Auch im privaten Bereich haben derartige Dinge Einzug gehalten, beispielsweise bei der Heimautomatisierung, bei neuartigen Unterhaltungsangeboten oder als nützliche Assistenten im Alltag.

Aufgrund der voranschreitenden Verbreitung von Dingen, die ihren Kontext wahrnehmen und publizieren können, und der damit einhergehenden Verwe-

bung der physischen mit der virtuellen Welt, wird eine immer feingranularere Überwachung der Umwelt möglich. Ungeachtet der Bedenken hinsichtlich der Privatsphäre von Benutzern, deren Kontext sich unausweichlich mit dem Kontext von Dingen überschneidet, reichen die Visionen von intelligenten Dingen, über intelligente Häuser und Städte bis hin zu einem intelligenten Planeten, der durch ein feinmaschiges Netz an Sensoren eine „elektronische Haut“ verliehen bekommt und sein eigenes Wohlbefinden überwachen kann [Onl99].

Für die Durchdringung der Welt und unseres Alltags mit datenverarbeitenden Geräten prägte Mark Weiser bereits Anfang der 90er Jahre den Begriff des *Ubiquitous Computing* [Wei91]. Im Rahmen seiner Vision der „unsichtbaren Datenverarbeitung“ werden die heutigen Computer als generische Werkzeuge in den Hintergrund treten und durch eine Vielzahl intelligenter, spezialisierter Alltagsgegenstände ersetzt, die ihre Umwelt wahrnehmen, miteinander kooperieren und sich so an die Bedürfnisse der Benutzer adaptieren, dass deren Aufmerksamkeitsfokus verstärkt auf den Zielen ihrer Tätigkeiten, und weniger auf den Mitteln und Wegen diese zu erreichen, liegen kann.

Der stetige technologische Fortschritt brachte in den vergangenen Jahren bereits eine Reihe von Technologien hervor, die heute als Wegbereiter für Weisers Vision gelten und bei denen Kontextdaten eine besondere Schlüsselrolle einnehmen. Hierzu gehören insbesondere die Sensornetze (*Wireless Sensor Networks*) [SMZ07], die Identifizierung von Objekten mittels Radiowellen (*Radio Frequency Identification*, RFID) [Rou08] sowie die Entwicklungen im Bereich der mobilen Datenverarbeitung und Kommunikation [Rot05].

Eine wesentliche Implikation aus diesem Entwicklungstrend ist der Umstand, dass die Menge an Daten, die erhoben, verarbeitet und kommuniziert werden, in Zukunft weiter drastisch ansteigen wird. Auch ist anzunehmen, dass die Daten verschiedenster Quellen miteinander verknüpft und in Beziehung gesetzt werden, um eine umfassende Sicht auf die Vorgänge in der physischen, logischen und sozialen Umwelt zu erhalten. Anwendungen, die derlei Daten verwenden, ordnet man der Klasse der *kontextbasierten Anwendungen* zu [CK00]. Diese Klasse zeichnet sich dadurch aus, dass Kontextdaten die Grundlage für eine adäquate Präsentation von Informationen oder Verhaltensadaptionen der Anwendung selbst darstellen. Dies erfordert eine detaillierte und vor allem auf bestimmte Aspekte der Umwelt fokussierte Sicht.

Um eine solche *kontextbasierte Sicht*, d.h. eine Sicht, die alle für einen bestimmten Anwendungszweck relevanten Informationen in einer geeigneten Repräsentation enthält, zu erlangen, bedarf es der Vermittlung von Kontextdaten zwischen Produzenten (z.B. Sensoren) und Konsumenten (z.B. Menschen oder Anwendungen). Neben der Datenerhebung und -übertragung gehören hierzu vor allem auch die Datenverarbeitung, -verwaltung und -abfrage. Insbesondere im Rahmen mobiler, ubiquitärer Systeme birgt dies eine ganze Reihe von Herausforderungen. Diese sollen im folgenden Abschnitt aufgezeigt und in Bezug zu dieser Arbeit gesetzt werden. Im Anschluss daran folgen die konkreten Forschungsfragen sowie die Zielsetzung dieser Dissertation.

1.1. Herausforderungen

Um Menschen und Dingen eine adäquate Sicht auf die Vorgänge der Umwelt zu ermöglichen, bedarf es der Vermittlung geeigneter Kontextrepräsentationen. Wie bei der „Vermittlung von Wissen“ oder der „Vermittlung von Eindrücken“ kommt es dabei darauf an, die Kontextinformation für den Konsumenten in verständlicher Art und Weise zu überliefern. Kontext kann hierbei nach [DA99] als jedwede Information zur Beschreibung der Situation einer Entität verstanden werden. Dies können physische Attribute (z.B. Temperatur), aber auch Daten virtueller Sensoren (z.B. Log-Datei eines Servers) sein. Da Kontext ein sehr allgemeines Konzept darstellt und die einen Kontext konstituierenden Informationen sehr heterogen sind, bedarf es Standards, gemeinsamer semantischer Übereinkünfte sowie geeigneter Datenmodelle, damit Produzenten und Konsumenten sinnvoll miteinander interagieren können.

In den bis heute realisierten Anwendungen sind Produzenten und Konsumenten meist eng gekoppelt und es finden sich 1:1-, 1:n- oder n:1-Beziehungen [PHPS10]. In solchen Konstellationen ist die Programmlogik zur Verarbeitung von Kontextdaten häufig auf einen bestimmten Anwendungsfall zugeschnitten und ein Wechsel des Anwendungsfalls zieht eine Umprogrammierung der Geräte nach sich. Jedoch erleichtern solche Beziehungen die Interaktion, da sich eine Seite an die Vorgaben der anderen bereits zur Entwurfszeit anpassen kann. Wünschenswert wäre allerdings eine n:m-Beziehung, bei der Produzenten ihre Daten beliebigen Konsumenten zur Verfügung stellen und Konsumenten auch zur Laufzeit dynamisch geeignete Produzenten auswählen können. Unter diesen Voraussetzungen müssen entweder global einheitliche Standards etabliert oder die Interaktion über einen Vermittler geführt werden, der zwischen den heterogenen Technologien, Protokollen, Datenmodellen, Anforderungen etc. vermitteln kann.

Es gibt bereits eine Vielzahl von Standards, welche verschiedene Aspekte der Interaktion abdecken. Allerdings wird es aufgrund der herrschenden Heterogenität nicht möglich sein, einen sinnvollen einheitlichen Standard für die Vermittlung von Kontextinformationen zwischen beliebigen Dingen untereinander sowie zwischen Dingen und Menschen zu etablieren, da die einzugehenden Kompromisse zu groß wären. Ein Grund hierfür liegt u.a. in der Ressourcenausstattung der idealerweise zu berücksichtigenden Geräte, die von einem simplen Sensor, der mittels eines proprietären Protokolls kommuniziert, bis hin zu leistungsstarken Computern mit einem kompletten Web Service Stack, reichen. Unter Berücksichtigung mobiler Teilnehmer verschärfen sich zudem die Anforderungen, da mit Änderung des physischen Aufenthaltsorts häufig auch Änderungen in der logischen Umwelt (z.B. Wechsel der Endpunktadresse) einhergehen. Einerseits sind diese dynamischen Kontextwechsel besonders relevant, da anwendungsseitig die Notwendigkeit zur Anpassung an den jeweiligen Kontext steigt. Andererseits müssen jedoch die Kontextinformationen ggf. über inhärent störanfällige Kommunikationskanäle mit geringen Datenraten und hohen Latenzen vermittelt werden. Kurzum, ein einheitlicher Standard

für die Vermittlung von Kontextinformationen in solch heterogenem Umfeld ist nicht sinnvoll zu realisieren.

Ähnliche Ansätze wurden bereits von einer Reihe von Forschungsprojekten mit verwandten Zielen gewählt. Jedoch liegt bei existierenden Forschungsarbeiten der Fokus auf der reinen Datenvermittlung, die ebenfalls besondere Herausforderungen stellt. Hier sind insbesondere die massive Menge an Daten, die von Sensoren erhoben werden und die Teil eines zu vermittelnden Kontextes sein können, zu berücksichtigen. Optimistischen Prognosen zufolge sollen bis zum Jahr 2020 ca. 200 Milliarden Teilnehmer im Internet sein [IDC12]. Die Menge an Informationen, die dabei lediglich durch Sensoren bzw. intelligente Dinge produziert und kommuniziert wird, soll einen wesentlichen Teil des Gesamtverkehrsaufkommens von geschätzten 40 Zetabytes (40 Trillionen Gigabyte) ausmachen [IDC12]. Die existierenden Arbeiten konzentrieren sich insbesondere auf diese Herausforderung und versuchen durch Ausnutzen besonderer Eigenschaften des Kontextes sowie verteilten Systemarchitekturen die Kontextdaten von den Produzenten zu den Konsumenten zu übertragen. Dabei müssen sich oft sowohl Produzenten als auch Konsumenten an die neu eingeführten Abstraktionen anpassen. Eine Vermittlung kontextbasierter Sichten, die keine Anpassung seitens der Konsumenten verlangen würde, wird nicht adressiert.

Zur Erstellung kontextbasierter Sichten ist es erforderlich, die rohen Kontextdaten auf dem Weg vom Produzenten zum Konsumenten aufzubereiten, d.h. zum Beispiel in geeignete Datenmodelle zu überführen, Einheiten zu konvertieren, Merkmale zu extrahieren, weitere Informationen hinzuzufügen etc. Aufgrund der unterschiedlichen Anforderungen der Konsumenten an diese Sichten, gibt es auch hier keine generische Lösung. Stattdessen muss die Aufbereitung individuell nach Maßgabe der Konsumenten erfolgen. Diese müssen einem Vermittler ihre Anforderungen in geeigneter Weise verständlich machen können, sodass dieser eine entsprechende Verarbeitung vornehmen kann. Je nach Anwendungsfall kann dies auch komplexe Verarbeitungsschritte erfordern, die vieler Ressourcen bedürfen und dennoch zeitnah ein Resultat liefern sollen. Daraus wiederum ergeben sich besondere Anforderungen hinsichtlich der Skalierbarkeit, Robustheit und Flexibilität eines Vermittlers, um sowohl zwischen räumlich benachbarten Knoten, ggf. sogar rein lokalen Anwendungen, als auch auf globaler Ebene vermitteln zu können. Und schließlich kann es auch notwendig sein, dass ein Konsument steuernd auf einen Produzenten einwirken muss, um bereits die Datenerhebung an die eigenen Erfordernisse anzupassen, wodurch eine bidirektionale Interaktion über einen Vermittler möglich sein muss.

Hinsichtlich der oben erwähnten Menge an Daten ist zudem zu überlegen, ob und wie diese persistiert und ggf. abgefragt werden können. Etablierte relationale Datenbanksysteme, wie sie heutzutage vielfach eingesetzt werden, eignen sich nur bedingt für ein solch massives Volumen heterogener Daten [JC09]. Es gilt, andere Konzepte zur Verwaltung von Daten für den gegebenen Einsatzkontext zu untersuchen. In diesem Zusammenhang steht auch die Datenab-

frage seitens der Konsumenten. Man unterscheidet hierbei drei wesentliche Arten von Abfragen: i) Ad-hoc Abfragen beziehen sich auf die aktuelle Situation einer Entität, ii) historische Abfragen berücksichtigen auch vergangene Zeitreihen und iii) ereignisbasierte Abfragen, die erst in der Zukunft, bei Eintreten eines bestimmten Ereignisses, ein Ergebnis liefern. Bei allen Arten von Abfragen muss es zusätzlich die Möglichkeit geben, dass Konsumenten Abfrageergebnisse abonnieren können, um regelmäßige Aktualisierungen zu erhalten (Publish/Subscribe-Prinzip). Außerdem können sich alle Arten von Abfragen auch auf einen komplexen oder aggregierten Kontext beziehen, welcher Daten verschiedener Produzenten fusioniert und dabei ggf. auch kausale und temporale Beziehungen zwischen den Daten berücksichtigen kann.

1.2. Forschungsfragen

Aus den vorgestellten Herausforderungen, lässt sich nun folgende Forschungsfrage, die im Rahmen dieser Arbeit beantwortet werden soll, ableiten:

- Wie lassen sich individuell aufbereitete kontextbasierte Sichten zwischen heterogenen Konsumenten und Produzenten in mobilen, ubiquitären Netzen vermitteln ?

Die Forschungsfrage hat einen hohen gestalterischen Schwerpunkt, der sich in einzelne Teilaspekte untergliedern lässt. Zu jedem dieser Aspekte lassen sich weitere Teilfragen konkretisieren:

- Wodurch ist eine kontextbasierte Sicht charakterisiert ?
- Welche funktionalen und nicht-funktionalen Anforderungen werden an eine Vermittlung seitens der Produzenten und Konsumenten gestellt ?
- Für welche Aufgaben müssen im Rahmen der Vermittlung, insbesondere unter Berücksichtigung mobiler und ressourcenschwacher Geräte, Unterstützungspunkte geboten werden ?
- Für welche Unterstützungspunkte existieren bereits adäquate Lösungen und wie lassen sich diese in den Prozess der Vermittlung integrieren ?
- Wie lassen sich offene Unterstützungspunkte geeignet konzipieren und realisieren ?
- Wie lassen sich geeignete Schnittstellen definieren, die Entwicklern und Benutzern einen einfachen Umgang mit kontextbasierten Sichten ermöglichen ?

1.3. Zielsetzung

Die Ausstattung von Dingen mit Sensoren zur Wahrnehmung ihrer Umwelt und der Fähigkeit Wahrnehmungen zu verarbeiten und zu kommunizieren,

eröffnet fundamental neue Möglichkeiten im öffentlichen, wirtschaftlichen und privaten Bereich. Nicht nur Menschen können sich somit jederzeit ein genaues Bild bestimmter Aspekte der Umwelt machen, auch Dinge selbst können sich untereinander über ihren aktuellen Kontext austauschen und sich der Umwelt bzw. den Bedürfnissen ihrer Nutzer anpassen.

Hierfür ist es erforderlich, die Wahrnehmung über Sensoren zunächst zu interpretieren, zu analysieren und geeignet zu repräsentieren, um möglichen Interessenten dann eine adäquat aufbereitete Sicht auf den aktuellen Kontext zu vermitteln. Aufgrund der großen Heterogenität seitens der Beteiligten, der Kontextinformationen, der technischen Voraussetzungen und letztlich auch der Anforderungen der Anwendungen ist die Vermittlung einer solchen kontextbasierten Sicht nicht einheitlich möglich. Es bedarf eines flexibleren Ansatzes, der alle Bedürfnisse adäquat umsetzen kann.

Ziel dieser Arbeit ist es daher, aufbereitete Kontextinformationen zwischen Produzenten und Konsumenten zu vermitteln und diese dabei dahingehend zu entlasten, dass sie sich nicht existierenden Standards, etwaigen Anforderungen an die Ressourcenausstattung oder technischen Erfordernissen anpassen müssen. Dies gibt ihnen die Möglichkeit, die (wenigen) vorhandenen Ressourcen wie Rechenleistung, Speicherplatz und Energie für ihren eigentlichen Anwendungszweck zu nutzen.

Bezogen auf die Entwicklung kontextbasierter Anwendungen im Allgemeinen soll das Ergebnis dieser Arbeit die Entwickler dahingehend unterstützen, dass ein Großteil der Verarbeitungslogik zur Aufbereitung der Kontextinformation durch einen Vermittler vorgehalten wird, wodurch Wiederverwendung von Programmcode unterstützt, Mash-up-ähnliche Anwendungen ermöglicht, die Zeit zur Umsetzung kontextbasierter Anwendungen verkürzt und die Anwendungen selbst einfacher auch zur Laufzeit an neue Anforderungen angepasst werden können.

Benutzer, die sich über bestimmte Aspekte einer Umwelt informieren oder benachrichtigt werden möchten, sobald bestimmte Ereignisse eintreten, sollen einfache Benutzungsschnittstellen erhalten, die es ihnen gestatten, Kontextdatenquellen aufzufinden, einheitlich abzufragen, die Ergebnisse mit anderen Informationsquellen zu verknüpfen und schließlich in eine adäquate und für sie verständliche Form überführen zu lassen.

Die wesentlichen konzeptionellen Ziele dieser Arbeit lassen sich folgendermaßen zusammenfassen:

- Identifikation von Anforderungen an die Vermittlung von kontextbasierten Sichten in mobilen, ubiquitären System, welche es Produzenten erlaubt, Kontextdaten mit beliebigen Konsumenten zu teilen und Letztere unterstützt, kontextbasierte Sichten von beliebigen Produzenten zu beziehen. Hierzu gehört auch eine Untersuchung existierender Arbeiten und deren Bewertung hinsichtlich der identifizierten Anforderungen.
- Entwurf eines Prozesses, welcher alle Aufgaben im Rahmen der Vermittlung von allgemeinen Kontextinformationen unter Berücksichtigung der

speziellen Anforderungen mobiler, ubiquitärer Systeme abdeckt, sowie die Definition geeigneter Schnittstellen zur Initiierung des Prozesses seitens der Entwickler und Benutzer.

Ergänzend zu dem konzeptionellen Teil soll eine Middleware entworfen werden, welche als Vermittler zwischen Produzenten und Konsumenten agiert. Hierbei sollen Entwurfsentscheidungen, welche den nicht-funktionalen Anforderungen mobiler, ubiquitärer Systeme Rechnung tragen, getroffen und dabei existierende Standards des Sensor Webs [ZSM09] und des Internet der Dinge [MF10] berücksichtigt werden. Eine exemplarische, prototypische Realisierung eines Vermittlers soll die prinzipielle Umsetzbarkeit des Konzepts beweisen.

1.4. Ergebnisse und Beiträge

Der zentrale Forschungsaspekt dieser Dissertation beschäftigt sich mit der Untersuchung und Erarbeitung von Konzepten zur Vermittlung kontextbasierter Sichten zwischen Konsumenten und Produzenten in mobilen, ubiquitären Systemen, welche den Entwurf und die Entwicklung kontextbasierter Anwendungen unterstützen sollen.

Als Ergebnis wurden wesentliche (nicht-)funktionale Anforderungen an einen Vermittlungsprozesses erhoben, im Rahmen dessen Kontextdaten nach Maßgabe der Produzenten und Konsumenten in eine geeignete Repräsentation überführt werden können, sodass die möglicherweise ressourcenarmen und ggf. mobilen Teilnehmer von der Verarbeitungslast befreit werden. Unter Berücksichtigung der Anforderungen wurden wesentliche Vermittlungsaufgaben identifiziert, existierende Lösungsmöglichkeiten für Teilprobleme aufgezeigt sowie neue Lösungen für offene Probleme konzipiert und schließlich die Architektur eines Vermittlers entworfen, welcher alle notwendigen Prozessschritte sowie dazugehörige Verwaltungsaufgaben umsetzt. Um den zuvor genannten Herausforderungen Rechnung zu tragen, kann dieser Vermittler in einem verteilten System durch eine Vielzahl autonomer, aktiver Komponenten realisiert werden, sodass größtmögliche Flexibilität, Erweiterbarkeit, Skalierbarkeit, Robustheit etc. geboten sind.

Durch Berücksichtigung etablierter Standards der Sensor Web Enablement-Initiative des Open Geospatial Consortiums sowie der Auto-ID-Standards von EPCglobal bzw. Global Standards One ist ein Vermittler zudem in der Lage, mit existierenden Systemen aus den Bereichen des Sensor Webs sowie des Internet der Dinge zu interoperieren und höherwertige Aufgaben, z.B. das Erstellen kontextbasierter Sichten für die (mobilen) Klienten, zu übernehmen. Somit können Kontextdaten durch Konsumenten leichter aufgefunden und Daten verschiedenster Quellen miteinander verknüpft werden, die Verarbeitungslast kann in die Infrastruktur verschoben und ressourcenschwache Geräte am Rand des Systems können somit entlastet werden.

1.5. Aufbau der Arbeit

Das folgende Kapitel 2 führt zunächst in die Grundlagen der Kontextdatenverarbeitung ein. Angefangen mit grundsätzlichen Möglichkeiten zur Erhebung von Kontextdaten mittels Sensoren und Sensornetzen werden Definitionen und Klassifikationen von Kontext vorgestellt, bevor typische Funktionen der Kontextdatenverarbeitung erläutert werden. Kapitel 3 widmet sich dem zweiten Grundlagenbereich: den mobilen und ubiquitären Systemen. Hier werden die besonderen Charakteristika derartiger Systeme hervorgehoben, die eine Vermittlung kontextbasierter Sichten motivieren.

In Kapitel 4 folgt dann die Konzeption eines Vermittlungsprozesses, wobei zunächst die unterschiedlichen Rollen der Beteiligten detailliert werden, um dann die einzelnen Aufgabenbereiche zu präsentieren. Dies legt die konzeptionellen Grundlagen für das 5. Kapitel, welches den Begriff einer kontextbasierten Sicht definiert, mittels einer Reihe von Anwendungsbeispielen deren Nutzen aufzeigt und auf dieser Basis anhand einer Analyse funktionaler und nicht-funktionaler Anforderungen den Vermittlungsprozess für kontextbasierte Sichten konkretisiert. Anschließend werden für alle Aufgabenbereiche die sich ergebenden Teilprobleme sowie etwaige Lösungsmöglichkeiten aufgezeigt.

Im 6. Kapitel wird eine systematische Untersuchung verwandter Arbeiten, die sich im weitesten Sinn mit der Kontextdatenvermittlung im Sensor Web bzw. im Internet der Dinge beschäftigen, durchgeführt und alle Arbeiten anhand des zuvor aufgestellten Anforderungskatalogs bewertet. Im Anschluss wird in Kapitel 7 eine Diskussion über geeignete Entwicklungsparadigmen zur Realisierung eines Vermittlers geführt und schließlich der softwaretechnische Entwurf sowie eine exemplarische, prototypische Implementierung vorgestellt. Eine qualitative und quantitative Evaluation dieses Ansatzes bzw. dessen Umsetzung erfolgt schließlich im 8. Kapitel, bevor Kapitel 9 die Ergebnisse diskutiert, die Arbeit zusammenfasst und einen Ausblick auf Anknüpfungspunkte und Erweiterungsmöglichkeiten gibt.

2. Grundlagen der Kontextdatenverarbeitung

Kontext kann als eine Situationsbeschreibung aufgefasst werden, welche Informationen über den Zustand einer Entität oder Umgebung zu einem bestimmten Zeitpunkt repräsentiert. Die Informationen werden meist durch physische oder virtuelle Sensoren erhoben und in einer geeigneten Repräsentation zur weiteren Nutzung bereitgestellt. Da Kontext häufig eine Aggregation unterschiedlicher Daten, von verschiedenen, auch räumlich getrennten Sensoren darstellt, müssen diese Sensordaten zunächst zusammengeführt, interpretiert und schließlich in eine geeignete Repräsentation überführt werden, damit sie von Konsumenten genutzt werden können. In der Praxis existieren hierfür eine Reihe typischer Verarbeitungsfunktionen, welche die „rohen" Sensordaten für die Konsumenten aufbereiten und als Kontextinformationen bzw. -wissen zur Verfügung stellen können.

Um die Grundlagen für das Verständnis dieses Prozesses zu legen, werden im folgenden Abschnitt 2.1 zunächst Sensoren, Sensornetze und das Sensor Web vorgestellt. Abschnitt 2.2 soll im Anschluss den Kontextbegriff erläutern, verschiedene Klassifikationssysteme für Kontextinformationen vorstellen und den Nutzen von Kontext aufzeigen. Schließlich wird in Abschnitt 2.3 ein Überblick über gängige Verarbeitungsfunktionen zur Erstellung geeigneter Repräsentationen gegeben.

2.1. Sensoren, Sensornetze und das Sensor Web

Sensoren sind meist auf die Wahrnehmung eines oder weniger Phänomene bzw. Umweltattribute spezialisiert und haben ein räumlich stark begrenztes Wahrnehmungsfeld. Um den Skopus der Wahrnehmung zu vergrößern, können sie sich daher zu sogenannten Sensornetzen zusammenschließen. Diese decken meist eine größere Region ab, werden von einer organisatorischen Einheit betrieben und dienen häufig einem dedizierten Anwendungszweck. Um die Daten auch global verfügbar zu machen und einen inter-organisationalen und anwendungsübergreifenden Zugriff zu ermöglichen, können die Daten schließlich im Sensor Web veröffentlicht werden. Diese drei verschiedenen Skalen werden im Folgenden beschrieben.

2.1.1. Sensoren

Üblicherweise versteht man unter einem Sensor ein kleines Bauelement, welches ein Phänomen bzw. Attribut der physischen Welt misst, ggf. verarbeitet und kommuniziert. Beispiele für Sensoren im Alltag sind Bewegungs- und Ortsbestimmungssensoren, Geschwindigkeits- und Beschleunigungssensoren, Belastungs- und Berührungssensoren, Druck-, Temperatur- und Feuch-

tigkeitssensoren, Sensoren zur Detektion von Gasen, bestimmten Chemikalien oder Strahlungen sowie akustische und optische Sensoren [Fra10]. Da es neben diesen physischen Sensoren noch eine beliebige Vielfalt sogenannter virtueller Sensoren gibt, soll zunächst der Begriff des Sensors konkretisiert werden.

> *"[A sensor is an] entity that provides information about an observed property at its output. A sensor uses a combination of physical, chemical or biological means in order to estimate the underlying observed property." [Ope08b]*

Laut dieser Definition liefert ein Sensor Informationen über eine bestimmte wahrgenommene Eigenschaft, wobei diese ein physikalisches, biologisches oder chemisches Phänomen der Umwelt repräsentiert [Ope08b]. Der Sensor verwendet entsprechend physikalische, chemische oder biologische Mittel, um eine Schätzung des Wertes dieser Eigenschaft vorzunehmen. Hervorzuheben ist hier, dass die Ausgabe eines Sensors also nicht unbedingt einer Tatsache entsprechen muss, sondern durchaus ungenau und fehlerbehaftet sein kann. Da diese Definition jedoch durch Einschränkung auf physikalische, chemische oder biologische Mittel eine wesentliche Art von Sensoren ausschließt, z.B. Anwendungen, die über eine Schnittstelle Sensordaten zum Abruf bereitstellen und somit selbst als abstrakter Sensor fungieren, soll eine weitere Definition gegeben werden.

> *"A sensor is a component that measures the state of the environment." [SWSVR05]*

Die Autoren Sgroi et al. erläutern, dass zu diesem Verständnis eines Sensors neben den physischen Sensoren explizit auch sogenannte *virtuelle Sensoren* zählen [SWSVR05]. Virtuelle Sensoren seien beispielsweise oben erwähnte Anwendungen, die stellvertretend für einen physischen Sensor Daten über den Zustand der physischen Welt weitergeben. Auch wenn in dieser Definition bereits von virtuellen Sensoren gesprochen wird, so nehmen auch diese Autoren eine Einschränkung auf die physische Umwelt vor. Attribute der logischen Umwelt, z.B. die Auslastung eines Prozessors, die Endpunktadresse eines Gerätes oder die Belegung einer Variablen, können mit diesen Sensordefinitionen noch nicht erfasst werden.

> *"A sensor is a device that receives a stimulus and responds with an electrical signal" [Fra10]*

Diese Definition verallgemeinert noch ein wenig mehr, da sie auf dem generischen Begriff des Stimulus beruht, welcher für eine Menge, eine Eigenschaft oder einen Zustand steht [Fra10]. Es wird keine weitere Aussage über die Art des Stimulus getroffen, wohl aber über die Ausgabe des Sensors: ein elektrisches Signal. Im Rahmen dieser Arbeit ist diese Einschränkung jedoch sinnvoll, da Sensoren als Teil informationsverarbeitender Systeme angesehen werden und andere Ausgabeformate, z.B. elektrochemische, außer Acht gelassen werden können. Übersetzt man den englischen Begriff „device" nicht

mit „Gerät“, sondern mit „Instrument“ oder „Hilfsmittel“, so fallen auch oben erwähnte virtuelle Sensoren unter diese Definition.

Im Rahmen dieser Arbeit sollen virtuelle Sensoren jedoch, anders als in [SWSVR05], nicht lediglich als Stellvertreter für physische Sensoren betrachtet werden. Auch Software, die Attribute ihrer (logischen) Umwelt wahrnehmen kann, soll als virtueller Sensoren verstanden werden können. Somit ist es beispielsweise auch möglich anwendungs- oder systeminterne Attribute, Leistungskennzahlen oder Benutzerinteraktionen als Stimuli aufzufassen.

Von den einfachen physischen und virtuellen Sensoren als Primärquellen für Sensordaten lassen sich noch die *komplexen Sensoren* unterscheiden [Ope08b]. Diese Art von Sensoren erhält Sensordaten von Primärquellen als Eingabe, kann diese Daten weiterverarbeiten und als höherwertige Sensordaten selbst wieder zur Verfügung stellen[1]. Beispiele hierfür wären z.B. ein Unwettersensor, welcher auf Basis unterschiedlichster Messwerte die Wahrscheinlichkeit eines Unwetters emittiert oder ein Aktivitätssensor, welcher die momentane Aktivität eines Nutzers unter Verwendung verschiedenster Primärsensoren ableitet.

2.1.2. Sensornetze

Ein einzelner Sensor hat meist ein lokales, sehr eingeschränktes Wahrnehmungsfeld. Um größere Gebiete der physischen Umwelt abzudecken, werden in der Praxis meist eine Vielzahl an Sensoren in der Umwelt ausgebracht. Jeder dieser Sensoren führt in bestimmten Zeitintervallen Messungen durch und kommuniziert diese an ein oder mehrere seiner Nachbarn in Funkreichweite. Auf diese Weise werden die Sensordaten an den Rand des Netzes zu einer Basisstation übermittelt und können dort weiterverarbeitet werden [AV10]. Ein solches Netzwerk von Sensoren wird als *(Wireless) Sensor Network* bezeichnet.

> *“A sensor network is an infrastructure comprised of sensing (measuring), computing, and communication elements that gives an administrator the ability to instrument, observe, and react to events and phenomena in a specified environment. [...] The environment can be the physical world, a biological system, or an information technology framework.“ [SMZ07]*

Es handelt sich also um ein Netzwerk von Sensoren, welche die Fähigkeit besitzen, bestimmte Stimuli der physischen oder logischen Umwelt[2] wahrzuneh-

[1] In [AHS06b] werden komplexe Sensoren auch als virtuelle Sensoren verstanden. Da sie oft durch Software realisiert sind, entspricht das auch dem Verständnis in dieser Arbeit. Allerdings lassen sich komplexe Sensoren weitergehend abgrenzen, als dass diese auch als *Sensorsystem*, also eine Aggregation physischer Sensoren, die sich auf einer Plattform befinden (z.B. eine Wetterstation oder ein Satellit) [BEJ+11], verstanden werden können.

[2] An Sensornetze, welche ausschließlich aus virtuellen Sensoren bestehen, werden geringere Anforderungen gestellt, da sie meist innerhalb von infrastrukturbasierten Systemen ausgeführt werden. Die in diesem Abschnitt vorgestellten Sensornetzwerke, bestehend aus physischen Sensoren, unterliegen aufgrund der geringen Ressourcenausstattung wesentlich höheren Restriktionen. Der folgende Teil dieses Unterabschnitts bezieht sich daher nur auf solch Sensornetze.

men, ggf. lokal zu verarbeiten, und das Ergebnis dann in Form einer Nachricht über einen (meist kabellosen) Kommunikationskanal an benachbarte Knoten weiterzuleiten. Das Ziel einer solchen Nachricht ist ein spezieller Knoten im Sensornetz, der als Basisstation (engl. *Sink*) bezeichnet wird. Ein solcher Knoten dient als eine Art Gateway zwischen dem Sensornetz und dem Benutzer bzw. Administrator, welcher zum Beispiel über das Internet Sensordaten abfragen oder Anweisungen (bspw. Konfigurations- oder Aktuatorbefehle) an das Netzwerk übermitteln kann [AV10].

Auf der rechten Seite der Abbildung 2.1 ist die grundlegende Architektur eines Sensornetzes mit einer Reihe von Sensorknoten sowie der Basisstation dargestellt. Die Kommunikation von einem Sensor hin zur Basisstation verläuft meist über mehrere Stationen mittels Multi-Hop-Übertragungsprotokollen, da die Funkreichweite der gängigen Sensoren weniger als 100 m beträgt und der Energieaufwand mit zunehmender Distanz überproportional steigt. Die Basisstation hingegen ist in den meisten Fällen an das Internet angebunden und kann somit die erhaltenen Sensordaten veröffentlichen oder Anweisungen empfangen [AV10].

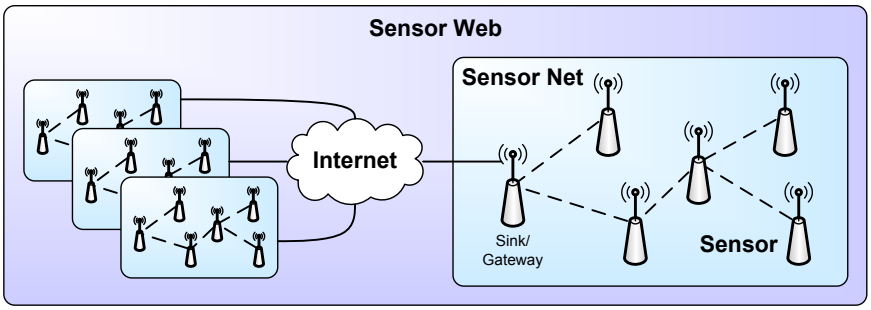


Abbildung 2.1.: Typische Architektur eines Sensornetzes bzw. Sensor Webs

Da Sensornetzwerke typischerweise über einen längeren Zeitraum ohne etwaige Wartung funktionieren sollen, die Sensoren jedoch meist nur einen begrenzten Energievorrat haben, konzentriert sich die Forschung im Bereich der Sensornetzwerke insbesondere auf die Optimierung des Energieverbrauchs, das Vorverarbeiten von Daten auf den Sensorknoten sowie das Nachrichtenrouting innerhalb des Sensornetzes [BEJ+11, AHS06b, SPL+04]. Im Rahmen dieser Arbeit sollen die Vorgänge innerhalb eines Sensornetzes nicht weiter betrachtet werden. Stattdessen werden Sensornetze als eine Art Blackbox angesehen, welche Anweisungen als Eingabe entgegennimmt und Sensordaten als Ausgabe produziert.

Typische Anwendungsbereiche für Sensornetze finden sich in vielerlei Domänen und fallen meist in eine der folgenden Kategorien: Umwelt, Gesundheit, Gebäude, Industrie und Militär [AV10]. Konkrete Anwendungsbeispiele sind die Gebäudekontrolle und Heimautomatisierung, Umweltüberwachung (Wetter, Erdbeben, Lawinen, Tsunamis, Tiere) und Landwirtschaft, Verkehrsflusssteuerung, Patientenüberwachung, Prozesskontrolle, Werkzeugwartung, Inventarisierung etc. [SMZ07].

2.1.3. Sensor Web

Während Sensoren ein lokales Wahrnehmungsfeld besitzen, erlauben Sensornetze, als Zusammenschluss vieler räumlich verteilter Sensoren, die Überwachung größerer Regionen. Sie sind jedoch heutzutage meistens spezifisch auf einen Anwendungsfall zugeschnitten und unterliegen der Kontrolle einer Organisation, obwohl unter Umständen auch darüber hinausgehendes Interesse an den Daten bestünde [ZSM09]. An diesem Punkt setzt das sogenannte *Sensor Web* an. Es erweitert den Skopus des Auffindens, Austauschs und Verarbeitens von Sensordaten, indem es auf globaler Ebene anwendungsübergreifend eine Vielzahl an Sensoren, Sensornetzen und anderen Sensordatenquellen (z.B. Simulatoren oder Datenbanken) miteinander verknüpft (siehe Abbildung 2.1) [FMF09, Nit09, ZSM09].

Konzeptionell kann es als eine kollaborative, zusammenhängende, konsistente und vereinigte Metaplattform zur Sammlung, Fusion und Distribution beliebiger Sensordaten angesehen werden [Ope08b]. In Analogie zum World Wide Web (WWW) stellt es jedoch anstatt allgemeiner Informationen insbesondere Informationen über die physische Umwelt zum Abrufen und zur weitergehenden Verarbeitung zur Verfügung [BFJP10].

Vom technischen Standpunkt aus betrachtet, basiert es auf etablierten Konzepten, Standards und Protokollen des WWW und abstrahiert somit von den technischen Details und der Heterogenität der Sensornetze. Allerdings wird der Begriff des Sensor Webs durchaus verschiedenartig gebraucht [ZSM09]. So existieren bereits verschiedene Realisierungen, welche jede für sich beansprucht, ein Sensor Web zu sein (vgl. Kapitel 6). Zyl et al. argumentieren jedoch, dass es lediglich ein einziges Sensor Web geben kann, in welchem andere Sensor Web-Implementierungen als konstituierende Komponenten aufgehen, so wie das WWW, Peer-2-Peer-Netzwerke, Multiagentensysteme, Grid-Technologien etc. unterschiedliche Ansätze bzw. Paradigmen darstellen, die Teil des Internets sind [ZSM09].

Die *Sensor Web Enablement Initiative* [Ope12b] des Open Geospatial Consortiums hat durch eine Reihe von Standards inzwischen eine prominente Rolle und somit den stärksten Einfluss sowohl auf die Begriffsbildung als auch auf die technischen Grundlagen [ZSM09]. Die Standards des Sensor Web Enablement (SWE) umfassen Datenmodelle zur Beschreibung von Sensoren und Sensormessungen und Schnittstellenmodelle für eine Reihe notwendiger Dienste zum Auffinden von Sensoren und Daten, zum Abfragen von Daten, zum Benachrichtigen von Klienten, sobald bestimmte Daten verfügbar sind, und zum Übermitteln von Anweisungen an Sensoren [Ope08b] (vgl. auch Abschnitte 5.4.9 und 7.1.4.1). Durch die breite Akzeptanz dieser Standards hat sich somit auch eine technische Basis für das Sensor Web etabliert, welche durch offene Implementierungen (z.B. durch *52° North* [52N12b] oder *Constellation SDI* [Con12]) einfach in bestehende Systeme integriert werden kann.

Momentan sehen die Standards jedoch statt einer globalen Verknüpfung von Sensornetzen lediglich unabhängige, verteilte Dienste mit jeweils eigenen Da-

tenbasen vor und sind somit dem konzeptionellen, übergeordneten Ziel eines globalen Netzes noch fern [Ope08b]. Mit zunehmender Verbreitung von Sensornetzen und deren Einzug in immer neue Anwendungsdomänen werden weitere Anforderungen an die Standards hinzukommen [BL09]. Auch bleibt abzuwarten, inwiefern das heutige Internet den Anforderungen, insbesondere hinsichtlich der Skalierbarkeit, gerecht wird, denn das Sensor Web wird auch als ein sogenanntes *high fan-in*-System betrachtet [FJK+05]. Im Kern finden sich die traditionellen, leistungsstarken Server, die über breitbandige Verbindungen mit der Peripherie des Netzes, repräsentiert durch Sensoren und Sensornetze, verbunden sind. Somit wird das Datenaufkommen nicht über das ganze Netz verteilt, sondern von außen in den Kern der Infrastruktur geleitet, welche somit eine wesentlich höhere Last verarbeiten muss.

2.2. Kontext und Kontextbewusstsein

Sensordaten repräsentieren ganz allgemein Informationen über die physische oder logische Umwelt. Werden sie in Bezug zu einer beliebigen Entität gesetzt, bezeichnet man diese Art von Daten auch als *Kontext* dieser Entität [Win01]. Das Wissen um den Kontext erlaubt es Entitäten einerseits die Interaktion mit anderen Entitäten untereinander abzustimmen, andererseits erlaubt es einer Entität auch ihr eigenes Verhalten an den eigenen Kontext zu adaptieren. Um ein Verständnis von Kontext zu erlangen, soll im Folgenden daher der Kontextbegriff vorgestellt und eine Kategorisierung vorgenommen werden. Im Anschluss daran soll die Eigenschaft des Kontextbewusstseins und der sich daraus ergebende Nutzen verdeutlicht werden.

2.2.1. Kontextdefinition

Der Kontextbegriff wird im Alltag sowie in der Literatur häufig unscharf verwendet und ist mit unterschiedlichen Bedeutungen überladen. Insbesondere auch in den Bereichen der Datenverarbeitung und der Mensch-Maschine-Interaktion wird der Begriff verschiedenartig gebraucht. Im Folgenden soll daher anhand einer Reihe von Definitionen das Begriffsbild von Kontext, welches in dieser Arbeit verwendet wird, konkretisiert werden.

Viele Autoren definieren Kontext durch Angabe von Synonymen (z.B. Umgebung, Situation) oder Beispielen (z.B. Ort, Temperatur, Aktivität). Solche indirekten Definitionen sind in ersterem Fall zu allgemeingültig und in letzterem Fall unvollständig [ZLO07, Hen03]. Einen Mittelweg versuchen Dey und Abowd mit einer der am häufigsten zitierten Definition von Kontext im Bereich der Informationstechnologie:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." [DA99]*

Diese Definition macht keine Einschränkungen bezüglich der Art und Menge an Informationen, die als Kontext einer Entität betrachtet werden können, jedoch schränkt sie die Art der Entitäten, auf die Kontext bezogen werden kann, einerseits auf Personen, Plätze und Objekte, andererseits aufgrund deren Relevanz für die Interaktion zwischen einem Nutzer und einer Anwendung ein. Kontextinformationen sind jedoch allgemeiner zu fassen, da ihr Nutzen nicht zwangsläufig einen „Anwendungsbezug“ (im Sinne der Datenverarbeitung) haben muss. Der Umstand beispielsweise, dass ein Wald brennt, muss nicht zwangsläufig relevant für die Interaktion mit einer Anwendung sein, trotzdem würde man die Informationen in Bezug zur Entität Wald als Kontext verstehen und nutzen wollen.

Eine weitere Definition liefert Chen in [Che04]. Auch wenn diese den Kontext durch Aufzählung definiert, schränkt sie die kontextbezogenen Entitäten nur unwesentlich und auf ein praktisches Maß ein. Der Kontextbegriff, so wie er im Rahmen dieser Arbeit verstanden wird, lehnt sich eng an diese Definition von Chen an:

> *Context is information about a location, its environmental attributes (e.g., noise level, light intensity, temperature, and motion) and the people, devices, objects and software agents that it contains. Context may also include system capabilities, services offered and sought, the activities and tasks in which people and computing entities are engaged, and their situational roles, beliefs, and intentions.“ [Che04]*

Die Definition lässt, im Gegensatz zu Deys und Abowds Definition, den Anwendungszweck offen. Kontext wird als eine ortsbezogene Information verstanden, welche Attribute der Umwelt, der Menschen, Geräte, Objekte und Software-Agenten mit einschließt und auch Systemeigenschaften, Informationen über Dienste sowie die Aktivitäten und Aufgaben von Menschen und datenverarbeitenden Geräten sowie deren Rollen, ihren Glauben (im Sinne von Wissen) und Intentionen (im Sinne von Plänen) berücksichtigt. Für die Verwendung in dieser Arbeit soll lediglich der Ortsbezug von Kontext dahingehend aufgeweicht werden, als dass Kontextinformationen nicht an einen Ort (der physischen Welt) gebunden sein müssen, sondern einen beliebigen Bezugspunkt haben können. Dies kann ein Ort sein, jedoch beispielsweise auch eine Person oder ein Objekt (siehe Definition von Dey und Abowd) sowie ein System, eine Anwendung, eine Aktivität oder auch ganz allgemein eine Information, welche den Kontext von Daten (z.B. Text) repräsentiert.

Durch Aufhebung der Einschränkung auf einen Anwendungszwecks und einen Bezugspunkt wird der Kontextbegriff wieder nahezu allgemeingültig, was in der Literatur kritisiert wird (siehe oben). Henricksen separiert den eigentlichen Kontextbegriff von den Konzepten der Kontextmodellierung und der Kontextinformation [Hen03]. Ihrer Argumentation folgend, wird der oben definierte Kontextbegriff daher als Kontextinformation verstanden und somit dahingehend eingeschränkt, dass Kontextinformationen von Sensoren oder Menschen erhoben und in Form eines Kontextmodells bereitgestellt werden

müssen. Die Art des Modells ist hierbei offen gelassen und dient lediglich dazu, den Daten eine Struktur und Bedeutung zu verleihen.

Zusammenfassend kann Kontext somit als beliebige Information verstanden werden, welche einen Bezugspunkt aufweist und durch Sensoren oder Menschen in Form eines Modells bereitgestellt werden kann. Im Folgenden sollen verschiedene Klassifikationen vorgestellt werden, welche einen Überblick über Arten von Kontextinformationen geben.

2.2.2. Kontextklassifikation

Verschiedene Arten von Kontextdaten haben unterschiedliche Eigenschaften, welche die Methoden der Datenerhebung, -verarbeitung und -nutzung sowie geeignete Anwendungsbereiche bestimmen. So existieren in der Literatur eine Reihe verschiedener Klassifikationssysteme, welche unterschiedliche Systematiken vorschlagen, nach welchen Kontextdaten in bestimmte Klassen einzuteilen sind [Wis09].

Im Rahmen dieser Arbeit soll eine Klassifikation lediglich die unterschiedlichen Aspekte von Kontext verdeutlichen. Aus diesem Grund werden im Folgenden drei Klassifikationen vorgestellt, welche den Kontext hinsichtlich ihrer Bezugspunkte klassifizieren. Eine umfassende Übersicht alternativer Möglichkeiten der Klassifikation findet sich in [KO04].

Schmidt et al. ordnen Kontextinformationen drei verschiedenen Dimensionen zu: Umwelt, Selbst (engl. *Self*) und Aktivität [SAT+99]. Zum Umweltkontext gehören Informationen über die physische sowie die soziale Umwelt einer Person. Der Selbst-Kontext dieser Person beschreibt dann den physiologischen und kognitiven Zustand einerseits, andererseits den Zustand datenverarbeitender Geräte dieser Person, da diese Geräte für den von Schmidt et al. intendierten Anwendungszweck eine besondere Rolle spielen. Der Aktivitätskontext schließlich umfasst das Verhalten der Person sowie deren aktuelle Aufgaben.

Dieser sehr personenzentrierten Klassifikation von Kontext stellen Zimmermann et al. eine allgemeinere, entitätsbezogene Einordnung gegenüber [ZLO07]. Sie ordnen Kontextinformationen bezogen auf eine Entität einer von fünf Klassen zu: Individualität, Zeit, Ort, Aktivität, Beziehungen. Die Arten von Kontextinformationen bezogen auf die Klasse der Individualität sind abhängig von der Art der Entität. Diese Klasse ist vergleichbar mit dem Selbst-Kontext von Schmidt et al., bezieht sich jedoch explizit nicht nur auf Personen, sondern auf alle natürlichen und künstlichen sowie realen und virtuellen Entitäten. Die Klassen Zeit und Ort beziehen sich auf die zeitlich-räumlichen Koordinaten einer Entität, wobei auch hier unter Umständen ein anderes Verständnis von Raum und Zeit, z.B. in virtuellen Umgebungen, angewendet werden kann. Der Aktivitätskontext beschreibt die Ziele von Entitäten und wie diese erreicht werden können, bzw. welche Aufgaben eine Entität gerade ausführt, um ein bestimmtes Ziel zu erreichen. Der Beziehungskontext umfasst die sozialen, funktionalen oder kompositionalen (Aggregation oder Assoziation) Beziehungen einer Entität zu einer oder mehr anderen Entitäten.

Nahezu dieselben Klassen finden sich auch bei der Einteilung von Cassens und Kofod-Petersen [KPC06]. Auch hier gibt es einen persönlichen Kontext (vergleichbar mit dem Selbst- oder Individualitätskontext), einen sozialen Kontext (vergleichbar mit dem Beziehungskontext), einen Aufgabenkontext (vergleichbar mit dem Aktivitätskontext) und einen räumlich-zeitlichen Kontext. Zusätzlich führen Cassens und Kofod-Petersen, genau wie Schmidt et al., noch den Umweltkontext als eigene Klasse ein. Bei Zimmermann et al. ist diese Art von Kontext dem Individualitätskontext untergeordnet.

Alle drei Systematiken umfassen näherungsweise dieselben Arten von Kontext, jedoch in leicht unterschiedliche Klassen aufgeteilt. Auch ist allen Klassifikationen gemein, dass der Kontext an eine Entität gebunden ist. Lediglich die Definition von Schmidt et al. ist auf Personen fokussiert, während die anderen beiden beliebige physische und virtuelle Entitäten einbeziehen. Da nun mit der Definition von Kontext und der Einordnung von Kontextinformationen in unterschiedliche Klassen eine Grundlage für das allgemeine Verständnis gelegt ist, soll im folgenden Abschnitt der Nutzen von Kontext dargelegt werden.

2.2.3. Kontextbewusstsein und -adaption

Auf Basis der Definitionen von Kontext sowie insbesondere auch aus den Klassifikationen von Kontextinformationen lassen sich diese gegenüber allgemeinen, nicht entitätsbezogenen Informationen abgrenzen. Es ist jedoch schwierig, wenn nicht gar unmöglich, eine solche Abgrenzung zu jedem Zeitpunkt vornehmen zu können, da dieselbe Information, sobald sie einen Entitätsbezug bekommt, zum Kontext dieser Entität gehört. Kontext kann daher eher als eine Aussage über die Relevanz von Informationen in einer bestimmten Situation angesehen werden [Wis09, Dou04].

Daraus folgt, dass der grundlegende Nutzen bzw. das Ziel der Verwendung von Kontextinformationen darin liegt, dass man sich einer Situation bewusst werden möchte, mit allen Informationen, die zu einem Zeitpunkt und für einen bestimmten Zweck relevant sind. Der Begriff des Kontextbewusstseins (engl. *Context Awareness*) bedeutet dabei nicht nur, dass Kontextdaten über die aktuelle Situation vorliegen, sondern diese Daten müssen für den Empfänger, der sich einer Situation bewusst werden möchte, auch verständlich und interpretierbar sein. Voraussetzung hierfür ist, dass Daten erst durch Organisation und Strukturierung zu bedeutungs- und wertvollen, zu nützlichen und relevanten Informationen transformiert [RH08] und dem Empfänger anschließend zugänglich gemacht werden.

Ist man sich eines Kontextes bewusst und möchte in Abhängigkeit von diesem bestimmte Aktionen ausführen, spricht man von Kontextadaption (engl. *Context Adaptation*). Nach [RH08] bedarf es hierfür nicht nur Informationen, sondern auch Wissen (engl. *Knowledge*), welches die Transformation von Informationen zu Anweisungen erlaubt. Wissen zeichnet sich durch einen höheren Grad angenommener Gültigkeit sowie unter Umständen einen größeren Bestand an Fakten, Theorien und Regeln aus [Wik13b] und kann über unter-

schiedliche Prozesse gewonnen werden. Hierzu gehören beispielsweise die Synthese von Zeitreihen mehrerer Informationsquellen sowie Erfahrung und/oder Lernen [RH08]. Auf diese Prozesse wird im weiteren Verlauf der Arbeit, insbesondere in den Kapiteln 4 und 7, weitergehend Bezug genommen. Zusammengefasst erfordert das Bewusstwerden eine Interpretation von Daten und eine darüberhinausgehende Adaption bedarf zusätzlichen Wissens.

Die Voraussetzung für eine Interpretation ist, dass die Struktur und Semantik der Daten bekannt ist. Der Urheber und der Empfänger müssen ein gemeinsames Konzept der Daten haben. Aufgrund der großen Heterogenität auf beiden Seiten, bedarf es daher häufig einer Vorverarbeitung der Daten auf dem Weg zwischen Urheber und Empfänger, um die originalen Daten zu organisieren, zu strukturieren und in eine geeignete Repräsentation zu überführen. Dies wird im nächsten Abschnitt 2.3 weitergehend vertieft.

Wozu die Daten auf Empfängerseite letztlich weiter verwendet werden, ist unerheblich. Einerseits können die Informationen lediglich zur Vergewisserung eines Umstands dienen (z.B. Uhrzeit, aktueller Aufenthaltsort, freier Arbeitsspeicher), andererseits, wie oben angedeutet, auch eine Reaktion oder Anpassung auf die aktuelle Situation auslösen [SAW94]. Hierfür wird vielfach auch der Begriff des „intelligenten Systems“ (engl. *Smart System* bzw. *Smart Application*) verwendet, welcher durch [AS09] folgendermaßen beschrieben ist:

> *The Context-aware or smart systems are those that are able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account. [AS09]*

“Smart“ sollte in diesem Zusammenhang weniger mit dem Begriff der „Intelligenz“, als Sammelbegriff für die kognitive Leistungsfähigkeit, sondern nach [MW12] eher mit „knowledgeable“ (dt. kenntnisreich, gut unterrichtet) oder „operating by automation“ (dt. Ausführung durch Automation) übersetzt werden [BL12]. Somit besitzen kontextbewusste Systeme Kenntnis über ihre Umwelt bzw. ihren Kontext und können ihre Operationen entsprechend anpassen bzw. auch automatisiert ausführen, um sowohl die Benutzbarkeit als auch die Effektivität zu verbessern.

Pascoe spricht dabei von unterschiedlichen Aspekten des Kontextbewusstseins [Pas98]: Als grundlegende Fähigkeit nennt er die Wahrnehmung (engl. *Contextual Sensing*), was oben bereits auch unter dem Begriff des Kontextbewusstseins eingeordnet wurde. Auf dieser Basis kann ein System sich an den Kontext adaptieren (engl. *Contextual Adaptation*). Sofern es auch die Fähigkeit besitzt den Kontext anderer Entitäten wahrzunehmen, kann es seinen Wahrnehmungsbereich ggf. vergrößern und weitere für einen Anwendungszweck nützliche Informationen oder Dienste auffinden (engl. *Contextual Resource Discovery*). Neben der Wahrnehmung, Reaktion und Interaktion mit der Umwelt kommt noch eine vierte Fähigkeit hinzu: das kontextbezogene Anreichern der Umwelt mit Informationen (engl. *Contextual Augmentation*). Eine ähnliche Unterteilung findet sich auch in [DA99]. Hier werden kontextbewusste Anwen-

dungen in Abhängigkeit davon, wie Kontextinformationen verwendet werden, in drei unterschiedliche Kategorien eingeteilt: i) Anwendungen, die Kontext präsentieren, ii) Anwendungen, die je nach Kontext automatisch einen Dienst ausführen, und iii) Anwendungen, die Kontext mit weiteren Informationen anreichern können. Die Wahrnehmung und Interpretation des Kontextes wird hierbei ausgeschlossen und stattdessen argumentiert, dass dies auch in Ebenen unterhalb der Anwendungsschicht, beispielsweise durch eine Middleware, geschehen kann [DA99].

Die Möglichkeiten der Reaktion bzw. Adaption sind, sofern sich Menschen eines Kontextes bewusst werden, beliebig und sollen daher im Folgenden nicht weiter behandelt werden. Auch die Möglichkeiten eines Softwaresystems sind mannigfaltig, lassen sich jedoch auf typische Adaptionsreaktionen eingrenzen (siehe [Jös08, Gei11]). Im Rahmen dieser Arbeit spielt die Reaktion auf einen Kontext nur eine untergeordnete Rolle, da es im Wesentlichen um das Bewusstsein bzw. Bewusstwerden geht. Wie bereits angedeutet, bedarf es hierfür der Wahrnehmung, Analyse und Interpretation. Die Wahrnehmung geschieht mittels Sensoren (vgl. Abschnitt 2.1.1), welche Kontextdaten liefern. Für eine Interpretation ist es jedoch häufig notwendig, diese originalen Daten zunächst aufzubereiten, in eine geeignete Darstellung zu bringen und zu analysieren. Im Folgenden sollen daher gängige Verarbeitungsfunktionen vorgestellt werden, welche in vielen Szenarien praktische Anwendung finden.

2.3. Kontextdatenverarbeitung

Um sich eines Kontextes bewusst zu werden und die Daten nutzen bzw. sich ggf. an den Kontext anpassen zu können, bedarf es zunächst einer Analyse und Interpretation sowie einer geeigneten Repräsentation der Daten [EC08]. Auf diese Weise können Informationen und ggf. sogar Wissen aus den Daten abgeleitet werden.

Hierfür ist es in vielen Fällen jedoch erforderlich, zunächst die Rohdaten der Sensoren aufzubereiten. Hierzu gehört beispielsweise das Dekodieren der Daten, z.B. von einem proprietären Binärformat des Produzenten in ein Textformat, sowie weitergehend eine Überführung in eine gängige Beschreibungssprache bzw. ein Datenmodell. Auch das Konvertieren von Einheiten sowie das Anreichern der Daten mit Metainformationen, z.B. semantischen Annotationen, oder zusätzlichen Informationen aus externen Quellen gehören zu gängigen Vorverarbeitungsschritten. Die so vorbereiteten Datensätze können dann als Eingabe für Analysewerkzeuge dienen, zum Beispiel um Muster oder Trends in den Daten zu erkennen und so höherwertige Informationen abzuleiten. Schließlich kann es notwendig sein, die Verarbeitungskette wieder in umgekehrter Reihenfolge zu durchlaufen, um die neu gewonnenen Informationen wieder in eine Repräsentation zu bringen, die auch von den Konsumenten verstanden wird.

Prinzipiell lassen sich beliebige Schritte zur Verarbeitung von Kontextdaten vorstellen. Und diese Schritte lassen sich verschiedentlich kombinieren.

Es ist daher schwerlich möglich eine umfassende und vollständige Aufzählung möglicher Verarbeitungsschritte vorzunehmen. Jedoch finden sich in der Praxis eine Reihe gängiger, anwendungsunabhängiger Verarbeitungsfunktionen [CKDB06, FJK+05], welche im Folgenden kurz dargestellt werden sollen.

Reduktion Sensoren erheben kontinuierlich Messwerte, welche meist in regelmäßigen zeitlichen Abständen veröffentlicht, d.h. in Form einer Nachricht versendet werden. Um Ressourcen zu schonen und die Infrastruktur zu entlasten, kann und sollte die Datenmenge möglichst schon am Anfang einer Verarbeitungskette reduziert werden. Bei der verlustbehafteten Reduktion werden nicht benötigte Datensätze vollständig aus dem Datenstrom entfernt. Auf diese Weise lassen sich frühzeitig als fehlerhaft erkannte Datensätze löschen, z.B. ungültige Messungen oder durch Übertragungsfehler verursachte unvollständige Datenpakete [FJK+05]. Auch redundante Informationen können aus dem Datenstrom entfernt werden: Manche Sensoren versenden wiederholt den gleichen Messwert, auch wenn keine Änderung des gemessenen Umweltattributs erfolgte. Derlei redundante Informationen können unter Umständen gefiltert werden, sodass nur Ereignisse, d.h. Änderungen der Umweltattribute, weitergereicht werden. Je nach Anwendungszweck können zudem spezielle Filter (z.B. Hochpass-/Tiefpass- oder Kalman-Filter) eingesetzt werden, welche Datenwerte aussortieren, die nicht innerhalb bestimmter Grenzbereiche liegen, bzw. Datenreihen glätten [BLHS04, FJK+05, BDF+07].

Auch mittels Kompressionsalgorithmen lässt sich die Datenmenge reduzieren, wodurch sich insbesondere bei Datensätzen mit geringer Entropie das Volumen verringern lässt. Im Gegensatz zu obigen Methoden werden hier allerdings keine (vollständigen) Datensätze aus dem Strom entfernt, sondern im Fall verlustfreier Techniken lediglich das Datenvolumen reduziert und bei Einsatz verlustbehafteter Verfahren irrelevante Details aus den Datensätzen entfernt.

Aggregation Die Aggregation ist eine Sonderform der Reduktion. Zwar werden auch hier komplette Datensätze aus dem Strom entfernt, die Informationen werden jedoch zusammengefasst und gehen somit nicht vollständig verloren [BLHS04, FJK+05]. Das Zählen oder Aufsummieren von Datensätzen, die Mittelwertbildung, das Berechnen von Minima und Maxima sowie der Standardabweichung stellen typische Aggregationsfunktionen dar. Auch eine Klassifikation von Daten kann als eine Art der Aggregation angesehen werden. Hierbei werden Datensätze vordefinierten oder erlernten Klassen zugeordnet. Erfolgt die Klassifikation auf Basis mehrerer Datensätze, so reduziert sich auch das Volumen. Beispielsweise könnte man eine Menge an Beschleunigungsmessungen dahingehend klassifizieren, ob eine Person gerade steht, geht oder rennt und diese Information dann stattdessen für die weitere Verarbeitung nutzen.

Konversion Bei der Konversion wird meistens eine Einheit oder ein Format in eine andere Einheit bzw. ein anderes Format überführt. Gängige Beispiele für Einheiten-Konversionen sind das Überführen einer Temperaturmessung von °Celsius in °Fahrenheit oder einer Währung von Euro in Dollar [FJK+05]. Datenstrukturen, z.B. Bilder, können von einem Format (z.B. JPEG) in ein anderes Format (z.B. PNG) überführt werden, wobei man hier auch von Transkodierung spricht [BLHS04]. Auch Transkriptionen, d.h. die Umschreibung zwischen verschiedenen Noten- oder Schriftsystemen (z.B. von Kyrillisch nach Lateinisch) sowie das Umwandeln gesprochener Sprache in Text, oder Translationen, also das Übersetzen von Texten einer Sprache in eine andere Sprache, sowie die mathematische Verschiebung (z.B. Koordinaten- oder Vektortransformationen) zählen zu gängigen Konversionsfunktionen.

Archivierung Das Archivieren von Kontextdaten, zum Beispiel zur späteren Analyse eines vergangenen Kontextes, ist auch eine gebräuchliche Verarbeitungsfunktion. Im Gegensatz zu allen bisher vorgestellten Funktionen werden die Daten hierbei nicht verändert, sondern lediglich persistiert und anschließend weitergereicht. Das Archivieren kann in unterschiedlichen Formen geschehen. Gängig sind dabei das einfache Festschreiben einer Menge von Messwerten in einer Datei, das Speichern in einer Datenbank oder das Aufrufen eines Dienstes, welcher seinerseits das Archivieren übernimmt.

Augmentation Das Augmentieren, also das Anreichern von Daten mit zusätzlichen Informationen, wird in vielen Anwendungsszenarien benötigt [BLHS04]. Es sind nahezu beliebige konkrete Augmentationsfunktionen denkbar: Anwendungsunabhängig ist zum Beispiel das Anreichern von Daten mit Zeitstempeln, Logging-Informationen oder Leistungsindikatoren. Für bestimmte Arten von Sensordaten lassen sich typische Funktionen identifizieren, z.B. das Anreichern von geographischen Positionsdaten mit einer postalischen Adresse, das Hinzufügen von Informationen aus dem Lesespeicher eines RFID Tags auf Basis des dazugehörigen elektronischen Produktcodes oder das Anhängen bibliographischer Daten im Falle einer ISBN-Nummer.

Es gibt jedoch auch eine Vielzahl von Anreicherungsfunktionen, die anwendungsspezifisch sind und detailliertes Wissen über die Semantik von Sensordaten und mit ihnen verknüpfbaren Informationen erfordern. Beispiele hierfür sind das Anhängen von Stammblättern bei gegebener Kundennummer, das Augmentieren von Namen mit Kontaktdaten oder das Anreichern einer Adresse im World Wide Web mit allen auf dieser Seite befindlichen Querverweisen.

Ableitung Bei einer Ableitung (auch Schlussfolgerung oder Inferenz) werden aus Informationen neue Informationen „gewonnen“. Meist wird hierfür das logische Schließen in Form einer Deduktion, Abduktion oder Induktion verwendet, bei der mittels einer Logik Ursachen, Wirkungen und Gesetze bzw. Regeln in Beziehung zueinander gesetzt werden. Meldet ein Sensor beispielsweise eine erhöhte Feuchtigkeit (Wirkung), könnte mit Hilfe der Regel, dass es bei

Regen feucht wird, auf die Ursache geschlossen werden, dass es geregnet hat (Abduktion). Ein Beispiel für eine Deduktion wäre bei gegebenem Gesetz, dass beim Autofahren nicht telefoniert werden darf, und der Prämisse, dass eine Person gerade im Auto sitzt und fährt, ein ankommender Anruf vom System nicht durchgestellt wird (Wirkung).

Ein weiterer klassischer Anwendungsfall sind semantische Ableitungen. Werden Sensordaten semantisch annotiert, so wird den Daten eine Bedeutung verliehen. Haben zwei Parteien dasselbe Konzept (z.B. eine Ontologie) vorliegen, haben sie auch dasselbe Verständnis dieser Bedeutung. Dabei ist es unerheblich, ob z.B. die eine Partei von einem Temperatursensor und die andere von einem Thermometer spricht. Auch lassen sich logische Schlussfolgerungen dahingehend führen, dass beides zu dem Konzept eines Wettersensors generalisiert wird und somit entsprechende Abfragen beantwortet werden können.

Sonstige Funktionen Wie bereits angedeutet, sind die Verarbeitungsfunktionen, die auf Kontextdaten angewendet werden können, beliebig und lassen sich daher auch nicht umfassend hier aufführen. Es gibt jedoch noch eine Reihe gängiger Funktionen, die nicht in obige Kategorien passen, aber häufig Verwendung finden. Hierzu gehören zum Beispiel das Sortieren einer Menge von Daten nach bestimmten Kriterien, das Verschlüsseln (z.B. bevor die Daten archiviert werden) oder das Erkennen bestimmter Merkmale in den Daten. Dies kann beispielsweise eine Kanten-Detektion in einem Bild als Vorstufe eines Algorithmus zur Kollisionsvermeidung für mobile Sensorplattformen sein oder eine Erkennung von Gesichtern oder allgemein Mustern in Daten [BDF+07]. Auch Prognosen zukünftig zu erwartender Kontexte finden sich häufig in der Praxis, z.B. Wettervorhersagen oder der nächste zu erwartende Aufenthaltsort einer Person.

2.4. Zusammenfassung

Der Zustand einer Entität zu einem bestimmten Zeitpunkt lässt sich (zumindest partiell) mit Hilfe von Sensoren erfassen. Physische, virtuelle und komplexe Sensoren erlauben hierbei die Erhebung nahezu beliebiger Attribute der physischen und logischen Umwelt dieser Entität. Durch Vernetzung der Sensoren untereinander oder über das Internet lässt sich dabei der Skopus der Betrachtung beliebig vergrößern. Indem Sensordaten in Beziehung zu einer Entität gesetzt und interpretiert werden, erhält man Informationen über deren Kontext, d.h. über die Situation in der sich die Entität gerade befindet. Durch eine weitere Aufbereitung, Aggregation und Verknüpfung mit zusätzlichen Informationen können diese Informationen schließlich zu Wissen, das heißt einem Bestand an Fakten, Theorien und Regeln abgeleitet werden.

Das Gewahrsein des eigenen Zustands ist ein erster Schritt hin zu der Fähigkeit einer Entität, sich der aktuellen Situation entsprechend anzupassen bzw. zu verhalten. In Bezug auf Anwendungen bedeutet dies zum Beispiel, dass sich eine Anwendung optimal an den Nutzer und den gegebenen Nut-

zungskontext adaptieren oder bestimmte Aufgaben zu einem geeigneten Zeitpunkt automatisiert durchführen kann.

Um jedoch von den Sensordaten hin zu Kontextinformationen oder sogar zu Kontextwissen zu gelangen, müssen die Rohdaten der Sensoren entsprechend aufbereitet und verarbeitet werden. Je nach Anwendungszweck müssen hierfür eine Reihe von Verarbeitungsfunktionen zum Aggregieren, Konvertieren, Archivieren, Anreichern, Schlussfolgern etc. angewandt werden. Bevor dieser Prozess in Kapitel 4 wieder aufgegriffen und detailliert wird, sollen im folgenden Kapitel zunächst die Grundlagen der mobilen, ubiquitären Systeme gelegt werden, da diese mit ihren spezifischen Charakteristika besondere Herausforderungen an die Verarbeitung mit sich bringen.

3. Grundlagen mobiler, ubiquitärer Systeme

Im vorangegangenen Kapitel wurde der Kontextbegriff eingeführt, es wurden die Fragen geklärt, wie Kontextdaten erhoben und verwendet werden können, und grundlegende Verarbeitungsschritte, die für eine einfache Nutzung von Kontextinformationen erforderlich sein können, aufgezeigt. In diesem Kapitel sollen die Grundlagen der mobilen, ubiquitären Systeme erläutert werden, deren Eigenschaften auch die Basis eines zukünftigen Sensor Webs betreffen. Das Internet heutzutage ist für die kommenden Anforderungen nur bedingt vorbereitet. Diese entstehen im Wesentlichen einerseits durch die riesigen Mengen automatisch (z.B. durch Sensoren) erhobener Daten, andererseits durch mobile Nutzer sowie Dinge als Nutzer. Es sollen daher die Eigenschaften der neuen Nutzerarten und die hieraus entstehenden Anforderungen an die Kommunikation aufgezeigt werden. Dies ist dahingehend von Interesse, als dass die neuen Teilnehmer einerseits kontinuierlich große Mengen an Kontextdaten produzieren, andererseits der Nutzen von Kontextdaten, z.B. zur Anpassung des Verhaltens von Dingen und Anwendungen, stärker in der Vordergrund rückt. Abschnitt 3.1 stellt daher zunächst die Herausforderungen vor, die das Paradigma des Mobile Computings mit sich bringt. Einen Schritt weiter in die Zukunft geht das Ubiquitous Computing, dessen Vision und Wege zur Realisierung dieser Vision in Abschnitt 3.2 erläutert werden.

3.1. Mobile Computing

Nach der zentralisierten Datenverarbeitung durch Mainframe-Computer in den 70er und 80er Jahren, den verteilten Systemen in den 90er Jahren und dem Durchbruch des globalen Internets zur Jahrtausendwende, stellt das Paradigma des Mobile Computing die vierte Evolutionsstufe der rechnergestützten Datenverarbeitung dar [BZ10]. In diesem Abschnitt sollen nach einer Einführung und Begriffsklärung mit den mobilen Geräten und der mobilen Kommunikation zwei wesentliche Aspekte des Mobile Computing näher beleuchtet werden. Die sich daraus ergebenden Herausforderungen für den Entwurf und die Entwicklung mobiler, verteilter Systeme führen zu dem sogenannten mobilen Dilemma, welches den Nutzen von Kontextdaten zur Adaption von Anwendungen an eine dynamische Umgebung motiviert.

3.1.1. Einführung und Begriffsklärung

In der Literatur und im Alltag gibt es eine Vielzahl an unterschiedlichen Auffassungen von der Tragweite des Begriffs *Mobile Computing*. In seiner einfachsten Bedeutung geht es schlicht darum, einen Computer samt aller für eine Aufgabe benötigten Daten an einen anderen Ort mitzunehmen [Uni12].

Andere Definitionen fassen den Gegenstandsbereich etwas konkreter und beziehen sich auf mobile Kommunikation, mobile Geräte sowie Anwendungen für diese Geräte [Rot05]. Zusätzlich können auch die Besonderheiten von mobilen Umgebungen sowie daraus abgeleitete Implikationen für den Zugriff auf Daten und Dienste [Fuc09] zu einem besseren Verständnis des Begriffes beitragen. Im Folgenden soll anhand dreier Definitionen der Begriff des Mobile Computing, so wie er in dieser Arbeit verstanden wird, verdeutlicht werden:

> *„Das Forschungsgebiet Mobile Computing befasst sich sowohl mit Fragen der Kommunikation von mobilen Benutzern (Mobilkommunikation) als auch mit mobilen Endgeräten und den dazugehörigen Anwendungen.“ [Rot05]*

In dieser Definition wird, wie oben bereits angedeutet, der Gegenstandsbereich des Mobile Computing konkretisiert. Es geht hier nicht nur darum, dass ein Gerät samt Daten von einem Ort zum anderen transportiert wird. Vielmehr wird hervorgehoben, dass die Mobilität besondere Anforderungen nicht nur an die Hardware eines Gerätes stellt, sondern auch an die drahtlose Kommunikation, damit man auch unterwegs auf entfernte Daten und Dienste zugreifen kann, sowie die Anwendungen, welche unter anderem neue Bedienkonzepte sowie einen sparsamen Umgang mit Ressourcen berücksichtigen müssen.

> *„Das Ziel des Mobile Computing ist es, den Benutzer und dessen Anwendungen mit effektiven rechnerunterstützten Konzepten, Verfahren und Lösungen zu versorgen, die es ihm ermöglichen, in einem heterogenen Umfeld mit stets unsicherer Verbindungslage (private) Daten und Informationen zu lesen und zu bearbeiten, und dies unabhängig von Ort und Zeit.“ [Fuc09]*

Diese Definition abstrahiert vom Gegenstandsbereich dahingehend, dass nicht mehr wie in vorheriger Definition einzelne zu berücksichtigende Aspekte aufgeführt werden, sondern stattdessen das Ziel definiert wird, aus welchem sich diese (und weitere) Aspekte (z.B. Heterogenität, Dynamik, Sicherheit) ableiten lassen.

> *„Mobile Computing bezeichnet die Gesamtheit von Geräten, Systemen und Anwendungen, die einen mobilen Benutzer mit den auf seinen Standort und seine Situation bezogenen sinnvollen Informationen und Diensten versorgt.“ [BZ10]*

Auch diese Definition fokussiert das Ziel des Mobile Computing, nämlich den mobilen Zugriff auf Daten und Dienste. Sie ergänzt jedoch mit der Berücksichtigung der Situation eines Benutzers einen wesentlichen Aspekt: Da Benutzer und Gerät mobil sind, ändert sich ihr Kontext kontinuierlich. Hierdurch entsteht ein Unterstützungsbedarf seitens des Benutzers, der fortwährend neuen (logischen) Umgebungen ausgesetzt ist und sich in diesen zurechtfinden muss. Um diesem Bedarf beizukommen, müssen sich Gerät und

Anwendungen an die jeweilige Situation adaptieren können, um dem Benutzer auch unterwegs adäquat bei der Bearbeitung von Aufgaben zu unterstützen.

Im Rahmen dieser Arbeit spielen die Hardware mobiler Geräte sowie mobile Kommunikationstechnologien lediglich eine untergeordnete Rolle. Es sollen insbesondere (mobile) Anwendungen fokussiert und dahingehend unterstützt werden, dass diese sich eines Kontextes bewusst werden können. Hierbei geht es nicht zwangsläufig um den Kontext der Anwendung, des Gerätes oder des Benutzers selbst, sondern um den Kontext einer beliebigen (entfernten) Entität, welcher im Rahmen einer Aufgabe relevant ist. Um einen Kontext in geeigneter Weise, d.h. aufgabenadäquat präsentiert, zur Verfügung zu stellen, müssen jedoch die besonderen Herausforderungen, die sich aus den Eigenschaften mobiler Geräte und Kommunikationstechnologien ergeben, berücksichtigt werden. Im Folgenden sollen daher die besonderen Eigenschaften der mobilen Geräte und der mobilen Kommunikation näher erläutert werden.

3.1.2. Mobile Geräte

Mobile Geräte unterscheiden sich von stationären Geräten im Wesentlichen dadurch, dass sie ihren Aufenthaltsort ändern und dennoch weiterhin Aufgaben verarbeiten können. Dies führt zu zwei grundsätzlichen Anforderungen an die Geräte: sie müssen geringe Ausmaße und eine autonome Energieversorgung haben [Fuc09]. Geräte mit diesen Eigenschaften können nach [Rot05] in fünf unterschiedliche Klassen eingeteilt werden:

- Mobile Standardcomputer (z.B. Notebook oder Tablet)
- Bordcomputer (z.B. Motorsteuerung oder Navigationssystem)
- Handhelds (z.B. PDA, GPS-Empfänger, Telefon)
- Wearables (z.B. „intelligente“ Kleidung oder Schmuck)
- Chipkarten und Etiketten (z.B. SIM-Karte oder RFID Chip)

Die letzte Klasse hat zwar meist keine eigene Energieversorgung, verfügt jedoch in der Regel über einen Prozessor, einen internen Speicher und kann programmiert werden. Zusätzlich zu dieser Klassifizierung sollen außerdem noch Sensorknoten, welche, in der Umwelt ausgebracht, autonom Messwerte erheben und kommunizieren können, als weitere Klasse genannt werden (vgl. Abschnitt 2.1.1).

Im Vergleich zu herkömmlichen Computern haben alle Geräte dieser Klassen, neben der bereits erwähnten Tatsache, dass ihre Ausmaße gering sind und sie über eine eigene Energieversorgung verfügen, weitere gemeinsame Eigenschaften und damit verbundene Einschränkungen. Diese sind im Vergleich zu stationären Computern spezifisch für mobile Geräte und nicht durch den Stand der Technologie beeinflusst [Rot05, AGRS05]:

Ressourcen Bedingt durch die geringen Ausmaße verfügen mobile Geräte über eine verhältnismäßig (im Gegensatz zu einem Standardcomputer) geringe Ressourcenausstattung. Da der Energievorrat endlich ist, können lediglich leistungsschwache Prozessoren eingesetzt werden. Auch sind flüchtiger Speicher und Persistenzspeicher relativ klein dimensioniert.

Drahtlose Kommunikation Da die Geräte portabel sind und meist auch unterwegs auf Dienste zugreifen und Daten austauschen sollen, ist eine drahtlose Kommunikation unerlässlich. In den meisten Fällen basiert diese auf Funktechnik. Da jedoch die Ausbreitung von Funksignalen äußeren Einflüssen unterliegt, weisen Funkverbindungen eine hohe Dynamik bzgl. der Verbindungsqualität auf: Datenraten und Latenzzeiten variieren über die Zeit und Verbindungen können unvorhergesehen abbrechen.

Ein-/Ausgabeschnittstelle Aufgrund ihrer geringen Größe besitzen die Geräte auch meist nur eine eingeschränkte Ein-/Ausgabeschnittstelle. Etwas größere Geräte, wie die mobilen Standardcomputer oder Handhelds, werden zunehmend mit berührungsempfindlichen Bildschirmen ausgestattet. Auch die Interaktion mittels Spracheingabe und -ausgabe findet vermehrt Akzeptanz. Kleinere Geräte wie Sensoren oder Chipkarten besitzen meist keine entsprechenden Schnittstellen. Eine Interaktion verläuft ausschließlich mittels des Austauschs von Nachrichten über die Kommunikationsschnittstelle.

Sicherheit Eine Konsequenz der Mobilität ist die Erfordernis Sicherheitsaspekte auf mehreren Ebenen zu beachten. Das Risiko, dass ein mobiles Gerät beschädigt oder gestohlen wird, ist im Vergleich zu stationären Gerät ungleich höher. Der Übertragungskanal ist prinzipiell unsicher, da Funkverbindungen abgehört und manipuliert werden können. Außerdem speichern viele Geräte teils sehr sensible Daten (im Falle von Telefonen z.B. Kontaktlisten und Termine), die auf Anwendungsebene vor unberechtigtem Zugriff geschützt werden müssen.

Betrieb und Interaktion Mobile Geräte sind auf eine endliche Energiequelle angewiesen. Dennoch ist ein Teil der Geräte, z.B. Telefone oder Sensoren, darauf ausgelegt, kontinuierlich in Betrieb zu sein, was lediglich durch fortgeschrittene Energiesparmaßnahmen erreicht werden kann. Während des Betriebs können eine Reihe von Aufgaben im Hintergrund verarbeitet werden, Interaktionen mit dem Benutzer jedoch finden relativ selten und meist nur kurzzeitig statt.

Unter anderem aufgrund dieser Eigenschaften unterliegen die mobilen Geräte besonderen Einschränkungen, welche die wesentlichen Herausforderungen für den Entwurf und die Entwicklung von Anwendungen für mobile Geräte sowie die Kommunikation mit diesen Anwendungen darstellen [AGRS05].

3.1.3. Mobile Kommunikation

In der obigen Definition von [Rot05] wurde die mobile Kommunikation als eines der drei wesentlichen Gebiete des Mobile Computing genannt. Sie ist dahingehend von Bedeutung, als dass die meisten mobilen Geräte (z.B. mobile Standardcomputer oder Telefone) entweder universellen Einsatzzwecken dienen und daher standardmäßig über einen Kommunikationskanal verfügen, ganz speziellen Zwecken dienen (z.B. Sensoren) und als Teil eines Größeren eine eng umrissene Aufgabe erledigen und daher mit anderen kommunizieren müssen oder, wie im Falle von Chipkarten und Etiketten, Kommunikation ein notwendiger Aspekt zur Erbringung einer Funktion darstellt.

Bei der Kommunikation lassen sich grundlegend zwei Arten von Vernetzung unterscheiden [Rot05]:

Ad-hoc Vernetzung Bei der ad-hoc Vernetzung wird spontan eine direkte Verbindung zwischen zwei oder mehr mobilen Geräten aufgebaut, ohne dass die Kommunikation hierbei durch eine Infrastruktur unterstützt wird. Eine solche Verbindung besteht meist nur für einen kurzen Zeitraum, ist lokal beschränkt und idealerweise aufwandsarm zu initiieren.

Mobile Vernetzung Die mobile Vernetzung (engl. *Mobile Networking*, oftmals auch mit *Nomadic Computing* gleichzusetzen) involviert im Gegensatz zur ad-hoc Vernetzung eine Infrastruktur. Das bedeutet, ein mobiles Gerät kommuniziert indirekt mit einem oder mehreren anderen Geräten, wobei eine Infrastruktur eine vermittelnde Rolle übernimmt. Der Zugriff auf das Internet über entsprechende Zugangspunkte stellt hier die häufigste Form der mobilen Vernetzung dar.

Neben dieser Unterscheidung nach Art der Vernetzung hat sich noch eine weitere Unterteilung etabliert, welche ausschließlich die Reichweite der Kommunikation berücksichtigt. So lassen sich beispielsweise *Wireless Personal Area Networks* (WPAN), *Wireless Local Area Networks* (WLAN), *Metropolitan Area Networks* (MAN) oder *Wide Area Networks* (WAN) unterscheiden. All diese Arten von Netzen stellen spezifische Herausforderungen an die bei der Kommunikation verwendeten Konzepte, Protokolle und Technologien und es existieren eine Vielzahl unterschiedlicher Standards, welche die Kommunikationsabläufe in den entsprechenden Bereichen regeln [CDK12, SMZ07, PL05].

Auch wenn eine Reihe von mobilen Endgeräten für den Zugriff auf eine Kommunikationsinfrastruktur standardisierte Protokolle wie das *Internet Protocol* [Com06] oder das *Transmission Control Protocol* [Com06] unterstützen, so stellen mobile Geräte auf nahezu allen Ebenen des ISO/OSI-Referenzmodells besondere Anforderungen [Rot05]. Auf den unteren Ebenen betrifft dies beispielsweise den Zugriff und die verlässliche Übertragung von Daten über ein gemeinsam genutztes Medium. Auf der Vermittlungsebene müssen Besonderheiten der Mobilität berücksichtigt werden, sodass ein mobiles Gerät kontinuierlich eine Verbindung aufrecht erhalten oder eine Verbindung annehmen

kann, auch wenn es seinen Aufenthaltsort ändert. Besonderheiten der Flusskontrolle und Einhaltung von Dienstgüteregelungen stellen neue Anforderungen an die Transportschicht und auf der Anwendungsebene schließlich lassen sich nahezu beliebige Unterstützungspunkte vorstellen, z.B. das automatische Auffinden und Vermitteln von Diensten, kontextabhängiges Auslösen von Aktionen etc.

Zusammen mit der Tatsache, dass drahtlose Verbindungen inhärent störanfällig sind, geringe Datenraten und hohe Latenzen aufweisen, stellen auch diese Aspekte Herausforderungen dar, die bei dem Entwurf und der Entwicklung von Software für mobile Geräte beachtet werden müssen.

3.1.4. Mobiles Dilemma

Wie bereits erläutert, müssen bei der Entwicklung mobiler Systeme insbesondere die Randbedingungen, die durch die Eigenschaften der mobilen Geräte sowie der mobilen Kommunikation vorgegeben sind, berücksichtigt werden. Da die Kommunikationsverbindungen störanfällig sind, geringe Datenraten und hohe Latenzen aufweisen und darüber hinaus auch Kosten verursachen können, könnte eine Schlussfolgerung lauten, dass in einem mobilen System möglichst viel Funktionalität in das mobile Endgerät verlagert werden müsste, damit die Kommunikation auf ein Minimum reduziert werden kann. Auf der anderen Seite sind mobile Geräte jedoch arm an Ressourcen und können leichter beschädigt werden oder abhanden kommen. Unter dieser Prämisse würde es sich anbieten, die Funktionalität in der Infrastruktur anzusiedeln, um die mobilen Geräte zu entlasten. Diesen Widerspruch bezeichnet man als das *Mobile Dilemma* [Fuc09, AGRS05]. Die Frage, welche Seite idealerweise eine Funktion erbringt, lässt sich per se nicht beantworten und ist von der jeweiligen Anwendung abhängig.

Idealerweise sollte man mobile Systeme daher so entwerfen, dass die Anwendungen ihr Verhalten kontextabhängig adaptieren können. Bei guter Verbindungslage könnten Aufgaben in die Infrastruktur delegiert werden, um die Ressourcen des Gerätes zu schonen. Verschlechtert sich die Verbindung, müssen entsprechende Maßnahmen ergriffen werden, um die aktuell in der Infrastruktur ausgeführte Aufgabe und ggf. dazugehörige Daten zurück auf das mobile Gerät zu transferieren, damit eine dortige Weiterausführung ermöglicht wird. Das bedeutet, die Mobilität sollte nicht vollständig transparent für die Anwendungen sein. Hierfür bedarf es anwendungsseitig unter anderem einer möglichst guten Kenntnis des aktuellen Kontextes sowohl des Gerätes als auch der Infrastruktur, welcher zur Laufzeit als Entscheidungsgrundlage für eine mögliche Verlagerung von Funktionen dient [Fuc09].

Die Beschaffung geeigneter Sensordaten zur Bestimmung des Kontextes ist jedoch nicht trivial. Zuerst müssen geeignete Sensordatenquellen aufgefunden und anschließend nach Maßgabe der Anwendungen aufbereitet werden, damit sie in angemessener Form präsentiert werden können. Auch hierbei ist zu beachten, dass die Ressourcen eines mobilen Geräts möglichst wenig bela-

stet werden [Fuc09], was dahingehend schwierig ist, als dass einige der in Abschnitt 2.3 vorgestellten Verarbeitungsfunktionen relativ berechnungsintensiv sind. Das Auffinden der Daten kann beispielsweise über das in Abschnitt 2.1.3 vorgestellte Sensor Web geschehen und wird somit zwangsläufig in die Infrastruktur delegiert, bei der Aufbereitung jedoch gerät man in das mobile Dilemma. Bevor eine Lösung dieses Dilemmas in Form der Kontextdatenvermittlung in Kapitel 4 detailliert betrachtet wird, werden im folgenden Abschnitt zunächst die Grundlagen der ubiquitären Datenverarbeitung vorgestellt, welche weitere Herausforderungen mit sich bringen.

3.2. Ubiquitous Computing

Ubiquitous Computing wird nach der Ära des Mobile Computing als fünfte Evolutionsstufe der Datenverarbeitung angesehen [BZ10, Wik12a]. Bereits jetzt, da die wegbereitenden Technologien langsam Einzug in den Alltag finden, sind einige mit diesem Paradigma verbundene Visionen in greifbarer Nähe. Im Folgenden soll daher die Vision des Ubiquitous Computing sowie die daran angelehnte Idee des Future Internets vorgestellt werden.

3.2.1. Einführung und Begriffsklärung

Der Begriff *ubiquitär* bedeutet soviel wie „überall verbreitet“ [Ins12] bzw. „überall zur selben Zeit existieren bzw. sein“ [MW12]. Etwas ist also omnipräsent, ständig verfügbar und durchdringend bis zum Punkt des Unterbewussten [Bar08]. Im Jahr 1993 publizierte Mark Weiser in einem Artikel seine Vision über die Rolle des Computers im 21. Jahrhundert. Hierin schrieb er:

> *„Ubiquitous Computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.“ [Wei93]*

Der Grundgedanke hierbei ist, dass Computer, als Werkzeuge, noch immer einen sehr großen Aufmerksamkeitsfokus verlangen. Im Gegensatz zu vielen anderen Werkzeugen, z.B. der Schrift, ist man sich der Verwendung dieses Werkzeugs immer bewusst. Wenn Menschen etwas ausreichend gut verstanden und den Umgang damit gelernt haben, hören sie auf, sich diesem bewusst zu sein. Die Werkzeuge bzw. die Benutzung dieser Werkzeuge tritt in den Hintergrund und erlaubt das Fokussieren neuer Ziele über diese Werkzeuge hinaus [Wei91].

Computer sollen also nicht mehr als dedizierte Geräte wahrgenommen, sondern in Alltagsgegenstände integriert werden und diesen eine gewisse Intelligenz verleihen. Somit verschwinden sie aus dem Sichtfeld, werden unsichtbar, bis sie nicht mehr von unserem Alltagsgefüge unterscheidbar sind [Wei91]. Die intelligenten Gegenstände (engl. *Smart Objects*) können ganz natürlich verwendet werden, bieten jedoch eine höhere Qualität dahingehend, dass sie sich

den Bedürfnissen des Nutzers anpassen, autonom Teilaufgaben verrichten und mit anderen intelligenten Gegenständen kooperieren können, um einen weitergehenden Mehrwert zu bieten.

Die Vision des Ubiquitous Computing beschreibt jedoch weniger einen technischen Fortschritt hin zu intelligenten Gegenständen, sondern vor allem ein neues Paradigma, welches die Beziehung zwischen Mensch, Computer und Umwelt tangiert [Fuc09]. Im Gegensatz dazu fokussiert der von der Industrie hervorgebrachte Begriff des *Pervasive Computing* Techniken und Konzepte zur Realisierung von Weisers Vision [Fuc09].

Hierfür bedürfen Gegenstände zunächst einer digitalen Identität, damit sie im Informationsaustausch mit anderen Gegenständen sowie der Umwelt eindeutig und automatisch identifiziert werden können. Außerdem müssen die Gegenstände mit Sensoren ausgestattet werden, damit sie ihren eigenen Kontext, also ihren Zustand sowie den ihrer Umwelt, wahrnehmen können. Gegebenenfalls bedarf es auch Aktuatoren, um auf die Umwelt einwirken zu können. Neue Ein- und Ausgabeschnittstellen sind vonnöten, damit ein Benutzer auf natürliche Weise mit den Gegenständen interagieren kann. Der Informationsaustausch als Grundlage für die Kooperation intelligenter Gegenstände verläuft dann über Netzwerke verschiedener Ausprägungen mit jeweils spezifischen Anforderungen, die sowohl den lokalen als auch den globalen Austausch ohne nennenswerten Konfigurationsaufwand ermöglichen. Und schließlich erlaubt erst eine weitergehende Miniaturisierung all jener Komponenten, dass diese in Alltagsgegenständen untergebracht werden können.

Im Hinblick auf den Informationsaustausch, soll im Folgenden das Internet der Dinge als Teil des sogenannten *Future Internets* vorgestellt werden, welches den globalen Austausch objektbezogener Informationen zwischen Menschen und Objekten (auch Objekten untereinander) ermöglicht und eine Schlüsselrolle auf dem Weg zum Ubiquitous Computing einnimmt.

3.2.2. Future Internet und das Internet der Dinge

Das heutige Internet basiert in grundlegenden Teilen immer noch auf den ursprünglichen Standards und Architekturentscheidungen, die in den siebziger Jahren entwickelt wurden. Die Nutzung des Internets hat sich jedoch dahingehend verändert, dass frühere Entwicklungen einer Neuauflage bedürfen [AIM10, HHM10]. Als Beispiel sei hier die Evolution des *Internet Protocols* genannt, welches über die Jahre mit *Mobile IP* und *IPSec* erweitert wurde, um den neuen Anforderungen an Mobilität und Sicherheit gerecht zu werden, und jetzt mit *IPv6* eine komplette Neuauflage erfährt, um den steigenden Ansprüchen Rechnung zu tragen [Com06, HHM10, BFdM10].

Während zu Beginn des Internets vornehmlich Texte zwischen wenigen Knoten übertragen wurden, wandelte sich sowohl die Art der übertragenen Daten als auch die Anzahl beteiligter Knoten. Auch Ansprüche an die Sicherheit der Daten, die Notwendigkeit mobile Endpunkte einzubeziehen, weitergehende Qualitätsanforderungen zu berücksichtigen, energieeffizient und zuverlässig

zu arbeiten sowie ein selbstorganisierendes Verhalten der Infrastruktur durch Berücksichtigung von Kontextinformationen zu erreichen und höherwertige Dienstleistungen bereits im Netz selbst anzubieten, erfordern einen schrittweisen Umstieg auf zeitgemäße Standards und neue architektonische Ansätze [ZPT+11, HHM10, BFdM10].

Der Begriff *Future Internet* steht dabei ganz allgemein für eine Vielzahl an Forschungsaktivitäten mit jeweils unterschiedlichem Fokus, jedoch gleichem übergeordneten Ziel. Es subsumiert verschiedene Bestrebungen, bspw. das *Internet der Energie*, das *Internet der Dienste*, das *Internet der Menschen*, das *Internet der Medien* sowie das *Internet der Dinge* [HKS08, Eur11]. Das Internet der Dinge, im Rahmen dieser Arbeit die bedeutendste Einflussgröße, fokussiert Forschungsaktivitäten dahingehend, dass es sich auf einen nächsten zukünftigen Entwicklungsschritt konzentriert und somit die zu berücksichtigenden Aspekte einschränkt. In dieser Vision sollen nicht mehr nur Menschen, sondern auch beliebige Dinge zu aktiven Teilnehmern des Internets werden [MF10, JC09]. Im Folgenden soll diese Visionen näher vorgestellt werden.

Das Internet der Dinge Das Internet der Dinge steht für eine vernetzte Welt, in der nicht nur Menschen untereinander Informationen über das Internet austauschen, sondern vor allem auch Dinge mit Menschen sowie Dinge mit Dingen. Schätzungen aus dem Juni 2012 zufolge sind momentan ca. 2,4 Milliarden Computer und mobile Geräte an das Internet angeschlossen [Int12]. Im Zeitalter des Ubiquitous Computing, wenn Dinge mit entsprechenden Technologien ausgerüstet werden, erwartet [IDC12] bis zu 200 Milliarden Teilnehmer.

Die Vision beruht im Wesentlichen auf dem anhaltenden Fortschritt in der Mikroelektronik, Kommunikationstechnik und Informationstechnologie sowie dem Trend zur Miniaturisierung und damit einhergehendem Preisverfall entsprechender Komponenten, sodass diese vielfach auch in Gegenstände des täglichen Gebrauchs integriert werden können und somit den Weg hin zu Smart Objects ebnen [MF10]. Die so angereicherten Dinge können auf Dienste im Internet zugreifen, ihren eigenen Status publizieren, lassen sich ggf. auch aus der Ferne steuern und können zukünftig auch direkt mit anderen Dingen über das Internet kommunizieren, um mit diesen zu kooperieren.

Für diese Vision hat sich der Begriff „Internet der Dinge“ (engl. *Internet of Things*) etabliert, welcher allerdings vielfach unterschiedlich definiert und gebraucht wird. Ein Teil der Definitionen nimmt eine „Internet-orientierte“, ein anderer eine „Ding-orientierte“ und ein dritter Teil der Definitionen eine „Semantik-orientierte“ Perspektive ein [AIM10].

Die Internet-orientierte Sichtweise fokussiert hierbei insbesondere die existierenden und etablierten Internet-Standards, allen voran den IP-Stack. Hier geht es im Wesentlichen um die Erkenntnis, dass Dinge nicht über ausreichend Ressourcen (Rechenleistung, Speicher, Energie etc.) zur Kommunikation über den herkömmlichen IP-Stack besitzen. Aus diesem Grund soll der Stack dahingehend vereinfacht werden, dass auch Dinge über das IP-Protokoll kommuni-

zieren können. Bereits eine solche Vereinfachung soll das existierende Internet zu dem neuen Paradigma aufwerten [AIM10].

Die Auto-ID Labs [Aut12] nehmen in prominenter Position eine Ding-orientierte Sichtweise ein. Sie waren die Urheber des *elektronischen Produktcodes* (EPC) und entwickelten Standards für die automatisierte Erfassung und den Austausch von Objektidentitäten und -daten. Eine Vorreiterrolle nimmt hier insbesondere die *Radio Frequency Identification*-Technologie (RFID) ein, die das massenhafte Auslesen sogenannter RFID Tags, welche eine eindeutige ID tragen, ohne direkten Sichtkontakt erlauben [Rou08]. Durch Standardisierung von Daten- und Schnittstellenmodellen legten die AutoID-Labs (bzw. später *EPCglobal* und *GS1*) die Basis für den globalen Austausch objektbezogener Daten. Dabei beruhen die Standards auf etablierten Internet-Standards wie den IP- und HTTP-Protokollen und dem *Domain Name System* zum Auffinden von Objektdaten [EPC10]. Damit nehmen die Ziele der Auto-IDs Labs zwar eine Schlüsselrolle im Internet der Dinge ein, die Vision geht jedoch noch einen Schritt weiter. Denn nicht nur Identitäten von Objekten, sondern auch weitere Zustandsinformationen der Objekte und ihrer Umwelt (d.h. ihr Kontext), ihre Historie sowie ggf. Anweisungen an die Objekte sollen über das Internet der Dinge kommuniziert werden. Somit spielen auch Sensor- und Aktuator-Netzwerke sowie die hierfür entwickelten Konzepte und Standards (vgl. Kapitel 2) eine wichtige Rolle [MF10].

Die Semantik-orientierte Perspektive fokussiert die Bedeutung von Daten. Die Heterogenität an Dingen im Internet wird sehr groß sein. Wenn Dinge automatisch Dienste oder andere Dinge auffinden und mit diesen kommunizieren können sollen, müssen Aspekte der Repräsentation, des Speicherns, Suchens sowie Verknüpfens und Organisierens von Daten berücksichtigt werden [AIM10]. Eine gemeinsame Übereinkunft über Bedeutungen und Konzepte ist für das gegenseitige Verständnis von Dingen und Diensten unerlässlich.

Diese verschiedenen Sichtweisen fasst folgende Definition des „European Research Cluster on the Internet of Things“ zusammen:

> *„Internet of Things (IoT) is an integrated part of Future Internet including existing and evolving Internet and network developments and could be conceptually defined as a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities, use intelligent interfaces, and are seamlessly integrated into the information network.“ [Eur11]*

Die Erwartungen, die dabei verfolgt werden, lassen sich ebenfalls aus drei Perspektiven betrachten [MF10]. Aus *wirtschaftlicher Sicht* geht es um eine Optimierung der Warenlogistik sowie intra- und inter-organisationalen Prozessen. Zudem sollen neue Geschäftsfelder mit intelligenten Gegenständen sowie damit verbundenen Dienstleistungen eröffnet werden. Diese wiederum führen aus *individueller Sicht* zu den Erwartungen einer angenehmeren, unterhaltsa-

meren, unabhängigeren und sichereren Lebensweise. *Gesellschaftlich* und *politisch* gesehen soll dies schließlich zu einer insgesamt höheren Lebensqualität führen, insbesondere auch durch die Möglichkeit Informationen zu interessanten Aspekten nicht nur jederzeit und allerorten, sondern auch mit größerer Detailtiefe in nahezu Echtzeit erhalten zu können. Somit werden auch intelligente Assistenzsysteme realisierbar, welche Menschen in vielerlei Situationen im Alltag unterstützen können [MF10].

3.3. Zusammenfassung

Der Themenbereich Mobile Computing umfasst die Gesamtheit von mobilen Geräten, Kommunikationstechnologien sowie Anwendungen. Eine Reihe besonderer Eigenschaften ist für diesen Komplex kennzeichnend: Aufgrund der kleinen Bauart sind die Geräte in den ihnen zur Verfügung stehenden Ressourcen relativ beschränkt. Die Kommunikation zeichnet sich durch geringe Datenraten, hohe Latenzen und unvorhersehbare Verbindungsabbrüche aus. Anwendungen haben diese Herausforderungen dahingehend zu berücksichtigen, als dass diese die Benutzerinteraktion sowie ihre Funktion und ggf. ihre Architektur an ständig wechselnde Umgebungen anpassen müssen.

Bei der Vermittlung kontextbasierter Sichten spielen mobile Geräte in zweierlei Hinsicht eine bedeutende Rolle: Auf der einen Seite können sie durch eine Vielzahl eingebauter Sensoren eine Reihe unterschiedlicher Kontextdaten erheben und auf der anderen Seite bedürfen sie (bzw. die auf den Geräten laufenden Anwendungen) Kontextdaten, um sich an immer neue Umstände ideal anpassen zu können. Wie im vorherigen Kapitel 2 aufgezeigt wurde, müssen diese Daten vielfach aufwendig aufbereitet werden, was zu der Frage nach einem geeigneten Ausführungsort für die Aufbereitung und somit zum mobilen Dilemma führt.

Einen Evolutionsschritt weiter geht das Paradigma des Ubiquitous Computing. In dieser Vision verschärfen sich die Herausforderungen dahingehend, dass die Vielzahl mobiler Geräte und der von diesen produzierten Kontextdaten stark zunimmt und das Ziel, die Geräte mit Alltagsgegenständen zu verweben, einen noch höheren Grad der Anpassung seitens der Anwendungen erfordert. Unter anderem mit dem Future Internet und insbesondere mit dem Internet der Dinge, welches Informationen mit digitalen Identitäten von Dingen verknüpft, werden bereits wesentliche Voraussetzungen zur Realisierung dieser Vision geschaffen.

Es bedarf jedoch eines Brückenschlags zwischen der wachsenden Anzahl an mobilen Geräten, dem damit verbundenen massiv ansteigenden Volumen erhobener Sensordaten und der immer größer werdenden Notwendigkeit sich adaptieren zu müssen. Unter Berücksichtigung der geringen Ressourcenausstattung mobiler Geräte müssen Kontextdaten daher in geeigneter Weise zwischen den Produzenten und den Anwendungen vermittelt werden. Diese Herausforderungen sollen im folgenden Kapitel 4 konkretisiert werden.

4. Kontextdatenvermittlung

Nachdem die Grundlagen der mobilen, ubiquitären Systeme und insbesondere der Verarbeitung von Kontextdaten gelegt wurden, soll in diesem Kapitel das Konzept der Kontextdatenvermittlung vorgestellt werden. Hierbei geht es darum, Konsumenten eine einfache Möglichkeit an die Hand zu geben, um Kontextdaten in ihre Anwendungen zu integrieren und gleichzeitig eine Entkopplung von bestimmten Produzenten zu erreichen. Dies wird im Prozess der Kontextdatenvermittlung durch Einsatz eines Vermittlers erzielt, welcher zwischen der herrschenden Heterogenität beider Seiten vermitteln und nach deren Maßgabe zusätzliche Verarbeitungsfunktionen ausführen kann und somit die Verarbeitungslast an sich bindet. Dies soll letztlich auch den Entwicklungsaufwand für Anwendungen reduzieren, die Wiederverwendbarkeit von Verarbeitungsfunktionen unterstützen und somit auch komplexe kontextsensitive Anwendungen (auf mobilen Geräten) ermöglichen.

Ausgehend von einer einführenden Klärung des Vermittlungs- und Prozessbegriffs in Abschnitt 4.1 sowie den an der Vermittlung beteiligten Rollen in Abschnitt 4.2 wird im Anschluss der Ablauf eines Vermittlungsprozesses in Abschnitt 4.3 skizziert. Nachfolgend werden die wesentlichen Aufgaben erläutert, die einzelne Rollen bei der Prozessausführung übernehmen. Hierbei werden insbesondere auch die speziellen Herausforderungen im Kontext mobiler, ubiquitärer Systeme herausgestellt. Auf dieser Basis wird dann im darauffolgenden Kapitel 5 der Vermittlungsprozess um das Konzept kontextbasierter Sichten erweitert, (nicht-)funktionale Anforderungen aufgestellt und für einen Systementwurf geeignete Unterstützungspunkte identifiziert.

4.1. Einführung und Begriffsklärung

Der Vermittlungsbegriff ist sowohl in der Literatur als auch im täglichen Sprachgebrauch durch verschiedene Bedeutungen überladen. Es bedarf daher einer Abgrenzung, Einordnung und Definition dieses Begriffes, um die Bedeutung im Rahmen dieser Arbeit zu konkretisieren.

Nach [Ins12] bedeutet „vermitteln“ (unter anderem)

- *dafür sorgen, dass jemand etwas, was er anstrebt, bekommt*
- *jemandem verständlich machen, mitteilen, zeigen*

In der ersten Bedeutung spricht man beispielsweise von einer „Telefonvermittlung“ als einer Instanz, die einen Anrufer zu einem gewünschten Gesprächspartner weiterleitet, wobei ggf. auf zusätzliches Wissen, z.B. über die aktuelle Erreichbarkeit oder die entsprechende Durchwahl, zurückgegriffen

wird. Ähnliches gilt für eine „Partnervermittlung“, deren Aufgabe es ist, Menschen anhand ähnlicher Interessen und sonstiger Merkmale einander vorzustellen. Und Aufgabe einer „Arbeitsvermittlung“ ist es schließlich, eine passende Aufgabe für einen Arbeitssuchenden zu finden, wobei wiederum zusätzliches Wissen über Anforderungen des Jobs sowie Fähigkeiten des Suchenden herangezogen werden.

In der zweiten Bedeutung spricht man beispielsweise von der „Vermittlung von Wissen“, wenn eine Person versucht Sachverhalte in verständlicher Form einer anderen Person beizubringen, wobei „verständlich“ ggf. eine Aufbereitung der Inhalte oder eine besondere Methodik der Übermittlung impliziert. Gleiches gilt für die „Vermittlung von Eindrücken“, wobei versucht wird, eine Vorstellung oder Wahrnehmung in Worte, Bilder oder andere Medien, die einen subjektiven Eindruck adäquat repräsentieren können, zu fassen.

Der Vermittlungsbegriff umfasst also nicht nur eine reine Delegation von z.B. Informationen zwischen einem Anbieter und einem Anfragenden, sondern beinhaltet auch weitergehende Leistungen wie die Aufbereitung derselben, beispielsweise zum Zweck der besseren Verständlichkeit.

Im Englischen wird „vermitteln“ häufig mit „to mediate“ übersetzt und bedeutet nach [MW12] (unter anderem):

- *to act as intermediary agent in bringing, effecting, or communicating*

Frei übersetzt bedeutet „vermitteln“ also als zwischenliegender Agent in der Erbringung, Beeinflussung oder Kommunikation zu fungieren. Hier ist insbesondere der Begriff des „Agenten“ hervorzuheben, welcher für einen Auftraggeber gemeinhin eine fest umrissene Funktion autonom erfüllen, zur Erfüllung dieser Funktion allerdings sowohl zusätzliches Wissen heranziehen als auch beliebige Unterfunktionen ausführen oder delegieren kann.

Bezogen auf den Begriff der „Kontextdatenvermittlung“ (engl. *Context Data Mediation*) lassen sich daher folgende Aussagen ableiten:

- Jemand (oder etwas) fragt bestimmte Kontextdaten nach.
- Jemand (oder etwas) bietet Kontextdaten an.
- Ein Agent stellt eine Verbindung zwischen diesen Parteien her, wobei zusätzliche Informationen herangezogen und weitere Dienste zur Herstellung dieser Verbindung erbracht oder in Anspruch genommen werden können.
- Die Kontextdaten können für den Anfragenden durch den Agenten aufbereitet oder in besonderer Form übermittelt werden.

Auf Basis dieser Aussagen lässt sich die Vermittlung auch als ein *Prozess* auffassen. Der Prozessbegriff wird nach Johansson folgendermaßen definiert:

> *[A process is] a set of linked activities that take an input and transform it to create an output. Ideally, the transformation that occurs in the process should add value to the input and create an output that is more useful and effective [...] [Joh93]*

Demnach besteht ein Prozess aus einer Reihe miteinander verknüpfter Aktivitäten, welche Eingaben (z.B. Sensordaten) zu höherwertigen Ausgaben (z.B. Kontextinformationen oder kontextbasierten Sichten, vgl. Kapitel 5) umwandeln. Ähnlich wird der Begriff auch von Davenport definiert [Dav93]:

> *[A process is] a structured, measured set of activities designed to produce a specific output for a particular customer or market. [...] A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs [...]*

Mit dieser Definition konkretisiert Davenport den Prozessbegriff dahingehend, dass er fordert, dass ein Prozess ein definiertes Ende haben muss und zielgerichtet für einen bestimmten Klienten umgesetzt wird. Bezogen auf den Vermittlungsprozess bedeutet dies, dass der Prozess vorgibt, wie aus Sensordaten höherwertige Informationen abgeleitet und in eine geeignete Repräsentation überführt werden, damit dem Konsumenten (d.h. dem Auftraggeber) ein adäquater Eindruck des Kontextes vermittelt werden kann (vgl. Abbildung 4.1).

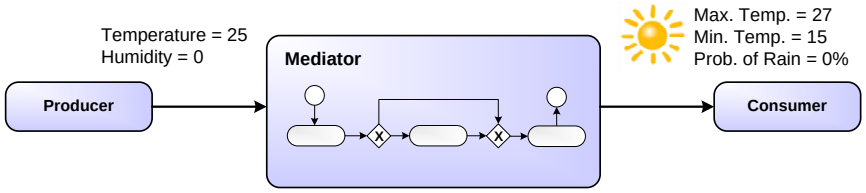


Abbildung 4.1.: Darstellung des Vermittlungsprozesses

Zusammenfassend lässt sich der Begriff der Kontextdatenvermittlung wie folgt definieren:

> ***Kontextdatenvermittlung*** *versteht sich als Prozess, der zu einer Anfrage aufbereitete Kontextinformationen über einen geeigneten Kommunikationskanal liefert. Anfragender und Anbieter werden hierbei durch eine zwischenliegende Instanz, welche den Prozess ausführt, entkoppelt. Zur Ausführung des Prozesses können beliebige zusätzliche Informationen herangezogen und weitere Dienste von der Instanz ausgeführt oder in Anspruch genommen werden.*

Nachdem nun der Begriff der Kontextdatenvermittlung hergeleitet und definiert wurde, werden im folgenden Abschnitt zunächst die Rollen, die an der Vermittlung beteiligt sind, betrachtet, um anschließend die einzelnen Aufgabenbereiche und Aufgaben im Rahmen der Vermittlung zu erläutern.

4.2. Vermittlungsrollen

Bei der Vermittlung lassen sich drei wesentliche Rollen unterscheiden: i) Produzent, ii) Konsument und iii) Vermittler. Da jedoch unter Berücksichtigung

mobiler und ubiquitärer Systeme keine Anforderungen an die Ressourcenausstattung und Leistungsfähigkeit von Produzenten und Konsumenten gestellt werden sollen, diese jedoch im Rahmen des Vermittlungsprozesses mit einem Vermittler auf verschiedene Weise interagieren müssen, wird zudem die zusätzliche Rolle des Stellvertreters eingeführt.

4.2.1. Produzent

Primäre Aufgabe eines *Produzenten* (lat. *producere* „hervorbringen“) ist die Beobachtung bestimmter Aspekte seiner Umwelt sowie die Veröffentlichung von Daten, die diese Aspekte beschreiben. Der Begriff umfasst prinzipiell alle Arten von physischen oder virtuellen Sensoren, kann jedoch nicht synonym verwendet werden, weil manche Sensoren (z.B. eine Uhr oder ein Fieberthermometer) ggf. keine direkte Möglichkeit der Veröffentlichung von Daten haben und daher nur einen Teil dieser Rolle ausfüllen.

Da die Technik hinter der Datenerhebung unerheblich ist, abstrahiert der Begriff des Produzenten hiervon. Während physische Sensoren (z.B. ein Mikrofon) analoge Signale digitalisieren, entfällt dieser Schritt bei virtuellen Sensoren (z.B. zum Beobachten des Arbeitsspeicherverbrauchs). Wichtiger ist die wesentliche Gemeinsamkeit, dass beide periodisch oder auf Anfrage Messwerte erheben und diese veröffentlichen können.

Bei dem Aspekt der Veröffentlichung ist lediglich gefordert, dass Produzenten einen Kanal haben, mittels dessen die erhobenen Daten kommuniziert werden können. Für die Kommunikation eingesetzte Technologien und Standards (z.B. *Ethernet* oder *ZigBee*) sowie verwendete Protokolle (z.B. *TCP* oder *Bluetooth*) sind hierbei nicht relevant. Über einen solchen Kommunikationskanal können zudem nicht nur Messwerte veröffentlicht werden, sondern auch Daten, z.B. Aufträge, empfangen werden. Solche Aufträge können den Produzenten zum Beispiel anweisen, sofort einen Messwert zu veröffentlichen oder einen Aktuator anzusteuern oder sie beziehen sich auf dessen Konfigurationseinstellungen bzw. Metadaten und können den Erhebungs- oder Veröffentlichungsprozess beeinflussen. Beispielsweise lassen sich über solche Aufträge Messfühler kalibrieren, die Erhebungsintervalle ändern oder Kommunikationsendpunkte festlegen.

> ***Produzenten*** *erheben Kontextdaten und veröffentlichen diese über einen Kommunikationskanal. Sie können zudem über diesen Kanal Aufträge entgegennehmen und entsprechend ihrer Fähigkeiten ausführen.*

4.2.2. Konsument

Primäre Intention eines *Konsumenten* (lat. *consumere* „verbrauchen“) ist das Beziehen und Verarbeiten von Kontextinformationen. Anwendungsprogramme, Geschäftsprozesse und Menschen stellen typische Konsumenten dar.

Im Gegensatz zu den Produzenten ist es bei Konsumenten nicht erforderlich, dass diese über einen Kanal zur Kommunikation über ein Netzwerk verfügen,

obwohl dies typischerweise der Fall ist. Der Kanal zwischen Konsument und Vermittler wird einerseits zur Übermittlung von Anfragen seitens des Konsumenten genutzt, andererseits zur Übermittlung der Ergebnisse. Eine ortsbasierte Anwendung zum Rufen eines Taxis wäre ein Beispiel hierfür. Aber auch ohne Zugriff auf das Netzwerk lassen sich Kontextinformationen konsumieren, z.B. die Bestellung eines Kontoauszugs bei der Bank, die der Kunde per Telefon aufgibt und als Brief zugeschickt bekommt.

Darüber hinaus ist es auch nicht relevant, wie und ob ein Konsument die bezogenen Kontextinformationen weitergehend verarbeitet. Im Rahmen des Vermittlungsprozesses wird ihm lediglich diese Absicht unterstellt und diese durch ein Angebot gängiger Dienste zur Erzeugung individueller kontextbasierter Sichten (vgl. Kapitel 5), d.h. aufbereiteter Kontextinformationen, unterstützt.

> ***Konsumenten*** *beziehen aufbereitete Kontextinformationen als Ergebnis auf eine entsprechende Anfrage.*

4.2.3. Vermittler

Eine Verbindung zwischen Produzenten und Konsumenten stellt ein *Vermittler* (lat. *Mediator* „Mittler“) her. Dieser hat im Wesentlichen die Aufgabe, Kontextdaten der Produzenten und Anfragen der Konsumenten entgegenzunehmen und entsprechend zu verarbeiten bzw. die Verarbeitung zu delegieren. Die Art der Verarbeitung hängt dabei maßgeblich vom intendierten Anwendungszweck ab, lässt sich aber auf eine Reihe häufig gebräuchlicher Dienste reduzieren (vgl. Abschnitt 2.3). Hierbei ist insbesondere die Datenstromverarbeitung relevant, deren Aufgabe es ist, Abfragen auf kontinuierlich eintreffenden Daten auszuführen und relevante Daten für die Konsumenten herauszufiltern. Des Weiteren muss ein Vermittler eine Reihe von Diensten zur Unterstützung der Prozessausführung realisieren, z.B. Persistenzdienste, eine Metadatenermittlung und -verwaltung zur Suche nach Produzenten, Zeitreihen und Verarbeitungsdiensten, Dienste für den Zugriffsschutz und die Ressourcenverwaltung sowie Kommunikationsdienste zum Bereitstellen der Ergebnisse.

Ein Vermittler entkoppelt somit Produzenten und Konsumenten. Dies ist insofern wichtig, als dass auf beiden Seiten eine große Heterogenität bzgl. Hard- und Software, aber auch hinsichtlich der Anforderungen sowie der Protokolle und Datenformate herrscht. Als vermittelnde Instanz kann der Mediator somit ungeachtet der technischen Diversität den Konsumenten einen transparenten Zugriff auf Daten der Produzenten ermöglichen, eine n:m-Beziehung zwischen Produzenten und Konsumenten herstellen und darüber hinaus durch Bereitstellen verschiedenster Verarbeitungsdienste die Aufbereitung von Kontextdaten unterstützen.

> ***Vermittler*** *beziehen im Rahmen eines Vermittlungsprozesses Kontextdaten von Produzenten und stellen diese auf Anfrage von Konsumenten in aufbereiteter Form wieder zur Verfügung. Hierfür können eigene Verarbeitungsdienste angeboten oder die Dienstausführung an Dritte delegiert werden.*

4.2.4. Stellvertreter für Produzenten und Konsumenten

Wie oben beschrieben, sollen im Rahmen dieser Arbeit geringstmögliche Anforderungen an Produzenten und Konsumenten gestellt werden. Produzenten müssen lediglich Kontextdaten erheben und in einem Netzwerk veröffentlichen können. Konsumenten müssen über einen beliebigen Kanal eine Anfrage stellen und Ergebnisse beziehen können. Aufgrund dieser geringen Anforderungen ist es auch äußerst ressourcenschwachen und sogar mobilen Instanzen möglich, solche Rollen auszuüben.

Allerdings sind die wenigsten Produzenten und Konsumenten per se für das Zusammenspiel mit einem Vermittler befähigt. Man müsste diese entsprechend anpassen, um z.B. eine Registrierung von Metadaten oder Anfragen bei einem Vermittler zu ermöglichen. In den wenigsten Fällen ist dies eine gangbare Lösung. Daher bedarf es der Rolle des *Stellvertreters*, der im Auftrag eines Produzenten oder Konsumenten vermittlungsspezifische Funktionen übernimmt.

Ähnliche Konzepte finden sich auch beispielsweise bei den Entwurfsmustern aus dem Bereich der Softwareentwicklung bzw. des entfernten Methodenaufrufs [TS07]. Als *Stub* bezeichnet man hier einen lokalen Platzhalter, welcher als Vermittler lediglich eine delegierende Funktion erfüllt und lokale Anfragen an eine (meist) entfernte Komponente, welche die eigentliche Funktionalität bereitstellt, weiterleitet. Eine ähnliche Aufgabe nimmt auch ein sogenannter *Proxy* wahr. Dieser muss jedoch nicht zwangsläufig lokal vorhanden sein, sondern kann die Kommunikation über verschiedene Instanzen innerhalb eines Netzes vermitteln [AIM10]. Im Gegensatz zu einem Stub kann ein Proxy zudem weitere Logik implementieren, z.B. das Filtern und Aggregieren von Nachrichten oder das Registrieren eines Klienten bei einem Vermittler. Ein *Adapter* hat im Wesentlichen die Aufgabe zwischen zwei verschiedenen Schnittstellen zu vermitteln, z.B. zwischen der proprietären Schnittstelle eines Produzenten und der offenen Schnittstelle eines Vermittlers [GHJV11]. Je nachdem welche Aufgaben der Stellvertreter im Rahmen der Vermittlung übernehmen soll (und welche beim Klienten verbleiben), nimmt der Stellvertreter die Rolle entweder eines Stubs, eines Proxies oder eines Adapters ein. In den meisten Fällen wird der Stellvertreter jedoch als Proxy agieren, wie die beiden folgenden Beispiele verdeutlichen.

Eine handelsübliche IP-Kamera beispielsweise kann Fotos über das IP-Protokoll in einem Netzwerk veröffentlichen, jedoch sind diese ohne zusätzliche Metadaten (z.B. über den Besitzer, den aktuellen Standort etc.) nicht allgemein zu gebrauchen. Um in einen Vermittlungsprozess integriert werden zu können, bedarf es eines Stellvertreters, der administrative Funktio-

nen, wie die Registrierung der Kamera bei einem Vermittler und die Metadatenverwaltung, übernimmt.

Auch Konsumenten bedürfen in vielen Fällen eines Stellvertreters, der sich zumindest um die Anfragestellung kümmert. Moderne Fernseher bieten beispielsweise häufig eine integrierte Aufnahmefunktion, die über das Internet aktiviert werden kann. Stellvertretend für den Fernseher könnte eine Person eine Anfrage stellen, die einen Vermittler anweist eine entsprechend formatierte Aufnahmenachricht an den Fernseher zu senden, sobald Dokumentationen mit dem Wort „Island“ im Titel laufen.

Wie zu sehen ist, spielen Stellvertreter eine wichtige Rolle im Vermittlungsprozess. Sie sind in der Lage die Funktionen zu übernehmen, die Produzenten oder Konsumenten nicht realisieren können, da sie entweder nicht speziell für die Vermittlung entworfen wurden, zu ressourcenarm sind oder es sich - wie im obigen Konsumentenbeispiel - anbietet, diese Anfragestellung von geographisch separierten Orten, z.B. aus dem Urlaub, vorzunehmen.

> ***Stellvertreter*** *übernehmen vermittlungsrelevante Funktionen für Produzenten oder Konsumenten.*

4.2.5. Beispiel für Vermittlungsrollen

In Abbildung 4.2 wird das Zusammenspiel der einzelnen Vermittlungsrollen anhand eines Anwendungsszenarios veranschaulicht. Auf der linken Seite findet sich ein Wireless Sensor Network bestehend aus mehreren Sensoren. Diese Sensoren können Temperatur-, Helligkeits- und Beschleunigungsdaten erheben und per Funk zu einer Basisstation übertragen. Die Basisstation ist per USB an einen Computer (in der Mitte) angeschlossen, welcher wiederum per WLAN mit dem Firmennetz verbunden ist. Auf der rechten Seite findet sich ein fliegender Quadrokopter mit hochauflösenden Kameras, welcher über ein WLAN-Netz Befehle in einem proprietären Format empfangen kann.

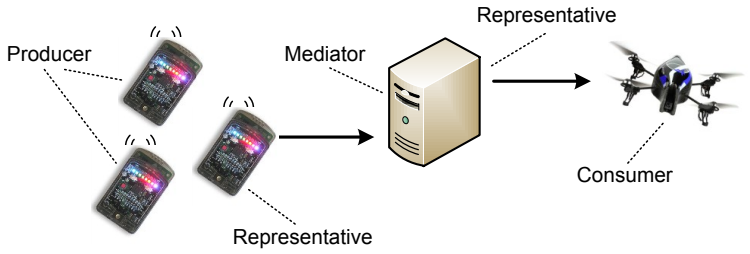


Abbildung 4.2.: Beispiel für Vermittlungsrollen

In einem Bürogebäude soll eine kostengünstige Einbruchsüberwachung installiert werden. Zu diesem Zweck werden an den Bürotüren der Mitarbeiter jeweils ein Sensor angebracht. Die Basisstation wird an einem Mitarbeiter-PC, der per WLAN an das Firmennetz angebunden ist, angeschlossen. Der Quadrokopter ist in einem kleinen Unterstand auf dem Dach des Gebäudes angesiedelt. Die Regel zur Einbruchsüberwachung lautet: „Sobald es in einem Büro

dunkel ist und zwischen 20 Uhr und 6 Uhr eine Beschleunigung der Bürotür registriert wird, soll der Quadrokopter zu dem Bürofenster fliegen und hochauflösende Videoaufnahmen machen“.

Die Sensoren stellen die Produzenten (engl. *Producer*) dar. Da jedoch nur die Basisstation mit dem Computer (per USB) kommunizieren kann, repräsentiert sie einen Stellvertreter (engl. *Representative*), der alle anderen Sensoren vertritt. Der Quadrokopter ist ein Konsument (engl. *Consumer*), welcher im Falle eines Einbruchs benachrichtigt werden und mit den Koordinaten des zu beobachtenden Raumes versorgt werden soll. Da der Quadrokopter selbst keine Abfragen formulieren kann, bedarf auch dieser eines Stellvertreters: den Computer. Über den Computer kann ein Mitarbeiter obige Regel formulieren. Diese wird bei einen Vermittler (engl. *Mediator*) registriert, welcher die Kontextdaten aller Sensoren über deren Stellvertreter entgegennimmt, die Regel auf die Kontextdaten anwendet und bei Übereinstimmung der Bedingung (Beschleunigung der Tür bei Dunkelheit in der Nacht) die entsprechende Aktion (Quadrokopter befehligen) ausführt.

4.3. Ablauf des Vermittlungsprozesses

Der Vermittlungsprozess wird von einem Konsumenten (oder dessen Stellvertreter) initiiert, sobald Kontextinformationen benötigt werden (vgl. Abbildung 4.3). Hierzu muss eine Anfrage formuliert werden, die nicht nur die gewünschte Information beschreibt, sondern auch eventuelle Aufbereitungs- bzw. Verarbeitungsvorschriften für kontextbasierte Sichten (engl. *Context-based View*, vgl. Abschnitt 5.1), die vorschreiben, wie z.B. Ergebnisse präsentiert und übermittelt werden sollen (vgl. Abschnitt 4.1). Die Formulierung kann durch eine (Meta-)Datensuche unterstützt werden, indem z.B. verfügbare Produzenten, Dienste und Informationen abgerufen werden können. Eine abschließend formulierte Anfrage wird über einen Kommunikationskanal an einen Vermittler übermittelt.

Ein Vermittler unterscheidet im Wesentlichen zwei verschiedene Abfragetypen, die sich durch ihre zeitliche Dimension voneinander abheben. Auf der einen Seite stehen Abfragen, die sich auf vergangene Datenwerte oder Zeitreihen beziehen (z.B. Niederschlagsmengen der letzten 10 Tage in Hamburg) und auf der anderen Seite stehen Abfragen, die in die Zukunft gerichtet sind (z.B. sende eine Nachricht, sobald die Temperatur in Hamburg 30° Celsius übersteigt). In dem ersten Fall ruft ein Vermittler entsprechende Daten oder Zeitreihen vergangener Messungen in Datenbanken ab, in dem zweiten Fall registriert er die Abfrage für die sogenannte Datenstromverarbeitung (engl. *Data Stream Processing* [CJ09]), welche kontinuierlich versucht die Abfrage anhand neu eintreffender Kontextdaten zu beantworten.

Steht ein Ergebnis für die Abfrage bereit, so können die Daten nach Maßgabe des Konsumenten weitergehend verarbeitet werden, damit diesem die Informationen in gewünschter Form bereitgestellt werden können (vgl. Abbildung 4.3, *High-level Data Processing*). So kann z.B. die Meldung, dass die Tempe-

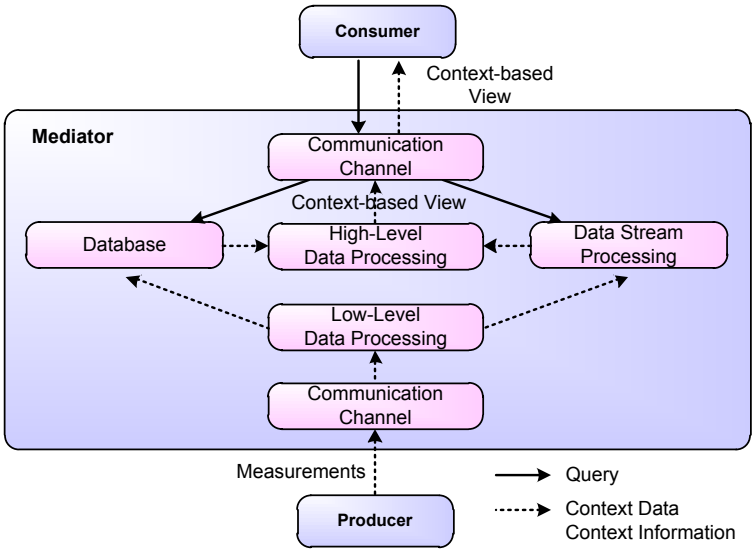


Abbildung 4.3.: Ablauf des Vermittlungsprozesses

ratur in Hamburg 30° überschritten hat, mit zusätzlichen Informationen zur Gesamtwetterlage angereichert und in ein entsprechendes Format konvertiert werden. Das Resultat der Verarbeitung muss dem Konsumenten schließlich über einen geeigneten Kommunikationskanal zur Verfügung gestellt werden.

Damit die Kontextdaten des Produzenten in Datenbanken gespeichert oder von der Datenstromverarbeitung behandelt werden können, muss sich ein Produzent zunächst bei einem Vermittler registrieren (oder durch einen Stellvertreter registriert werden). Bei der Registrierung kann ein Produzent ebenfalls Aufbereitungs- bzw. Verarbeitungsvorschriften angeben, die den Vermittler anweisen, die zukünftig empfangenen Messwerte nach Maßgabe des Produzenten zu verarbeiten (vgl. Abbildung 4.3, *Low-Level Data Processing*), z.B. mit Daten geographisch benachbarter Produzenten zu fusionieren, bevor sie der Datenstromverarbeitung zugeführt oder in einer Datenbank gespeichert werden.

Sobald ein Produzent registriert ist, kann er Messwerte an den Vermittler über einen Kommunikationskanal übertragen. Dieser verarbeitet die Daten entsprechend der registrierten Anweisungen, speichert sie ggf. in einer Datenbank und leitet sie zur Datenstromverarbeitung weiter. Dort werden alle Abfragen aller subskribierten Konsumenten gegen die eintreffenden Daten evaluiert. Führt die Evaluation einer Abfrage zu einem Ergebnis, werden die von den Konsumenten vorgegebenen Verarbeitungsschritte ausgeführt und das finale Resultat als kontextbasierte Sicht an den Konsumenten übermittelt.

4.4. Aufgaben im Rahmen des Vermittlungsprozesses

Nachdem nun der grobe Ablauf der Vermittlung skizziert wurde, soll in diesem Abschnitt der Prozess mit seinen unterschiedlichen Bestandteilen näher vorgestellt werden. Der Gesamtprozess lässt sich in verschiedene Aufgaben unterteilen, welche unterschiedlichen Aufgabenbereichen zugeordnet werden

können. Diese Unterteilung lässt sich einerseits in Hinblick auf die in Abschnitt 4.2 erläuterten Rollen vornehmen, andererseits jedoch auch durch die unterschiedlichen Dienste, die im Rahmen der Vermittlung ausgeführt werden. Allerdings sind die einzelnen Aufgaben nicht immer scharf voneinander zu trennen: Auf der einen Seite können einige Aufgaben durchaus auch von unterschiedlichen Rollen erbracht werden und auf der anderen Seite lassen sich auch Teilfunktionalitäten unterschiedlichen Aufgaben zuordnen.

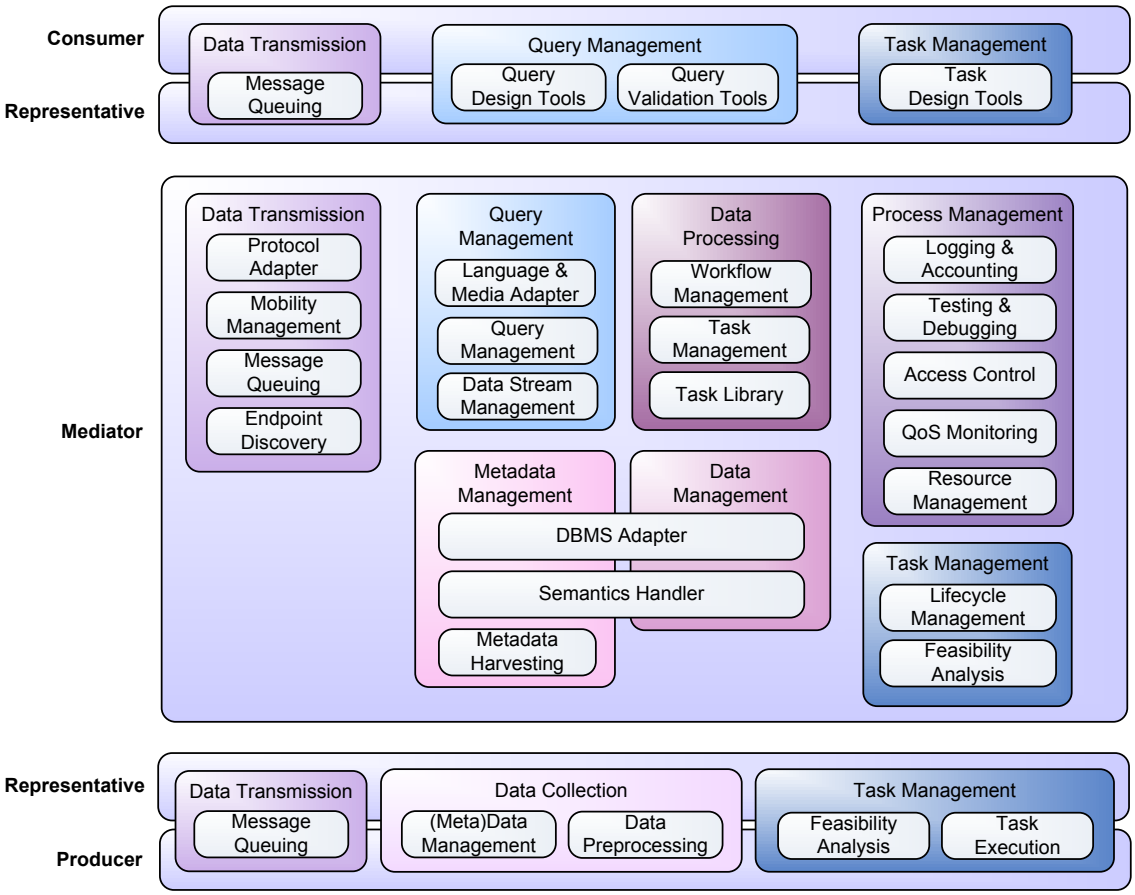


Abbildung 4.4.: Aufgabenbereiche und Aufgaben im Vermittlungsprozess

Abbildung 4.4 setzt die beteiligten Rollen sowie die Aufgabenbereiche und Aufgaben miteinander in Beziehung.

Datenerhebung In diesen Aufgabenbereich (engl. *Data Collection*) fällt neben dem Erheben von Sensordaten auch eine mögliche Vorverarbeitung der Daten durch den Produzenten sowie die Beschreibung dieser Daten und des Produzenten selbst durch Metadaten.

Datenübermittlung Sowohl Sensordaten von Produzenten als auch Abfragen und Aufträge von Konsumenten werden über einen Vermittler kommuniziert. Um der Heterogenität der Beteiligten sowie den eingesetzten Technologien Rechnung zu tragen gehört die Übersetzung zwischen unter-

schiedlichen Protokollen neben der Unterstützung für mobile Klienten zu den Kernaufgaben der Datenübermittlung (engl. *Data Transmission*).

Datenverarbeitung Um den Anforderungen der Produzenten und Konsumenten gerecht zu werden, kann ein Vermittler Sensordaten nach deren Maßgabe aufbereiten und somit die Verarbeitungslast zur Erstellung geeigneter Repräsentationen in die Datenverarbeitung (engl. *Data Processing*) und somit ggf. in die Infrastruktur verlagern.

Datenverwaltung Wesentliche Aufgabe der Datenverwaltung (engl. *Data Management*) ist das Persistieren von Sensordaten-Zeitreihen. Aufgrund der hohen Anforderungen durch Heterogenität, Volumen, Eigenschaften und Zugriffsarten der Daten bedarf es insbesondere der Unterstützung verschiedenster Persistenzmechanismen sowie der Möglichkeit semantischer Abbildungen und Ableitungen.

Prozessverwaltung Aufgaben der Prozessverwaltung zielen auf einen effektiven und effizienten Ablauf des Vermittlungsprozesses sowie dessen Administration. Hierzu gehören auch Aufgaben der Dienstgüteüberwachung, Zugriffskontrolle, Protokoll- und Abrechnungsaufgaben sowie die Ressourcenverwaltung.

Metadatenverwaltung Die Metadatenverwaltung (engl. *Metadata Management*) hält Daten über Produzenten, Konsumenten, Stellvertreter, Verarbeitungsdienste etc. vor. Aufgrund unterschiedlicher Zugriffsmuster und Anforderungen an die Datenhaltung gibt es eine Trennung zwischen der Verwaltung von Zeitreihen und Metadaten, obwohl beide ähnliche Funktionen zu erfüllen haben.

Abfrageverwaltung Dieser Aufgabenbereich der Abfrageverwaltung (engl. *Query Management*) dient wesentlich der Unterstützung einer Vielzahl von Sprachen für historische, ereignisbasierte und ad-hoc Abfragen sowie der Möglichkeit Abfragen über verschiedene Medien und auf unterschiedlichen Datenbank- bzw. Datenstrommanagementsystemen zu stellen.

Auftragsverwaltung Um Konsumenten die Möglichkeit zu geben, Einfluss auf die Produzenten bzw. deren Datenerhebung zu nehmen, bietet ein Vermittler über die Auftragsverwaltung (engl. *Task Management*) Möglichkeiten, Aufträge an die Konsumenten zu versenden und übernimmt deren Verwaltung samt Durchführbarkeitsanalysen.

Zusammengefasst beginnt der gesamte Prozess der Kontextdatenvermittlung also mit der Erhebung von Kontextdaten auf Seite des Produzenten und der Übertragung an einen Vermittler. Dieser bereitet die Daten auf und führt schließlich die Abfragen der Konsumenten auf diesen Datenströmen aus. Die Ergebnisse werden letztlich von den Konsumenten zur Analyse und Integration in übergeordnete Anwendungen und Prozesse verwendet.

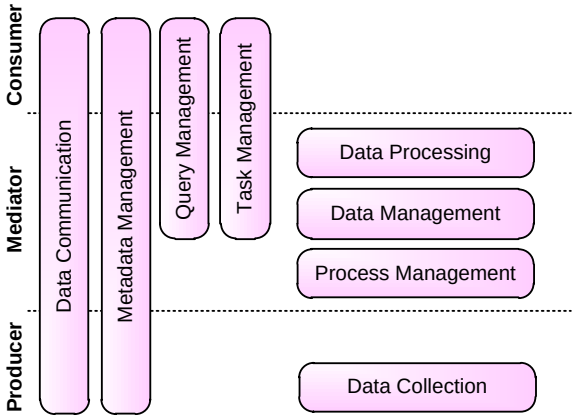


Abbildung 4.5.: Aufgabenbereiche und Rollenverteilung im Vermittlungsprozess

Alle Funktionen, die Bestandteil des Vermittlungsprozesses sind, lassen sich unterschiedlichen Aufgabenbereichen zuordnen, welche in der Verantwortung von einer oder mehreren Rollen liegen (siehe Abbildung 4.5). So tangieren die Bereiche der Datenübermittlung und der Metadatenverwaltung alle Rollen, während die Datenabfrage und die Auftragsverwaltung lediglich dem Konsumenten und dem Vermittler zufallen. Die Datenverarbeitung sowie die Daten- und Prozessverwaltung liegen alleinig in der Verantwortung des Vermittlers. Die Datenerhebung ist genau genommen unabhängig vom Vermittlungsprozess, wird hier jedoch auch aufgeführt, da sie Unterstützungspunkte bietet, an denen der Vermittlungsprozess anknüpfen kann (vgl. Abschnitt 5.4). Die folgenden Unterabschnitte gliedern sich nach diesen Aufgabenbereichen und beschreiben die jeweils einem Bereich zufallenden Aufgaben detaillierter.

4.4.1. Datenerhebung

Unter *Datenerhebung* versteht man das Erfassen von Messwerten bzw. Sensordaten, welche nach Dey et al. [DA99] als jedwede Art von Informationen, die zur Charakterisierung der Situation einer Entität verwendet werden können, definiert sind (vgl. Abschnitt 2.2.1). Bei der Erhebung von Daten lassen sich physische von virtuellen Sensoren unterscheiden. In physischen Sensoren sorgen spezifische Bauteile und Analog-Digital-Konverter dafür, dass analoge physikalische, chemische oder biologische Signale aufgenommen und digitalisiert werden [HC09, SMZ07]. Erst in digitalisierter Form werden diese Daten im Sensor weitergehend verarbeitet und ggf. schließlich kommuniziert. Virtuelle Sensoren unterscheiden sich in der Hinsicht von physischen Sensoren lediglich dadurch, dass anstatt physikalischer, chemischer oder biologischer Signale der Zustand digitaler Entitäten (z.B. Programmvariablen oder Dateieinträge) „gemessen“ wird und entsprechend eine Digitalisierung nicht erforderlich ist.

Während die Digitalisierung in physischen Sensoren durch Hardware-Bauelemente realisiert wird, ist die weitergehende Verarbeitung (unabhängig

von der Art des Sensors) in der Regel Aufgabe der Software, also des Betriebssystems, einer Middleware oder einer Applikation. Aufgrund der Vielzahl unterschiedlicher Einsatzzwecke haben sich ganz unterschiedliche Betriebssystem- und Middleware-Ausprägungen speziell für Sensoren entwickelt. Entsprechend mannigfaltig sind auch die potentiellen Funktionen zur Datenvorverarbeitung (engl. *Data Preprocessing*), die auf Sensoren realisiert werden können (vgl. Abschnitt 2.3). Hierzu gehören beispielsweise das Filtern und Säubern der Messwerte, die Aggregation von mehreren Messwerten (auch unterschiedlicher Quellen), das Konvertieren in bestimmte Formate, sowie das Speichern von Messwerten und Beantworten externer Datenabfragen [HC09, AV10].

Eine Reihe von Betriebs- und Middleware-Systemen erlaubt zudem die äußere Einflussnahme auf bestimmte Verarbeitungsfunktionen, auf Konfigurationseinstellungen des Sensors oder auf an einen Sensor angeschlossene Aktuatoren, was auch als *Tasking* bezeichnet wird [BEJ+11]. So können z.B. Entscheidungen hinsichtlich der Erhebungszeitpunkte, -zeiträume und -orte auch während der Erhebung an neue Umstände adaptiert, ein Sensor rekalibriert oder ein angeschlossener Aktuator aktiviert werden. Beispielsweise erlauben *Mires* [SaV+04] und *TinyDB* [MFHH05] das entfernte Abfragen von Sensorwerten. *Agilla* [FRL06] und das darauf basierende *In-Motes* [GB08] ermöglichen das Ausführen beliebiger Programme mittels mobiler Agenten, die zur Laufzeit auf den Sensor migrieren. Diese Einflussnahme ist auch im Rahmen des Vermittlungsprozesses von Bedeutung und wird in Abschnitt 4.4.8 weitergehend detailliert.

Weiterhin bedarf es der Verwaltung von Metadaten über einen Sensor. Hierzu gehören neben einer eindeutigen Kennung des Sensors beispielsweise auch Informationen über die aktuelle Position, die Messwerte (z.B. Art, Format), die Genauigkeit der Erfassung und die Erhebungsintervalle (weitere Beispiele für Metadaten finden sich u.a. in [BEJ+11, BWC09, FJK+05]). Derlei Daten sind insbesondere für Konsumenten von Interesse, um geeignete Produzenten von für sie relevante Kontextdaten zu finden. Auf dieser Ebene zeichnet sich eine einfache *Metadatenverwaltung* (engl. *Metadata Management*) für die Organisation, Speicherung und das Beantworten von Abfragen verantwortlich.

Die Diversität der Informationen zusammen mit der ebenfalls großen Heterogenität der Sensoren bergen auch besondere Herausforderungen bezüglich der Repräsentation von Metadaten und Kontextdaten. Schließlich sollen diese später seitens eines Vermittlers bzw. Konsumenten auch interpretiert werden können. Einfache Sensoren könnten beispielsweise die Metadaten oder Messwerte in einem proprietären Format repräsentieren und sind dann nur mit entsprechender Software des jeweiligen Herstellers analysierbar. Um jedoch Interoperabilität zwischen Sensoren und nachgeschalteten Verarbeitungsfunktionen im Vermittlungsprozess sicherstellen zu können, sollten Sensoren die Daten in standardisierten Formaten repräsentieren. Eine Reihe von Organisationen haben in diesem Bereich entsprechende Standards entwickelt: Beispielsweise hat die IEEE mit *IEEE 1451* [Nat12b, VPG08] eine Sammlung

von Schnittstellen- und Datenformatstandards veröffentlicht und das Open Geospatial Consortiums mit dem *Observations & Measurements*-Standard ein Informationsmodell für Messwerte und mit der *Sensor Model Language* ein Modell für Metadaten spezifiziert [BEJ+11].

4.4.2. Datenübermittlung

Der Aspekt der *Datenübermittlung* beschäftigt sich mit der Kommunikation zwischen Produzenten und Konsumenten bzw. einem Vermittler. Die Kommunikation kann bidirektional erfolgen: Sensordaten werden vom Produzenten in Richtung des Konsumenten übermittelt und Aufträge bzw. Abfragen nehmen den gegenläufigen Weg. Die Kommunikation erfolgt über einen Kommunikationskanal, der nicht zwangsläufig von beiden Rollen gemeinsam unterstützt werden muss. Dies ist der Heterogenität von Geräten und Standards an den Enden des Kommunikationskanals geschuldet. Da (mobile) Produzenten häufig sehr beschränkte Ressourcen haben, wurden für die Datenübertragung auf dieser unteren Ebene spezielle Protokolle entwickelt, die eine möglichst ressourcenschonende Kommunikation gewährleisten sollen. Auf Seiten der Konsumenten hingegen haben sich existierende Internet-Protokolle etabliert. Diesem Umstand ist es geschuldet, dass die Kommunikation zwangsläufig über einen Vermittler laufen muss, der zwischen unterschiedlichen Kommunikationstechnologien und Protokollen vermitteln kann.

Das Hauptproblem besteht dabei in der Vielzahl an Standards und Technologien, die jeweils auf einen Problembereich zugeschnitten und untereinander nicht kompatibel sind. Allein für Wireless Sensor Networks gibt es eine Vielzahl unterschiedlicher Ansätze [AV10]. Die IEEE konzentriert sich hier mit dem *802.15*-Standard zum Beispiel auf die ISO/OSI-Schichten 1 und 2 [Ins13]. Die darüber liegenden Schichten werden von der IETF mit *6loWPAN* [Int10c] und von der ZigBee Alliance mit *ZigBee* [Zig12] adressiert. Zusätzlich bietet die International Society for Automation (ISA) mit *SP100.11a* [Int10b] sowie die HART Foundation mit *WirelessHART* [HAR10] vertikale Lösungen über alle Schichten und schließlich gibt es noch eine Reihe weiterer, nicht standardisierter, proprietärer Mechanismen und Spezifikationen [Wik10], z.B. *Bluetooth* [Blu10], *EnOcean* [EnO13], *EnviroNet* [Gra13] und *MyriaNed* [CHE10]. In größeren Netzwerken und im Internet haben sich auf den ISO/OSI-Schichten 3 und 4 Kommunikationsprotokolle wie *(Mobile) IPv4*, *IPv6*, *UDP* und *TCP* etabliert [Com06, Rot05]. Diese spezifizieren nicht nur den Austausch von Daten, sondern auch die eindeutige Adressierung einzelner Endpunkte und ermöglichen so einen globalen Datenaustausch. Auf den höheren Schichten finden Protokolle wie *HTTP* [Int99], *XMPP* [XMP12], *IMAP* [Int03] und *SOAP* [W3C07] - um nur wenige prominente Protokolle zu nennen.

Bei Berücksichtigung mobiler Teilnehmer in der Rolle eines Produzenten oder Konsumenten ist der Aspekt der Datenübermittlung ungleich herausfordernder als in klassischen verteilten Systemen, da Kommunikationsverbindungen störanfällig sind und sich die Adressen der Endpunkte jederzeit und

unvorhersehbar ändern können [Rot05]. Hierfür bedarf es seitens eines Vermittlers einer entsprechenden Mobilitätsverwaltung (engl. *Mobility Management*). Deren Aufgabe ist die Vergabe eindeutiger Kennzeichner für alle (mobilen) Geräte, damit eine logische Adressierung erfolgen kann, ggf. das Zwischenspeichern von Nachrichten, sofern ein Gerät temporär nicht erreichbar ist, sowie die Buchführung über aktuelle Kommunikationskanäle eines Gerätes, um mit diesem bei Bedarf über entsprechende Techniken und Protokolle kommunizieren zu können. Bei Produzenten und Konsumenten reduzieren sich die Aufgaben der Mobilitätsverwaltung auf eine einfache Nachrichtenwarteschlange (engl. *Message Queue*), in welche unzustellbare Nachrichten eingereiht werden, falls sich temporär kein Vermittler in Kommunikationsreichweite befindet.

Darüber hinaus unterscheidet man bei der Kommunikation entsprechend dem Initiator des Aktes zwischen zwei Verfahren: *pull-basierte* und *push-basierte* Kommunikation [DGD05, TS07]. Bei der push-basierten Kommunikation werden Nachrichten proaktiv von der Quelle an den intendierten Empfänger übermittelt. Zum Beispiel beruht der Versand einer eMail an einen Mail-Server auf dieser Kommunikationsart. Demgegenüber steht die pull-basierte Kommunikation, bei welcher der Empfänger (periodisch) bei der Quelle Nachrichten abholt. Diese Form der Kommunikation findet sich zum Beispiel bei dem Empfang von eMails, wobei der Empfänger regelmäßig sein Postfach auf neue Nachrichten überprüft, aber insbesondere auch bei Systemen mit mobilen Teilnehmern ohne eine Mobilitätsverwaltung. Das System muss hierbei keine Kenntnis darüber haben, unter welcher Adresse oder mittels welchen Protokolls ein mobiler Teilnehmer bei Ankunft einer zu übermittelnden Nachricht erreichbar ist. Stattdessen speichert das System die Nachricht und stellt diese erst bei Abfrage seitens des Empfängers im Rahmen desselben Kommunikationsaktes zu. Häufig ist die push-basierte Kommunikation gegenüber der pull-basierten vorzuziehen, da sie weniger Nachrichtenaufkommen (durch Vermeidung periodischen Nachfragens) verursacht, eine zeitnahe Zustellung von Nachrichten ermöglicht und kein Zwischenspeichern von Nachrichten erfordert. Im Gegensatz dazu ist der Vorteil der pull-basierten Kommunikation, dass der Konsument selbst über den Zeitpunkt der Abholung bestimmen kann und dass die bidirektionale Kommunikation netzwerkübergreifend auch in sogenannten *privaten Netzen*[1] funktioniert, z.B. die Nachrichtenzustellung aus dem Internet auf ein mobiles Telefon.

4.4.3. Datenverarbeitung

Das übergeordnete Ziel im Vermittlungsprozess ist es, den Konsumenten genau die Kontextdaten in geeigneter Repräsentation zur Verfügung zu stellen,

[1] Private Netze benutzen einen eigenen, nicht-öffentlichen Adressraum für Endpunktadressen. Am Rand des Netzes, in IP-basierten Netzen z.B. durch einen *NAT-Router*, werden private in öffentliche Adressen übersetzt. Mobiltelefone beispielsweise befinden sich in privaten Netzen der Mobilfunkanbieter und sind daher aufgrund ihrer nicht-öffentlichen Endpunktadresse nicht ohne Weiteres aus IP-basierten öffentlichen Netzen erreichbar [VHP06, CDK12].

die sie zur Erbringung ihrer eigentlichen Funktion benötigen. Hierbei ist insbesondere bei Berücksichtigung ressourcenschwacher Endgeräte in der Rolle des Produzenten oder Konsumenten die Minimierung der Verarbeitungslast für diese anzustreben, damit die Ressourcen der Geräte nicht zu stark beansprucht werden. Hinzu kommt, dass viele dieser Geräte keine komplexen Verarbeitungsschritte ausführen können, da z.B. keine Fließkommaoperationen unterstützt werden und somit eine Vielzahl gängiger Algorithmen auf diesen Geräten nicht effektiv oder effizient ausführbar sind [SMZ07]. Um im Rahmen des Vermittlungsprozesses die Produzenten und Konsumenten zu entlasten, den Aufwand der Anwendungsentwicklung zu minimieren, Wiederverwendbarkeit von Funktionen zu unterstützen und komplexe kontextsensitive Anwendungen zu ermöglichen, bietet es sich daher an, gängige Verarbeitungsschritte durch einen Vermittler ausführen zu lassen.

Auf welche Weise bzw. wozu Sensordaten auf Seite des Konsumenten genutzt werden, ist immer vom jeweiligen Anwendungsfall abhängig. In der derzeitigen Praxis sind jedoch Produzenten und Konsumenten softwareseitig eng gekoppelt, das bedeutet, dass sich mindestens eine Seite bezüglich der Repräsentation, Interpretation und Verarbeitung von Kontextdaten sowie der Art der Kommunikation an die jeweils andere Seite anpassen muss [BS08, DPPR06]. Beispielsweise ist es einer ortsbasierten Anwendung häufig nicht aufwandsarm möglich, den Produzenten (also die Komponente zur Positionsbestimmung) zu wechseln, da Positionen in vielen Fällen unterschiedlich repräsentiert werden (z.B. in Mikrograd bei Android-Telefonen, im NMEA 0183-Format bei reinen GPS-Empfängern oder einfach als postalische Adresse) und entsprechend die Programmlogik zur Verarbeitung der Positionsdaten beim Wechsel angepasst werden müsste.

Durch diese Abhängigkeit ist es einerseits den Konsumenten nicht möglich, Kontextdaten zur Laufzeit von einem anderen Produzenten zu beziehen, und andererseits können Produzenten ihre Daten nicht an beliebige weitere Konsumenten ausliefern, sodass eine domänenübergreifende Wiederverwendung der Kontextdaten nur schwerlich realisierbar ist. Der wesentliche Grund hierfür liegt in einer fehlenden Übereinkunft bezüglich der Repräsentation, Interpretation und Kommunikation der Daten, was wiederum u.a. auf fehlende Standards zurückzuführen ist [BL09]. Es bedarf daher, genau wie bei der Datenübermittlung, einer vermittelnden Instanz, welche die Daten der einen Seite entgegennimmt und für die jeweils andere Seite gemäß deren Anforderungen aufbereitet.

Hierbei sollen die Daten idealerweise so weitgehend verarbeitet werden, dass die Konsumenten diese direkt weiterverwenden können. Nun gibt es jedoch möglicherweise gänzlich unterschiedliche Anforderungen an die Daten. Der eine Konsument bevorzugt Positionsdaten im NMEA 0183-Format, ein anderer Längen- und Breitengrade in Mikrograd. Dies impliziert, dass ein Vermittler die Daten für jeden Konsumenten individuell nach dessen Maßgabe aufbereiten und bereitstellen muss. Auf der anderen Seite kann es Verarbeitungsschritte geben (z.B. Entschlüsselung, Dekompression), die ein Vermitt-

ler zwangsläufig auf allen eintreffenden Daten eines Produzenten ausführen muss. Dies führt dazu, dass ein Vermittler auch nach Maßgabe des Produzenten Verarbeitungsschritte auf Sensordaten auszuführen hat, bevor er schließlich die Verarbeitungsschritte der Konsumenten bearbeitet.

Es existieren eine Reihe gängiger Verarbeitungsschritte, die im Umgang mit Kontextdaten domänenübergreifend Verwendung finden. In Abschnitt 2.3 wurden bereits mit der Aggregation, Reduktion, Konversion, Archivierung, Augmentation und Inferenz die wesentlichen Verarbeitungsklassen vorgestellt. Weitere gängige Verarbeitungsschritte im Vermittlungsprozess sind z.B. das Sortieren von Sensordaten, das Erstellen von Kontextmodellen und das Tasking von Produzenten. Für die meisten Verarbeitungsfunktionen existieren entsprechende Programmcode-Bibliotheken, welche von einem Vermittler in Form einer Aufgabenbibliothek (engl. *Task Library*) für die Produzenten und Konsumenten bereitgestellt werden können, um diesen ein einfaches Auffinden und Nutzen der Verarbeitungsfunktionen zu erlauben. Auch sollte die Bibliothek erweiterbar sein, sodass Klienten eigene Verarbeitungsfunktionen in Form von Programmcode oder Referenzen auf externe Dienste hinzufügen können.

Funktionen (z.B. Lernalgorithmen) können zustandsbehaftet sein, d.h. dass für einen bestimmten Produzenten oder Konsumenten der Zustand einer spezifischen Funktionsinstanz bei eintreffenden Anfragen wiederhergestellt und fortgeschrieben werden muss, was bei einer verteilten Ausführung des Vermittlungsprozesses besondere Herausforderungen birgt. Für den Umgang mit solchen Funktionen ist von einem Vermittler eine entsprechende Aufgabenverwaltung (engl. *Task Management*) vorzusehen, welche die Ausführungsanforderungen und den Lebenszyklus zustandsbehafteter Funktionen koordiniert.

Auch beschränkt sich die Datenverarbeitung häufig nicht auf die Ausführung lediglich einer Verarbeitungsfunktion. Aufgrund der starken Kohäsion solcher Funktionen bedarf es einer Möglichkeit seitens der Klienten eine Ausführungssequenz (auch als *Prozess* oder *Workflow* bezeichnet) zu formulieren. Die Orchestrierung solcher Sequenzen zur Laufzeit muss durch einen Vermittler über eine entsprechende Workflow-Verwaltung (engl. *Workflow Management*) realisiert werden.

4.4.4. Datenverwaltung

Im Rahmen der Datenverwaltung geht es um den Umgang mit Daten, die in einer Form von Datenbanksystem, Verzeichnis, Tabelle oder dergleichen persistiert sind. Hierzu gehören die Organisation der Daten auf physischer Ebene, das Bereitstellen logischer Abstraktionen, der Zugriffsschutz sowie die Suche und letztendlich die Abfrage der Daten[2].

[2]Zusätzlich können auch die Analyse, die Visualisierung und das Data Mining als Teil der Datenverwaltung angesehen werden [BDF+07]. Im Rahmen dieser Arbeit werden sie lediglich zur Unterstützung der Abfrageformulierung in Abschnitt 4.4.7 behandelt.

Die Herausforderungen an die Datenverwaltung im Vermittlungsprozess sind mannigfaltig. Gründe hierfür sind im Wesentlichen die massive Menge an Daten, deren Heterogenität (z.B. Inhalt, Semantik, Struktur, Format etc.), die Art der Nutzung (lesender vs. schreibender und privater vs. öffentlicher Zugriff sowie Transaktionserfordernisse, ggf. zeitkritischer Zugriff etc.) sowie die Verteilung der Daten über mehrere Systeme [BDF+07].

Auch muss die Datenverwaltung ganz unterschiedliche Inhalte handhaben [JC09]. Auf der einen Seite finden sich Kontextdaten, die kontinuierlich im System eintreffen. Hierzu gehören beispielsweise RFID-Leseereignisse oder mehrdimensionale Sensormesswerte, aber auch Zustandsinformationen über die aktuelle Prozessausführung (z.B. Logging- oder Abrechnungsdaten). Ein Teil der Kontextdaten kann einen geographischen Bezugspunkt enthalten, was insbesondere relevant ist, sofern die Daten persistiert und indiziert werden sollen. Auf der anderen Seite stehen große Mengen relativ statischer Daten, insbesondere Metadaten von zum Beispiel Messwerten, Produzenten und Konsumenten sowie Verarbeitungsdiensten etc. [JC09].

Weitere Herausforderungen sind beispielsweise ein geeigneter Zugriffsschutz, da diese Daten teils sensible Informationen darstellen können, praxisrelevante (verteilte) Transaktionskontrollmechanismen sowie letztlich auch eine Möglichkeit sowohl kontinuierlich eintreffende, transiente Daten als auch persistierte Daten aufzufinden und abzufragen (siehe auch Abschnitt 4.4.7) [JC09].

Die Nutzung existierender Datenbanksysteme stellt zwar einen guten Kompromiss zwischen den verschiedenen Anforderungen an die Datenverwaltung dar, jedoch können nicht alle Anforderungen gleichzeitig optimal erfüllt werden. Insbesondere der Umgang mit massiven Mengen sogenannter *Live-Daten*, also der kontinuierlich eintreffenden Datenströme, disqualifiziert ein einzelnes System für die alleinige Verwendung im Vermittlungsprozess [BDF+07].

In der Literatur finden sich eine Reihe unterschiedlichster Arten von Datenbanksystemen für den Umgang mit Kontextdaten, die in der Praxis Verwendung finden oder deren Verwendung vorgeschlagen wird. Hierzu gehören beispielsweise die klassischen relationalen Datenbanksysteme, räumlich-zeitliche Datenbanken, probabilistische Datenbanken, schemafreie, nicht-relationale Datenbanken, Sensordatenbanken etc. Im Folgenden sollen für eine Reihe der relevantesten Anforderungen entsprechende Datenbankmodelle vorgestellt werden.

Relationale Datenbanken Relationale Datenbanken sind heutzutage weit verbreitet. Die Gründe hierfür sind mannigfaltig. Sie beruhen auf dem leicht verständlichen relationalen Datenbankmodell, welches Relationen (Tabellen), Datenobjekte/Tupel (Zeilen in den Tabellen) und Attribute (Spalten in den Tabellen) vorsieht. Mittels der Attribute lassen sich außerdem Beziehungen zwischen einzelnen Datenobjekten unterschiedlicher Relationen herstellen. Operationen auf den Relationen sind durch die relationale Algebra formalisiert

und werden in der Praxis meist mittels der *Structured Query Language* (SQL) beschrieben [KE11].

Aufgrund ihrer weiten Verbreitung ist auch die Entwicklung der relationalen Datenbanken sehr weit fortgeschritten. Existierende Systeme sind robust und leistungsfähig, bieten fortgeschrittene Konsistenz- und Transaktionsmodelle, feingranulare Zugriffskontrollen, eine hohe Verfügbarkeit und Ausfalltoleranz etc. [KE11]. Zudem gibt es eine Reihe ausgereifter Werkzeuge, welche Anwender bei der Einrichtung, Pflege, Datenanalyse und Abfragestellung etc. unterstützen.

Prinzipiell lassen sich nahezu alle Arten von Daten in relationalen Datenbanken persistieren und beliebige Abfragen auf diesen Daten ausführen. Für einige Anwendungsfälle eignen sich relationale Datenbanken jedoch nur bedingt, da teils erheblicher Aufwand bei der Abfragestellung betrieben und mit Leistungseinbußen bei der Abfrageverarbeitung gerechnet werden muss. Dies gilt insbesondere für große Datenbestände [EBF+10, BDF+07]. Auch ist es unrealistisch, eine uniforme Struktur bzw. ein einheitliches Schema für derlei große Datenmengen, die heterogenen Quellen entstammen, zu erwarten [JC09].

Trotzdem können relationale Datenbanken zum Persistieren von Sensor- und Metadaten verwendet werden. Für eine Untermenge der Metadaten bietet sich deren Verwendung sogar an, da sich bestimmte Metadaten gut auf das relationale Datenmodell abbilden lassen. Konsistenz- und Transaktionskontrollen stellen zudem die Korrektheit der Verarbeitung dieser Daten sicher, Zugriffsschutzmechanismen beschränken den Zugang zu vertraulichen Informationen und Indizes erlauben den performanten Zugriff auch in großen Datenbeständen.

Die Eignung der Datenbanken zum Persistieren von Sensordaten wird hingegen kritisch gesehen, was im Wesentlichen der schieren Menge an Daten sowie den Zugriffsmustern geschuldet ist [PAF09, EBF+10, BDF+07]. Sensordaten gelangen als kontinuierliche Datenströme in die Datenbank. Hierdurch kommt es zu überproportional vielen Schreibzugriffen, die einen hohen Verwaltungsaufwand innerhalb der Datenbank nach sich ziehen. Demgegenüber stehen verhältnismäßig wenige lesende Zugriffe. Auch skalieren relationale Datenbanken nur bedingt gut. Insbesondere die horizontale Skalierbarkeit gehört nicht zu den Kernkompetenzen der relationalen Datenbanksysteme [EBF+10]. Weitere Einschränkungen ergeben sich aus der Natur der Sensordaten: Daten heterogener Quellen lassen sich nur umständlich in strikte, einheitliche Datenschemata abbilden [JC09]. Auch für die inhärente Ungenauigkeit bzw. Unsicherheit der Daten bzgl. ihrer Korrektheit bieten die klassischen Systeme keine Unterstützung [JC09]. Unter anderem aus diesen Gründen wurden auch alternative Datenbankmodelle entwickelt, welche im weiteren Verlauf dieses Abschnitts vorgestellt werden.

Probabilistische Datenbanken Traditionelle Datenbanken handhaben deterministische Daten, das bedeutet, die Daten werden als fehlerfrei und kor-

rekt angesehen [DS07]. Diese Annahme ist unabhängig davon, ob die Daten ursprünglich wirklich fehlerfrei und korrekt waren, sie erleichtert jedoch den Umgang mit diesen Daten. In vielen Anwendungsfällen, z.B. im Bankwesen oder der Lagerhaltung, sollte man auch von der Korrektheit ausgehen dürfen. Kontextdaten, die durch physische Sensoren erhoben werden, besitzen diese Eigenschaft nicht unbedingt, sie werden als inhärent unsicher angesehen. Die Unsicherheit basiert auf den Eigenschaften der Sensoren, welche Hardware-bedingt immer ein gewisses Rauschen aufweisen und deren Messwerte durch störende Einflüsse der Umwelt zusätzlich beeinträchtigt werden können [CM00, HC09]. Wenn also ein Temperatursensor eine Temperatur von 20,5 °C misst, muss dies nicht unbedingt der Realität entsprechen. Mit einer gewissen Wahrscheinlichkeit liegt der reale Wert höher oder niedriger. In traditionellen Datenbanken wird diese Unsicherheit jedoch nicht reflektiert.

Für den Umgang mit unsicheren Daten wurden sogenannte *Probabilistische Datenbanken* entworfen. Diese basieren auf dem Prinzip der *möglichen Welten* [SOK11]. Anstatt alle enthaltenen Daten als fehlerfrei und korrekt anzusehen, d.h. nur eine Welt zu repräsentieren, werden die Daten mit Wahrscheinlichkeiten versehen, sodass zu einer Abfrage durchaus mehrere Ergebnisse geliefert werden können, das jedes für sich eine mögliche Welt repräsentiert. Auf diese Weise wird der Unsicherheit der Daten Rechnung getragen und die Interpretation bzw. der Umgang mit dieser Unsicherheit weiter in Richtung Anwendung bzw. Benutzer verschoben.

Probabilistische Datenbanken stellen also eine Möglichkeit dar, insbesondere prinzipbedingt unsichere Sensordaten unter Beibehaltung dieser Information zu persistieren und abzufragen [LLRS97, DS07, BDF+07]. Es gibt bereits eine Reihe von Projekten, die Implementierungen hervorgebracht haben, entweder als Erweiterung einer existieren relationalen Datenbank oder als eigenständiges System, und als Einsatzkontext insbesondere auch die Verwaltung von Sensordaten erwähnen. Hierzu gehören beispielsweise *PrDB* [PrD12], *MayBMS* [May12] und der *BayesStore* [Bay12]. Eine Übersicht weiterer Projekte findet sich unter [Wik12c].

Räumlich-zeitliche Datenbanken Wie bereits erwähnt, eignen sich relationale Datenbanken nur bedingt für die Speicherung von Sensordaten, insbesondere solcher, die einen Ortsbezug aufweisen. Für eine Reihe von Anwendungsfällen ist es jedoch auch gar nicht notwendig ganze Zeitreihen von Messwerten zu persistieren. Stattdessen ist man vor allem am aktuellen (oder zukünftigen) Ort eines Objektes interessiert: „Welche Taxis werden in den nächsten 10 Minuten am Hauptbahnhof vorbeikommen ? “ [WXCJ98]. Die Objekte (z.B. Taxis) übermitteln in bestimmten Zeitintervallen ihren aktuellen Aufenthaltsort, der dann persistiert und abgefragt werden kann.

Relationale Datenbanksysteme und Abfragesprachen können derartige Anwendungsfälle nur bedingt unterstützen. Im einfachsten Fall würde man Ortsangaben als Attribute der Datenobjekte modellieren und diese stetig aktualisieren. Bei Systemen mit sehr vielen sich bewegenden Objekten würde dies

allerdings große Leistungsprobleme mit sich bringen, da nicht nur die Daten, sondern auch die Indizes der Datenbanken kontinuierlich angepasst werden müssten. Außerdem können sie nicht mit der inhärenten Unsicherheit bzw. Ungenauigkeit der Daten umgehen und es besteht die Schwierigkeit in der Definition und Verarbeitung von Abfragen mit räumlichen (der Hauptbahnhof z.B. könnte durch ein Polygon beschrieben sein) und temporalen Aspekten („in den nächsten 10 Minuten“) sowie allgemein kontinuierlichen Abfragen, die an Datenströme gestellt werden. Abfragesprachen wie SQL sehen hierfür keine Operatoren vor [LWZ04, FJK+05] und speziell entwickelte *räumliche Datenbanken*, *temporale Datenbanken* oder *probabilistische Datenbanken* decken jeweils nur einen dieser Aspekte ab.

Einer der ersten Ansätze, der diese Problematiken adressierte, wurde mit *Databases fOr MovINg Objects* (DOMINO) in [WXCJ98] vorgestellt. In diesem System wurden dynamische Attribute verwendet, die je nach Abfragezeitpunkt unterschiedliche Voraussagen für den aktuell realen Aufenthaltsort liefern. Eine eigene Abfragesprache führt räumliche (z.B. Distanz, Enthaltensein) und zeitliche Operatoren (z.B. Schließlich, Immer, Bis) ein. Zusammen mit speziellen Indizierungsmechanismen und der Handhabung von Datenungenauigkeiten offeriert dieses System eine geeignete Lösung für zuvor beschriebene Herausforderungen. Eine Übersicht weiterer Systeme für das Persistieren und zur Abfrage räumlich-zeitlicher Daten findet sich in [YSMMW11].

NoSQL-Datenbanken Die weit verbreiteten relationalen Datenbanken stoßen im Umgang mit großen Datenmengen aus heterogenen Quellen und sich häufig ändernden Anforderungen einer agilen Welt auf eine Reihe von Problemen. Hierzu gehört beispielsweise die Erfordernis eines strikten, relationalen Datenbankschemas. Das Schema gibt vor, wie die Daten in einer Datenbank logisch strukturiert werden und zusammenhängen [KE11]. Es gibt eine Reihe von Datenmodellen, z.B. Graphenmodelle, die nicht einfach bzw. effizient in einem relationalen Modell repräsentiert werden können, sodass sich andere Ansätze ggf. besser eignen [EBF+10]. Häufig ändern sich auch im Laufe der Zeit die Anforderungen, sodass Anpassungen an einem relationalen Schema, z.B. das Einfügen neuer Attribute oder Beziehungen, unumgänglich werden, wodurch Datenbanken, die große Datenmengen halten, stundenlang außer Betrieb gesetzt werden [EBF+10].

Im Umgang mit sehr großen Datenmengen im Terabyte- oder Petabyte-Bereich stoßen viele traditionelle Datenbanksysteme zudem auf horizontale und vertikale Skalierbarkeitsprobleme hinsichtlich entweder des effizienten Zugriffs oder der einfachen Umsetzung von Replikationsmechanismen. Auch die strikten Konsistenz- und Transaktionserfordernisse, die bei traditionellen Datenbanksystemen auf dem *ACID-Modell* [KE11] beruhen, erfordern dahingehend viele Verarbeitungsressourcen, dass diese für bestimmte Anwendungszwecke aufgeweicht werden können [EBF+10].

Aus diesen Gründen bedurfte es mit Hereinbrechen der Ära des Web 2.0 neuer Ansätze, welche besser auf die sich ergebenden Herausforderungen großer

Datenmengen abgestimmt waren [EBF+10]. Unter dem Begriff *NoSQL*[3] versteht man alternative Ansätze, welche zumindest einen Teil, der oben skizzierten Probleme relationaler Datenbanksysteme, aufgreifen. So besitzen NoSQL-Datenbanken kein relationales Datenmodell und haben keine oder schwache Schemarestriktionen, skalieren gut, lassen sich effizient und einfach verteilen und besitzen häufig ein weniger striktes Konsistenzmodell [EBF+10].

Aufgrund der Vielfalt existierender NoSQL-Systeme hat sich eine Unterteilung in verschiedene Gruppen etabliert [EBF+10]: *Key / Value / Tuple Stores*, *Column Stores*, *Document Stores* und *Graphendatenbanken*. Aber auch zum Beispiel objekt- oder XML-basierte Datenbanken kann man streng genommen den NoSQL-Datenbanken zuordnen, da ihnen kein relationales Datenmodell zugrunde liegt. Jede dieser Klassen bietet individuelle Vor- und Nachteile, je nach avisiertem Anwendungszweck.

In [EBF+10] findet sich eine Übersicht über eine Reihe von typischen Anwendungsfällen mit ihren jeweiligen Besonderheiten und entsprechend geeigneten NoSQL-Systemklassen bzw. konkreten Datenbanksystemen. Insbesondere heben die Autoren hervor, dass sich NoSQL-Datenbanken aus verschiedenen Gründen für u.a. Geo-, Social Web- und Sensordaten eignen, da hier Faktoren wie zum Beispiel Skalierbarkeit, Leistung und Flexibilität besonders zum Tragen kommen.

Sensordatenbanken Alle bisher vorgestellten Ansätze fallen unter die sogenannten Warenhausansätze (engl. *Warehousing Approaches*) [BGS01]. In derartigen Ansätzen werden Sensoren bzw. Sensornetzwerke lediglich zum Erheben von Daten verwendet, die weitere Verarbeitung und das Persistieren erfolgt schließlich in der Infrastruktur [BGS01]. Im Gegensatz dazu ist die Grundidee von Sensordatenbanken die verteilte Speicherung von Sensordaten auf den Sensorknoten selbst und die Speicherung von Metadaten an einer zentralen Abfrageschnittstelle (auch als Basisstation bezeichnet [MFHH05]) [HC09]. Sobald eine Abfrage an dieser Schnittstelle eintrifft, werden anhand der Metadaten geeignete Sensoren zur Beantwortung ausgewählt und die Abfrage bzw. Teile der Abfrage an diese übermittelt. Die Sensoren ihrerseits versuchen diese mittels der lokal vorhandenen Daten zu beantworten und für eine letztliche Aggregation zurückschicken. Auf diese Weise werden nur Daten, die für die Beantwortung einer Abfrage relevant sind, kommuniziert und somit Ressourcen geschont. Der Umgang speziell mit räumlich-zeitlichen oder unsicheren Daten wurde in dieser Forschungsrichtung noch nicht adressiert.

COUGAR [BGS01] sowie *TinyDB* [MFHH05] und auf einem etwas höheren Abstraktionsniveau und für größere Skalen geeignet auch *SStreaMWare* (siehe Abschnitt 6.1.8) repräsentieren Lösungen für derartige Sensordatenbanksysteme. Detailliertere Beschreibungen weiterer Sensordatenbanksysteme finden sich in [BAAIA09, HC09].

[3] NoSQL hat sich allgemein als Akronym für *Not only SQL* durchgesetzt [EBF+10].

Implikation für die Datenverwaltung

Sensoren produzieren kontinuierlich Messwerte, welche im Rahmen eines Vermittlungsprozesses verwaltet werden müssen. Neben den ohnehin großen Mengen an Metadaten stehen Systeme zur Datenverwaltung so vor gänzlich neuen Herausforderungen, was einerseits funktionale Anforderungen wie die Persistierung und die Abfrageverarbeitung, aber vor allem auch nichtfunktionale Anforderungen wie Skalierbarkeit, Zugriffseffizienz, Flexibilität, Robustheit etc. betrifft. Hinzu kommen die inhärente Unsicherheit der Daten, mit der die Datenverwaltung umgehen können sollte, die besondere Sensibilität der Daten, welche entsprechende Zugriffsschutzmaßnahmen erfordert sowie die Tatsache, dass ein Großteil der Sensordaten Georeferenzen trägt, welche besondere Anforderungen an die Indexierung und Abfrageverarbeitung stellt.

Da kein Datenbankmodell alle Herausforderungen meistern kann, wurden in diesem Abschnitt verschiedene Modelle mit ihren Besonderheiten auch hinsichtlich der Speicherung von Meta- und Sensordaten vorgestellt. Um optimal auf die Bedürfnisse der Datenverwaltung im Vermittlungsprozess eingestellt zu sein, bedarf es daher zwangsläufig einer Kombination verschiedener Modelle und Systeme und damit einhergehend einer entsprechenden Unterstützung, die sich durch nahezu alle Aufgabenbereiche des Vermittlungsprozesses zieht.

4.4.5. Prozessverwaltung

Im Rahmen des Vermittlungsprozesses muss das Zusammenspiel verschiedener Dienste und Ressourcen koordiniert und (ggf. seitens eines Administrators) überwacht werden. Diese Aufgaben obliegen der *Prozessverwaltung*, welche somit eine zentrale Rolle im Vermittlungsprozess einnimmt. Alle Anfragen und Dienstaufrufe werden hier protokolliert, bspw. um i) Zugriffsrechte auf Dienste und Daten zu kontrollieren, ii) Abrechnungen für Dienst- und Datennutzung zu erstellen, iii) Dienstgütevereinbarungen sicherzustellen und ggf. weitere Ressourcen zu allozieren und iv) Informationen über mögliche Fehler festzuhalten, die bei einer späteren Fehleranalyse hilfreich sein könnten, etc.

Kontextdaten stellen sehr sensible Informationen dar und unter Umständen möchte ein Produzent den Zugriff auf diese Daten daher auf einen bestimmten Konsumentenkreis eingrenzen. Dasselbe gilt auch für Verarbeitungsdienste, die nicht von jedem verwendet werden dürfen, z.B. weil sie intern Zugriff auf vertrauliche Informationen benötigen. Als Teil der Prozessverwaltung stellt die Zugriffskontrolle (engl. *Access Control*) sicher, dass nur autorisierten Instanzen der Zugriff gewährt wird. Es gibt verschiedene Möglichkeiten eine solche Zugriffskontrolle zu realisieren, in der Praxis basieren die Verfahren meist auf der Übermittlung eines Berechtigungsnachweises (engl. *Credential*) mit Hilfe dessen die Zugriffserlaubnis anhand von Zugriffskontrolllisten (engl. *Access Control Lists*) überprüft werden. Dies können z.B. Name und Passwort, eine Zugriffsmarke, biometrische Daten (z.B. ein Portraitfoto) oder auch bestimmte Kontextdaten (z.B. der aktuelle Aufenthaltsort) sein [ED06].

Abgesehen von der Einschränkung eines Zugriffs auf vertrauliche Dienste und Daten, kann auch ein Zugriffsprotokoll erstellt werden, welches u.a. auch im Rahmen des *Logging & Accounting* [ED06, SHG+08] zu Abrechnungszwecken dient. So ist es beispielsweise möglich, dass ein Produzent für den Zugriff auf seine Sensordaten eine Gebühr von den Konsumenten erhebt. Gleiches gilt für die Inanspruchnahme bestimmter Verarbeitungsdienste. Hierdurch kann eine kostenfokussierte Infrastruktur zu einer leistungsfokussierten Infrastruktur transformiert werden und so zu schnellerer Akzeptanz der Kontextdatenvermittlung in unternehmensübergreifenden Anwendungen führen [UHM11]. Eine Protokollierung der Zugriffe erlaubt somit das Erstellen von Rechnungen bzw. eine Verzögerung des Zugriffs bis eine entsprechende Gebühr verbucht wurde.

Das Zugriffsprotokoll kann weitergehend zum Testen und zur Fehleranalyse (engl. *Testing & Debugging*) verwendet werden. Da der Vermittlungsprozess eine Reihe von Verarbeitungsschritten enthalten kann, die verteilt und auch parallel ausgeführt werden können, ist es bei Auftreten eines Fehlers oftmals schwierig, diesen genau zu lokalisieren. Die Protokollierung der Geschehnisse im Rahmen des Vermittlungsprozesses kann hierbei unterstützen, indem sie die Aufrufreihenfolge von Diensten, ggf. mit ihren Aufrufparametern, anzeigt und so eine Wiederholung für Tests ermöglicht.

Der letzte Kernbereich der Prozessverwaltung betrifft die Dienstgüteüberwachung (engl. *Quality of Service Monitoring*) sowie damit einhergehend auch die Ressourcenverwaltung (engl. *Resource Management*). Abhängig von der Menge an eintreffenden Daten und aktiven Vermittlungsprozessen, kann ein Vermittler entscheiden, weitere Ressourcen zu allozieren oder vorhandene Ressourcen umzuwidmen, um bestimmten Leistungskriterien einer Dienstgütevereinbarung gerecht zu werden. Erhöht sich beispielsweise das Aufkommen an Sensordaten, so kann ein Vermittler zusätzliche Knoten in den Vermittlungsprozess einbinden, damit die Datenverwaltung über mehrere Knoten hinweg skalieren kann (engl. *Scale Out*). Sinkt die Zahl der aktiven Vermittlungsprozesse, können z.B. nicht weiter benötigte Knoten aus dem System entfernt und somit ggf. Kosten gespart werden. Auch könnte die Ausführung einzelner Dienste auf bestimmte Knoten ausgelagert werden, um im System einen Lastausgleich zwischen den Knoten herzustellen.

4.4.6. Metadatenverwaltung

Metadaten repräsentieren im Allgemeinen strukturierte Daten über andere Daten, Informationen, Objekte etc. [SL09]. Im Kontext des Vermittlungsprozesses repräsentieren sie also Daten über u.a. Messwerte, Dienste, Produzenten und Konsumenten. Zu den Metadaten der Produzenten gehören z.B. die Kennung des Sensors oder des Datenstroms, der Standort, Batteriezustand etc. Messwertmetadaten sind beispielsweise die Art des Messwertes (z.B. Temperatur), Zeitpunkt und Ort der Erhebung, die Maßeinheit oder das Format.

Daten über Konsumenten sind z.B. aktuell subskribierte Abfragen, Protokoll der Dienstzugriffe zu Abrechnungszwecken sowie die aktuelle Kommunikationsendpunktadresse etc. Und zu den Dienstmetadaten gehören die Funktionsbeschreibung (z.B. Maßeinheitenkonvertierung, Nachrichtenversand), Ein- und Ausgabeparameter, etwaige Zugriffsbeschränkungen, Kosten etc. [JBS09].

Dieser Abschnitt soll den Nutzen von Metadaten insbesondere für den Konsumenten bzw. die Abfrageverarbeitung aufzeigen. Metadaten über den Verlauf des Vermittlungsprozesses, zum Beispiel zu Debugging oder Abrechnungszwecken sollen nicht weiter betrachtet werden. Im Folgenden wird daher anhand eines Beispiels zunächst die Problematik der Abfrageformulierung skizziert und dadurch das Bereitstellen von Metadaten für den Konsumenten motiviert. Das Bereitstellen erfolgt zwar in erster Linie durch die Produzenten selbst, allerdings übernimmt ein Vermittler die Verwaltung und Auslieferung dieser relativ statischen Daten, um den Produzenten von sich wiederholenden Abfragen seitens der Konsumenten zu entlasten. Während die Herausforderungen des Metadatenpersistierens in Abschnitt 4.4.4 zur allgemeinen Datenverwaltung eingehender erläutert wurden, wird also am Ende dieses Abschnittes die Frage, wie Vermittler und Produzenten Metadaten austauschen bzw. wie die Daten der Datenverwaltung zugeführt werden, näher betrachtet.

Metadaten zur Unterstützung der Abfrageformulierung

Im Rahmen des Vermittlungsprozesses formuliert ein Konsument eine Abfrage nach Kontextdaten und kann optional bestimmte Verarbeitungsvorschriften für ein Ergebnis angeben (siehe Abschnitt 4.4.7). Hierfür ist es oft hilfreich, wenn der Konsument unter anderem Kenntnis darüber hat, welche Arten von Messwerten überhaupt verfügbar sind, von welchen Produzenten diese stammen und welche Dienste zur Verarbeitung des Ergebnisses bereitstehen [JBS09]. Das Bereitstellen solcher Daten erleichtert die Abfragestellung [MS06] und hilft bei der korrekten Angabe von Abfragekriterien. Ein Konsument könnte z.B. folgende Abfrage stellen wollen: „Wie hoch war die Durchschnittstemperatur in Hamburg in der letzten Woche“. Da natürlichsprachlich formulierte Abfragen schwer zu verarbeiten sind, muss die Abfrage in einer entsprechenden Abfragesprache (siehe Abschnitt 4.4.7) formuliert werden, z.B. in SQL als:

```
SELECT avg(dwd.TemperaturInC)
FROM DeutscherWetterDienst dwd
WHERE dwd.Stadt = 'HH' und dwd.Zeitstempel > 2012-05-23
```

Zur korrekten Formulierung einer Abfrage muss der Konsument also einen Bezeichner für den Produzenten ($DeutscherWetterDienst$), die genaue Themenbezeichnung ($TemperaturInC$), die Bezeichner für die Filterkriterien ($Stadt$ und $Zeitstempel$) sowie mögliche Belegungen der Filterkriterien (HH und $2012-05-23$) kennen, ansonsten kann die Abfrage nicht korrekt beantwortet werden. Möchte man die Ergebnisse zudem noch in einem bestimmten

Format präsentiert bekommen, Maßeinheiten umrechnen oder sonstige Verarbeitungsschritte ausführen lassen, so muss man einem Vermittler, ebenfalls unter Verwendung einer bestimmten Syntax, mitteilen, dass dieser entsprechende Dienste aufrufen soll. Hierfür sind Angaben zu z.B. Funktionsnamen, Eingabeparametern, ggf. Quality of Service-Parametern etc. zu machen.

Um die Abfrageformulierung zu unterstützen, bedarf es also entsprechender Metadaten. Bereits erwähnt wurde die Unterteilung in Produzenten-, Konsumenten-, Messwert- und Dienstmetadaten. Während Produzenten- und Dienstmetadaten relativ statisch sind und daher in Verzeichnissen bzw. Katalogen persistiert werden, enthalten die Messwertmetadaten häufig dynamische Attribute (z.B. den Zeitstempel). Sie haften den Messwerten oft direkt an und sind daher, genau wie die Messwerte selbst, von flüchtiger Natur (siehe Abschnitt 4.4.7, ereignisbasierte Abfragen). Der Zugriff auf die Metadatenverwaltung seitens des Konsumenten erfolgt analog der Abfragestellung auf persistierte Messwerte (siehe Abschnitt 4.4.7, historische Abfragen) und wird idealerweise durch ein entsprechendes Werkzeug unterstützt.

Metadaten für semantische Abbildungen und Ableitungen Eine weitergehende Unterstützung der Abfragestellung, die auf Metadaten beruht, stellen semantische Abbildungen und Ableitungen dar [JBS09, BDF+07]. Mit ihrer Hilfe ist es möglich, Teile der Abfrage dahingehend zu abstrahieren, dass man technische oder formale Details, z.B. den konkreten Namen des Datenstroms oder der Filterattribute nicht kennen muss. Obige Beispiel-Abfrage ließe sich somit natürlicher formulieren:

```
SELECT avg(Temperatur)
FROM Umweltinformationen
WHERE Stadt = 'Hamburg' und Zeitstempel > (Heute - 1 Woche)
```

Bei der semantischen Abbildung wird zu einem bestehenden Begriff bzw. Konzept (z.B. $Temperatur$) semantisch gleiche oder ähnliche Begriffe bzw. Konzepte gesucht und für die Abfragestellung verwendet. Ein Produzent könnte in seinen Metadaten z.B. auszeichnen, dass er „TemperaturInC“ misst, ein anderer Produzent könnte stattdessen „Außentemperatur“ angeben. Durch die semantische Abbildung würde eine Abfrage nach „Temperatur“ beide Produzenten berücksichtigen. Auch Begriffe wie $Heute$ und $Woche$ können durch die Abbildung interpretiert und bei der Abfrageverarbeitung durch entsprechend formalisierte Zeitstempel ersetzt werden.

Bei der semantischen Ableitung werden zudem Generalisierungen und Spezialisierungen von Konzepten berücksichtigt, sodass eine Abfrage nach $Hamburg$ auch die Spezialisierungen „Fuhlsbüttel“ und Standortangaben in geografischen Längen-/Breitengraden berücksichtigt. Ebenso könnte auch das Postleitzahlengebiet „22527“, je nach Anwendungsfall, zu „Hamburg“ generalisiert werden.

Semantische Abbildungen und Ableitungen sind jedoch nur möglich, wenn Bezeichnern eine Bedeutung zugeschrieben wird (z.B. mittels semantischer

Annotationen und Ontologien). Anhand der Bedeutungen kann man dann von den Bezeichnern selbst abstrahieren. Wege in diese Richtung beschreiten zum Beispiel Jirka et al. durch Einführung der *Sensor Instance Registry* und der *Sensor Observable Registry* in das Sensor Web Enablement-Standardisierungsrahmenwerk [JBS09]. Auch Pschorr et al. präsentieren in [PHPS10] einen Ansatz um das SWE-Rahmenwerk semantisch gewahr zu machen. Sie stellen mit *SemSOS* eine semantisch erweiterte Implementierung des *Sensor Observation Service* von 52° North vor. Auch Moodley und Simonis zeigen in [MS06] den Bedarf semantischer Infrastrukturen für das Sensor Web auf und präsentieren eine auf Ontologien beruhende Wissensschicht für die Sensor Web Agent Platform (siehe Abschnitt 6.1.3).

Metadaten-Harvesting

Bisher wurde beschrieben, welchen Nutzen Metadaten für einen Konsumenten darstellen und wie Abstraktionen bei der Abfragestellung unterstützen können. Es bleibt jedoch die Frage offen, wie ein Vermittler die Metadaten ermittelt (engl. *Metadata Harvesting*) bzw. wie die Metadaten in die Verzeichnisse gelangen. Zwei prinzipielle Möglichkeiten lassen sich hier unterscheiden [JBS09]:

- die proaktive (oder auch push-basierte) Eingabe seitens des Produzenten bzw. eines Stellvertreters und
- die reaktive (oder auch pull-basierte) Variante, bei welcher ein Vermittler Metadatenabfragen im Netzwerk verbreitet und die Produzenten daraufhin antworten.

Bei der proaktiven Eingabe registriert sich der Produzent (oder sein Stellvertreter) initial bei einem Vermittler unter Angabe seiner Metadaten und aktualisiert diese auf gleichem Wege, sobald sich z.B. seine Position, sein Batteriezustand oder andere Eigenschaften ändern. Bei der pull-basierten Variante reagieren die Produzenten auf eine Aufforderung, die (periodisch) von einem Vermittler in das Netzwerk versendet wird. Für einen solchen Broadcast muss der Vermittler vorab keine Kenntnis von den Produzenten bzw. ihren Endpunktadressen haben, entsprechende Protokolle wie z.B. *UPnP* [UPn12], *Zeroconf* [Gut01], *UDP* oder *IP* [Com06] sehen hierfür spezielle Mechanismen vor. Eine Übersicht über verschiedene sogenannte *Discovery-Verfahren* findet sich in [Bad07].

Das SWE-Rahmenwerk beispielsweise sieht vor, dass Produzenten sich proaktiv im System mit einer Reihe von Metadaten anmelden. Die Metadaten werden in Verzeichnissen persistiert und auf Nachfrage von Konsumenten aus den Verzeichnissen geladen. Das Auffinden der Produzenten über entsprechende Protokolle ist nicht spezifiziert. *SenseWrap* [EM09] hingegen delegiert Anfragen von Konsumenten direkt an die Produzenten, welche über Zeroconf-Protokolle aufgefunden werden.

4.4.7. Abfrageverwaltung

Um Kontextinformationen zu erhalten, muss der Konsument eine entsprechende Abfrage formulieren und an einen Vermittler übermitteln. Eine solche Abfrage besteht typischerweise aus mehreren Angaben: Neben Angaben an welchen Aspekten eines Kontextes der Abfragende interessiert ist, gehören hierzu auch Informationen zur gewünschten Quelle, also z.B. die eindeutige Kennung eines Sensors oder Datenstroms, geographische Gebietsangaben oder einfach die Art der gewünschten Information. Außerdem können Filterbedingungen angegeben werden, die sich meist auf Attribute des Kontextes beziehen, z.B. den Zeitpunkt der Erhebung oder Einschränkungen der Messwerte. Zusätzlich können auch noch Verarbeitungsanweisungen Teil einer Abfrage sein. Diese reichen von einfachen Aggregationsfunktionen bis hin zu komplexen Verarbeitungsketten (vgl. Abschnitt 4.4.3).

Die Art und Weise wie eine Abfrage formuliert wird, ist von mehreren Faktoren abhängig. Hierzu gehören beispielsweise die gewünschte zeitliche Dimension der Abfrage, die Erfahrenheit des Konsumenten, die Art des Mediums sowie der Kommunikationskanal. Bezüglich der zeitlichen Dimension lassen sich drei wesentliche Klassen von Abfragen unterscheiden [BDF+07, AV10]:

Ad-hoc Abfragen [4] beziehen sich auf die momentane Situation einer Entität, z.B. „Wie ist die aktuelle Temperatur in Hamburg-Fuhlsbüttel ? “.

Historische Abfragen beziehen sich auf vergangene Situationen, z.B. „Wie hoch war die Niederschlagsmenge in Hamburg im Mai 2012 ? “.

Ereignisbasierte Abfragen [5] werden erst in der Zukunft, bei Eintreten eines bestimmten Ereignisses, beantwortet, z.B. „Versende eine Benachrichtigung, sobald die Temperatur in Hamburg 30° übersteigt“.

Da ein Vermittler je nach zeitlicher Dimension einer Abfrage auf unterschiedliche Dienste zugreifen muss und ad-hoc sowie ereignisbasierte Abfragen einen eigenen Lebenszyklus durchlaufen, muss deren Ausführung durch eine entsprechende Abfrageverwaltung (engl. *Query Management*) koordiniert werden. Diese delegiert Abfragen je nach Art entweder an die Datenverwaltung (historische Abfragen), die Datenstromverwaltung (ereignisbasierte Abfragen) oder einen Produzenten (ad-hoc Abfragen). Von diesen wird die Abfrage schließlich ausgeführt und die Ergebnisse zurück an die Abfrageverwaltung übermittelt.

Die Formulierung von Abfragen kann aufgrund der vielfältigen Möglichkeiten und fehlender Standards und Vereinbarungen bzgl. zu nutzender Technologien ein potentielles Benutzbarkeitsproblem darstellen. Aus diesem Grund müssen Konsumenten durch entsprechende Werkzeuge zur Abfrageerstellung unterstützt werden. Hier spielt die Erfahrung des Nutzers eine Rolle, sodass zwischen normalen Nutzern und Experten unterschieden

[4] Auch als *Live-Abfrage* [BDF+07] oder *Snapshot-Abfrage* [AV10] bezeichnet.
[5] Auch als *Standing Queries* [FRL07] oder *Continuous Queries* [AHS06b] bezeichnet.

werden muss. Normale Nutzer bedürfen einer einfachen Art der Abfrage, idealerweise natürlichsprachlich oder unterstützt durch eine graphische Benutzungsoberfläche und eine integrierte Metadatenabfrage. Für Experten hingegen sind flexiblere, mächtigere und effizientere Abfrageschnittstellen notwendig [JC09]. Im Rahmen des Vermittlungsprozesses sollte eine entsprechende Werkzeugunterstützung zur Formulierung syntaktisch korrekter Abfragen (engl. *Query Design Tools*) für Konsumenten vorgesehen werden.

Neben der Syntax ist vor allem bei ereignisbasierten Abfragen auch die Semantik zu beachten, da hier kein sofortiges Ergebnis zurückgeliefert wird. Um sicherstellen zu können, dass eine ereignisbasierte Anfrage bei Eintreten entsprechender Ereignisse auch die vorgesehenen Ergebnisse liefert, bedarf es Werkzeugen zur Vorab-Validierung dieser Art von Abfragen (engl. *Query Validation Tools*). Da echte Sensoren nur bestimmte Attribute ihrer Umwelt wahrnehmen und diese nicht nach Belieben arrangiert werden kann, können virtuelle Sensorsimulatoren, die geeignete Testdaten produzieren, für diese Aufgabe eingesetzt werden. Der Konsument kann durch Nutzung entsprechender Software-Komponenten, welche die wesentlichen Eigenschaften echter Sensoren näherungsweise nachbilden können, die korrekte Semantik seiner Abfragen überprüfen.

Zusätzlich kann die Art des Eingabemediums und des Kommunikationskanals die Formulierung der Abfrage beeinflussen. Während ein Anwendungsentwickler beispielsweise historische Zeitreihen mittels einer textbasierten SQL-Abfrage über das Netzwerk abfragt, würde ein Bahnkunde die Fahrplaninformationen natürlichsprachlich per Telefon abfragen und ein Tourist Zusatzinformationen zu Sehenswürdigkeiten anhand eines aufgenommenen Fotos beziehen wollen. Entsprechende Adapter für unterschiedliche Abfragesprachen und -medien (engl. *Language and Media Adapter*) müssen vermittlerseitig angeboten werden.

Im Folgenden soll für alle oben erwähnten Klassen von Abfrage die existierenden Möglichkeiten der Abfrageformulierung anhand ausgewählter Beispiele erläutert werden.

Historische Abfragen

Historische Abfragen beziehen sich in kontextbasierten Systemen in erster Linie auf Zeitreihen, d.h. zeitlich geordnete Abfolgen von Daten, z.B. vergangene Messwerte. Neben Zeitreihen können jedoch auch beliebige andere Arten von Daten, z.B. Metadaten, durch diese Abfragen adressiert werden. Beispiel einer historischen Abfrage wäre: „Ermittle die Adressen aller Parzellenbesitzer, in deren Parzellen im Jahr 2012 mehr als 1000 mm Niederschlag gefallen ist“.

Ziel derartiger Abfragen sind Datenbanksysteme in denen historische Daten gespeichert sind. Traditionellerweise werden für solche Arten von Daten relationale Datenbanksysteme verwendet. Seit einigen Jahren finden zudem vermehrt auch objektorientierte und objektrelationale Datenbanksysteme sowie sogenannte NoSQL-Datenbanksysteme Verwendung [KE11, EBF+10], in

denen Daten schemafrei als Dokumente, Spalten, Schlüssel/Wert-Paare oder Graphen abgelegt werden (vgl. Abschnitt 4.4.4).

Für Abfragen im Bereich der relationalen Datenbanksysteme hat sich *SQL:2008* als de-facto Standard etabliert. Die Formulierung entsprechender Abfragen wird bereits durch eine Vielzahl von Werkzeugen unterstützt. Eine Übersicht weiterer, weniger verbreiteter Abfragesprachen für relationale Datenbanken findet sich z.B. in [KE11]. Objektorientierte oder objektrelationale Datenbanken sind weniger stark etabliert. Bei rein objektorientierten Datenbanken existiert mit der *Object Query Language* der Object Management Group ein Standard für Abfragen [CCBB00]. Objekt-relationale Datenbanken bauen häufig auf dem *SQL:2003*-Standard auf, variieren syntaktisch je nach verwendeter Datenbank jedoch leicht [KE11]. Grundsätzlich interessant sind auch Ansätze der objektrelationalen Abbildung (wie z.B. *Hibernate* [Com13b]), bei denen Objekte von Programmiersprachen auf Relationen eines Datenbanksystems abgebildet werden, sodass der Umgang erheblich vereinfacht wird. Im Rahmen dieser Arbeit ist eine solche Abbildung allerdings weniger relevant, da es eine enge Kopplung einzelner Teilsysteme erfordert. Im noch relativ jungen Bereich der NoSQL-Datenbanken hat sich noch kein Abfragestandard etabliert, auch weil die Datenverwaltung und -verarbeitung teils gänzlich unterschiedlichen Konzepten folgen. Hier konkurrieren eine Vielzahl von Ansätzen (eine Übersicht findet sich z.B. in [EBF+10]), wobei einige ebenfalls auf der SQL-Syntax aufbauen.

Die Verwendung historischer Abfragen findet sich zum Beispiel bei der Umsetzung der *Sensor Web Enablement*-Standards im Projekt *52°North* (siehe Abschnitt 6.1.2). In der Implementierung werden Zeitreihen und Sensormetadaten in einer relationalen PostgreSQL-Datenbank gespeichert. Der Zugriff des Konsumenten jedoch wird durch Web Service-Schnittstellen der *Sensor Instance Registry* und des *Sensor Observation Service* gekapselt, sodass der Konsument seine Abfrage in Form einer parametrisierten SOAP-Nachricht übermittelt [52N12b, Ope10a, Ope12a].

Ereignisbasierte Abfragen

Ereignisbasierte Abfragen sollen ein Ergebnis liefern, sobald ein bestimmtes Ereignis eingetreten ist, z.B. „Gebe den Ort eines Sensors aus, sobald dieser eine Temperatur von mindestens 80° Celsius gemessen hat“.

In klassischen Datenbanksystemen können für die Ausführung solcher und anderer Aktionen sogenannte *Trigger*-Funktionen verwendet werden. Diese werden aufgerufen, sobald Änderungen an einer Tabelle durchgeführt wurden, z.B. neue Datensätze eingefügt oder vorhandene geändert wurden [KE11]. Häufig werden diese Funktionen benutzt, um Konsistenzbedingungen sicherzustellen, können jedoch auch über *Stored Procedures* nahezu beliebigen Programmcode anderer Programmiersprachen ausführen und somit für ereignisbasierte Abfragen verwendet werden. Die Formulierung von Trigger-Funktionen geschieht meist mittels proprietärer Programmiersprachen der

entsprechenden Datenbankmanagementsysteme (z.B. *PL/SQL* [Ora12] oder *PL/pgSQL* [Pos12]).

In kontextbasierten Systemen ist es jedoch oft nicht üblich alle eintreffenden Kontextdaten in einem klassischen Datenbanksystem zu hinterlegen [BDF+07]. Gründe hierfür sind mannigfaltig. Viele Systeme beobachten beispielsweise die Umwelt, sollen aber nur bei abnormalen Bedingungen, z.B. einem Waldbrand, eine Aktion auslösen. Kontextdaten, die lediglich den Normalzustand beschreiben, sind weniger relevant und müssen daher nicht unbedingt persistiert werden. Außerdem werden die Daten kontinuierlich erhoben und bei Betrachtung größerer Zeitskalen würden klassische Datenbanksysteme mit der Menge an Daten nicht umgehen können, da sie zu schwergewichtig und langsam sind [BDF+07].

Statt solche Daten also in klassischen Datenbanksystemen zu speichern, werden zur Detektion von Bedingungen sogenannte datenstromverarbeitende (engl. *Data Stream Processing*, DSP) Systeme bzw. Systeme, die komplexe Ereignisse verarbeiten können (engl. *Complex Event Processing*, CEP), verwendet [MC11, Luc02]. Während in einer Datenbank die Daten gespeichert und von Zeit zu Zeit Abfragen an die Datenbank gestellt werden, speichern derartige *Datenstrom Management Systeme* (DSMS) die Abfragen und werten diese kontinuierlich gegen eintreffende Daten aus. Anschließend werden die Daten verworfen. Erst wenn eine Abfrage mit eintreffenden Daten in Übereinstimmung gebracht wird, wird ein Ergebnis generiert. Ist man nicht weiter an dem Ergebnis einer Abfrage interessiert, so muss die Abfrage explizit aus dem System entfernt werden.

Auch wenn bereits mit relationalen Abfragesprachen wie SQL relativ komplexe Abfragen, z.B. mit Aggregationen und Filterbedingungen über mehrere Tabellen, gestellt werden können, so erlauben viele datenstromverarbeitende Systeme noch weitaus mächtigere Abfragen, da sie aufgrund des zeitlich-sequenziellen Charakters von Datenströmen häufig auch kausale und temporale Bedingungen überprüfen können. In diesem Fall spricht man auch von der Fähigkeit der komplexen Ereignisverarbeitung [MC11, Luc02]. Ein Beispiel für eine komplexe Abfrage wäre: „Erzeuge einen neuen Sturmflut-Alarm, sobald in 3 aufeinander folgenden Stunden Pegelstation B nach Pegelstation A einen Pegelanstieg von mehr als 1 Meter meldet und Rückhaltebecken X innerhalb der letzten 3 Stunden kein Offen-Ereignis angezeigt hat“.

Betrachtet man die einzelnen Datenströme in DSP/CEP-Systemen als zeitlich ephemere Tabellen, so weisen solche Systeme grundlegende Ähnlichkeiten mit relationalen Datenbanken auf. Aus diesem Grund folgt die Syntax vieler Abfragesprachen für DSP/CEP-Systeme auch den SQL-Standards (z.B. Esper EQL [Esp13a] und CQL [ABW06]) [LWZ04]. Entwicklern mit Vorkenntnissen aus der Welt der Datenbanken wird auf diese Weise ein einfacher Einstieg ermöglicht. Lediglich Konstrukte zur Formulierung kausaler und temporaler Aspekte müssen neu erlernt werden. Ein prominentes Beispiel für derartige Systeme aus der Open Source-Gemeinde ist *Esper* [Esp13a]. Eine Übersicht weiterer Systeme findet sich unter anderem in [GRHY09].

Manche Systeme (z.B. Esper [Esp13a]) sehen zudem Konstrukte vor, die es erlauben, innerhalb einer ereignisbasierten Abfrage zusätzlich auch historische Abfragen an Datenbanken zu stellen. Derartig kombinierte Abfragen erleichtern es den Nutzern, neue Ereignisse im Kontext vergangener Beobachtungen besser einzuordnen [BDF+07].

Eine Art ereignisbasierte Abfrage findet sich beispielsweise bei der *SAP Auto-ID Infrastructure* (siehe Abschnitt 6.2.3). Hier können zur Laufzeit regeln definiert werden, die bei Eintreffen neuer Nachrichten von einer *Rule Engine* gegen eine Menge von Bedingungen ausgewertet werden und daraufhin Aktionen, z.B. eine Benachrichtigung, auslösen können [BLHS04]. Regeln werden als XML-Dokument repräsentiert, welches einem proprietärem Schema folgt. Alle Ereignisse sowie Metadaten werden zudem in Datenbanken persistiert. Auch *SStreamMWare* (siehe Abschnitt 6.1.8) verwendet ereignisbasierte Abfragen. Diese werden ebenfalls in einem proprietärem Format formuliert, um in der Hierarchie der Knoten verteilt verarbeitet werden zu können. Kontinuierlich eintreffende Messwerte von Sensoren sollen so möglichst weit unten in der Hierarchie zu höherwertigen Ereignissen aggregiert werden [GRL+08].

Ad-hoc Abfragen

Eine ad-hoc Abfrage soll die momentane Situation einer Entität bzw. eines Ortes zurückliefern. Eine Abfrage der aktuellen Windstärke und -richtung an der Hamburger Außenalster wäre ein Beispiel hierfür. Da Sensoren in den meisten Fällen von sich aus periodisch ihre Messwerte melden, findet eine solche Art von Abfrage selten Verwendung, da in der Praxis meist der zuletzt gemessene, persistierte Wert mittels einer historischen Abfrage ermittelt wird.

In zwei Fällen reicht dieses Vorgehen allerdings nicht aus: i) falls der zuletzt gemessene Wert zu weit in der Vergangenheit liegt und ii) falls die letzte Messung eines mobilen Produzenten (z.B. eines Satelliten) nicht an dem gewünschten Ort geschah. In beiden Fällen muss der Konsument dem Produzenten einen Auftrag erteilen, eine erneute Messung vorzunehmen. Dies bezeichnet man als *Tasking* [BEJ+11] (siehe auch Abschnitt 4.4.8). Nach Übermittlung eines solchen Auftrags schickt der Konsument eine weitere ereignisbasierte Abfrage, die bei Eintreffen des nächsten gewünschten Messwertes beantwortet wird.

Insbesondere bei ressourcenarmen Sensoren ist eine ad-hoc Abfrage jedoch oft nicht möglich, da derartige Sensoren nach Übermittlung eines Messwertes in einen Energiesparmodus wechseln und dabei ihren Kommunikationskanal schließen [MSPS08, MFHH05, HMCP04], sodass Aufträge erst in der nächsten Messperiode zugestellt würden und damit in vielen Fällen obsolet wären. Leistungsstärkere Sensoren oder insbesondere auch sogenannte *Smart Transducer* [SL08], die neben Sensoren auch Aktuatoren steuern können, erlauben hingegen zum Teil solche ad-hoc Abfragen, da sie kontinuierlich über das Netzwerk erreichbar sind.

In [SL08] wird zum Beispiel ein Transducer Web Service vorgestellt, der SOAP-Abfragen über unterschiedliche Kommunikationskanäle entgegennehmen kann, diese gemäß den *IEEE 1451.x*-Standards [Nat12b] übersetzt und zur Ausführung an die Sensoren und Aktuatoren weiterleitet. In einem Anwendungsbeispiel zeigen die Autoren, wie sich auf diese Weise ad-hoc Abfragen stellen und verarbeiten lassen. Bei einem anderen Beispiel der NASA aus [MSPS08] nimmt ein Satellit ein besonderes Phänomen wahr und verschickt daraufhin Aufträge an weitere Satelliten sowie unbemannte Luft- und Bodenvehikel, z.B. über einen *Sensor Planning Service* (einem Sensor Web Enablement-Standard), um weitere Details über dieses Phänomen zu sammeln. In [KGZ07] wird ein Sensornetz, bestehend aus Mobiltelefonen, vorgestellt, bei dem Programmierer Aufträge an Teilnehmer verschicken können, damit diese ad-hoc Daten über ihr Telefon sammeln und übermitteln können. Beide Beispiele bauen auf Web Service-Schnittstellen auf, d.h. eine Abfrage besteht aus einer SOAP-Nachricht, mit spezifizierten Abfragefeldern. Eine Unterstützung bei der Abfrageformulierung wird meist nur für das Erzeugen der SOAP-Nachricht über HTML-Formulare geboten, nicht jedoch für die Inhalte der Abfragefelder.

Abfragen mittels verschiedener Medien

Einleitend wurde festgestellt, dass es drei verschiedene Klassen von Abfragen gibt, die jeweils eigene Anforderungen an die Formulierung stellen und es entsprechend individueller Dienste und Werkzeuge zur Unterstützung bedarf. Dabei wurde implizit davon ausgegangen, dass Abfragen als Text formuliert und in Form einer Nachricht an einen Vermittler gesandt werden.

Dies muss jedoch nicht zwangsläufig der Fall sein. Ebenso ist es beispielsweise möglich eine Abfrage in mündlicher bzw. allgemein akustischer Form oder als Bild in visueller Form zu übermitteln. Beispiele hierfür wären der Telefonanruf bei einer elektronischen Auskunft, das im *Apple iPhone* integrierte *Siri* [Wik12e], die Abfrage von Informationen über ein aktuell gespieltes Musikstück mittels akustischem Fingerabdruck (z.B. mittels *Soundhound* [Sou12]) oder das Abrufen zusätzlicher Informationen zu einer Sehenswürdigkeit anhand eines Fotos bei *Google Goggles* [Goo12c].

Derartige Abfragen bedürfen auf Konsumentenseite oft wenig bis gar keiner Unterstützung im Rahmen des Vermittlungsprozesses, da die Formulierung der Abfrage sehr natürlich ist (z.B. Sprache) bzw. die Abfrage keiner besonderen Syntax folgen muss. Auf Seite eines Vermittlers hingegen müssen besondere Dienste bereitgestellt werden, um derlei Abfragen dahingehend transformieren zu können, dass sie z.B. einer Datenbank oder der Datenstromverarbeitung zugeführt werden können. Zu solchen Diensten gehören beispielsweise eine Spracherkennung oder Software zur Interpretation von Bildern, welche die wesentlichen Informationen aus dem übertragenen Medium extrahieren und in eine textuelle Abfrage transformieren können. Die weitergehende Abfrage-

verarbeitung erfolgt dann seitens eines Vermittlers analog zu rein textuellen Abfragen.

Abfrageunterstützung durch Datenvisualisierung und -exploration

Wie bereits erläutert, kann die Formulierung sinnvoller Abfragen durchaus schwierig sein. Abgesehen von der Verwendung der korrekten Syntax einer Abfragesprache müssen Metainformationen über Datenquellen und Attribute zur Verfügung stehen, sodass auch etwaige Bezeichner richtig verwendet werden. Ist all dies gegeben, kann es trotzdem passieren, dass eine Abfrage keine sinnvollen oder zumindest nicht die erwarteten Resultate liefert. Eine Abfrage nach der Durchschnittstemperatur über die letzten 50 Jahre in Hamburg liefert zum Beispiel dann kein sinnvolles Ergebnis, wenn in der Datenbank lediglich die Werte der letzten Woche gespeichert sind. Der Abfragende kann dies anhand des Abfrageergebnisses nicht feststellen. Daher ist es hilfreich, sich zunächst einen Überblick über z.B. die historischen Verläufe und die Wertespektren der Daten zu verschaffen [Le11].

Datenvisualisierung kann hierfür als ein Werkzeug zum phänomenologischen Verständnis von Zusammenhängen angesehen werden [LAE+10]. Ein solches Werkzeug bildet auch die Grundlage für die visuelle Datenexploration sowie visuelle Data Mining-Verfahren [Kei02], mit deren Hilfe sich implizite Muster bzw. Strukturen in Datenbeständen identifizieren lassen. Auf diese Weise helfen Visualisierung und Analyse bei der Erfassung von Charakteristika großer Datenbestände, was insbesondere in der Handhabung von Sensordaten ein unverzichtbarer Arbeitsschritt ist [BDF+07].

Bezüglich der Abfrageunterstützung werden in [Le11] geeignete Unterstützungspunkte für kontinuierliche Abfragesprachen vorgestellt, für jeden Unterstützungspunkt passende Visualisierungstechniken vorgeschlagen und gezeigt, dass die Formulierung selbst komplexer Abfragen durch Verwendung von Metadaten in Kombination mit Datenvisualisierungen unterstützt werden können.

Abfrageunterstützung durch Simulation von Produzenten

Während bei historischen und ad-hoc Abfragen zeitnah ein Ergebnis zurückgeliefert wird, kann eine Rückmeldung bei ereignisbasierten Abfragen beliebig verzögert oder ggf. auch gar nicht erfolgen, da derlei Abfragen in die Zukunft gerichtet sind. Außerdem umfassen Sprachen für ereignisbasierte Abfragen eine Reihe weiterer Sprachkonstrukte, z.B. um temporale und kausale Abhängigkeiten oder komplexe Muster zu definieren, was die Komplexität der Abfrageformulierung weiter erhöht.

Auch wenn das Formulieren von Abfragen durch graphische Benutzungsoberflächen oder Integration von Metadaten erleichtert werden kann, so verbleibt stets ein Restrisiko, ob die Abfrage tatsächlich bei der gewünschten Ereigniskonstellation ein Ergebnis liefert. Es bedarf daher des Testens solcher

Abfragen. Da die den Abfragen zugrundeliegenden Daten allerdings von Sensoren erhoben und deren Umwelt nicht beliebig arrangiert werden kann, um die gewünschte Ereigniskonstellation zu provozieren, gestaltet sich das Testen schwierig.

Eine Lösung hierfür bietet die Simulation von Produzenten, deren Verhalten mittels virtueller Sensoren nahezu beliebig nachgeahmt werden kann. Solche simulierten Sensoren können vom Konsumenten dahingehend konfiguriert werden, dass sie die gewünschte Ereigniskonstellation imitieren und entsprechend künstlich erzeugte Sensorwerte produzieren. Anhand dieser Daten lassen sich somit auch ereignisbasierte Abfragen validieren.

Es existiert eine Vielzahl von Werkzeugen zur Simulation von Sensoren bzw. Sensordaten. Vielfach, wie z.B. *J-Sim* [SCH+05], fokussieren diese jedoch insbesondere die Simulation von Sensornetzen und bieten neben der reinen Ausgabe von Sensordaten auch Funktionen zum Testen von Sensorprogrammen sowie zur Simulation von Kommunikationsverbindungen und der Datendiffusion bei Verwendung unterschiedlicher Routing-Protokolle etc., wodurch deren Bedienung zu komplex für obigen Anwendungsfall ist. Des Weiteren finden sich Simulationsrahmenwerke auch für spezielle Geräte, als Beispiel seien hier der Simulator für Oracles *SunSPOT*-Sensoren [Sun12] sowie der *SensorSimulator* [Pro13] für die Android-Plattform genannt. Mit *CASS* [PMHY07] und *Siafu* [NEC09] existieren zudem Rahmenwerke und Werkzeuge zur Simulation von Kontextdaten, die nicht nur reale Sensoren simulieren, sondern echte Geräte auch transparent einbinden können.

Implikation für die Abfrageverwaltung

Die Aufgabe der Datenabfrage befasst sich im Vermittlungsprozess mit der Formulierung von Abfragen mittels verschiedener Medien. Traditionellerweise werden Abfragen als Text formuliert, was jedoch zu Benutzbarkeitsproblemen führen kann. Andere Möglichkeiten der Abfrageformulierung mittels Bild und Ton stellen Alternativen hierzu dar. Ungeachtet des Mediums lassen sich drei wesentliche Abfrageklassen unterscheiden: i) historische Abfragen, die auf Zeitreihen vergangener Messungen angewendet werden, ii) ad-hoc Abfragen, die ein möglichst aktuelles Ergebnis liefern sollen, und iii) ereignisbasierte Abfragen, die in die Zukunft gerichtet sind.

In allen Klassen existieren eine Vielzahl von Abfragesprachen. Für die korrekte Formulierung einer Abfrage kann der Vermittlungsprozess den Abfragesteller durch geeignete Dienste und Werkzeuge unterstützen, z.B. indem über die Metadatenverwaltung (siehe Abschnitt 4.4.6) Metainformationen von Produzenten und Messwerten abgefragt werden und diese von entsprechenden Werkzeugen bei der Unterstützung der Abfrageformulierung eingesetzt werden (vgl. Abschnitt 5.4.7).

Weitergehend kann auch die Visualisierung von Zeitreihen eine Hilfe bei der Abfrageformulierung darstellen, da anhand historischer Daten nicht nur ein Überblick über Verfügbarkeit und Werteskalen gewonnen werden kann,

sondern auch implizit in den Daten enthaltene Muster bzw. Strukturen aufgedeckt werden können. Auch die Transformation von nicht-textuellen Abfragen, z.B. mittels Sprache oder Bildern, kann durch einen Vermittler übernommen werden, um den Konsumenten von dieser Aufgabe zu befreien und eine natürlichere Art der Abfrage zu erlauben.

Speziell zur Validierung der Korrektheit ereignisbasierter Abfragen lassen sich zudem simulierte Sensoren einsetzen, welche das Verhalten realer Sensoren imitieren und somit beliebige Ereigniskonstellationen produzieren können.

4.4.8. Auftragsverwaltung

In vielen kontextsensitiven Systemen verläuft die Kommunikation unidirektional vom Produzenten, welcher Messwerte an einen Konsumenten bzw. einen Vermittler überträgt. Es gibt jedoch eine Reihe von Anwendungsfällen, in denen Konsumenten Aufträge (engl. *Tasks*) an Produzenten schicken müssen. Beispiele hierfür sind die Kalibrierung oder Konfiguration eines Sensors, die ad-hoc Abfragestellung, das Ansteuern eines Aktuators als Bestandteil eines Smart Transducers oder das Schreiben von Informationen auf ein RFID Tag. Im Rahmen dieser Beauftragung (engl. *Tasking*) müssen zudem eventuell weitere Verwaltungsnachrichten, z.B. die Abfrage von Metadaten, Überprüfung der Durchführbarkeit eines Auftrags (engl. *Feasibility Analysis*), Statusabfragen, Abbruch-Anforderungen etc. an den Produzenten versendet werden [Ope11].

Hierbei eröffnen sich eine Reihe von Herausforderungen für die Auftragsverwaltung (engl. *Task Management*) im Rahmen des Vermittlungsprozesses. Auf Ebene des Konsumenten besteht die Herausforderung in der Formulierung des Auftrags. Je nach Anwendungsfall gibt es eine Vielzahl (teils proprietärer) Sprachen bzw. Dienstschnittstellen, die dies erlauben. Für das Beschreiben von RFID Tags hat GS1 den *EPC UHF Gen 2*- und den *EPC HF*-Standard [EPC10] verabschiedet, der die Kommunikation mit der RFID Middleware bzw. den RFID-Lesegeräten über Web Service-Schnittstellen regelt und entsprechende Schnittstellen auch für das Beschreiben von RFID Tags vorsieht. Kommunikation mit Smart Transducern ist durch *IEEE 1451* standardisiert und legt ebenfalls das Nachrichtenformat für die Kommunikationsschnittstellen des Transducers fest, allerdings auf den unteren Ebenen des ISO/OSI-Modells [Nat12b]. Auf höheren Ebenen spezifiziert beispielsweise die OGC u.a. mit dem *Sensor Planning Service* ein Schnittstellenmodell zur Beauftragung von Sensoren [Ope11]. Standards der höheren Ebenen nutzen häufig XML in Kombination mit SOAP oder JSON in Verbindung mit einer REST-basierten Schnittstelle zur Formulierung und Übertragung von Nachrichten, jedoch unterscheiden sich die Nachrichteninhalte und somit die Syntax zur Formulierung von Aufträgen beträchtlich.

Im Rahmen des Vermittlungsprozesses ist dies nur peripher relevant, da die Produzenten als Black Box verstanden werden [Ope11] und die Nachrichteninhalte ausschließlich von diesen interpretiert werden müssen. Das heißt

auf Seite der Produzenten bedarf es keiner direkten Unterstützung seitens des Vermittlungsprozesses. Konsumenten können durch geeignete Werkzeuge zur Formulierung von Aufträgen unterstützt werden, wobei dies durch die Verschiedenartigkeit der Standards lediglich auf syntaktischer Ebene durch Werkzeuge zur Formulierung von SOAP- oder JSON-kodierten Nachrichten sowie durch Abfragemöglichkeiten für Metadaten möglich ist. Ein Vermittler muss den Auftrag eines Konsumenten nur zu dem entsprechenden Produzenten delegieren. Dies bedarf insofern der Unterstützung, als dass der Produzent unter Umständen nicht immer erreichbar ist. Ein Sensor kann vorübergehend in den Schlafmodus wechseln und RFID Tags können aus dem Einflussbereich des RFID-Lesers verschwinden bzw. mobile Sensoren aus dem Bereich ihres letzten Kommunikationspunktes [AIM10]. Außerdem ist es, je nach verwendetem Kommunikationskanal, häufig nicht möglich, Push-Nachrichten an die Geräte zu verschicken, wenn diese sich in Netzwerken mit privater Adressierung (z.B. hinter einem NAT Router) befinden [VHP06, CDK12].

In all diesen Fällen muss im Rahmen des Vermittlungsprozesses die Nachricht zwischengespeichert und bei nächstmaligem Kontakt mit dem Produzenten übermittelt werden. Schließlich wird der Konsument idealerweise sowohl über die Nicht-Erreichbarkeit des Produzenten als auch über den Status der Ausführung informiert.

4.5. Zusammenfassung

Die Heterogenität von Kontextdatenquellen, dies umfasst insbesondere die unterschiedlichen Arten von Kontext, die verwendeten Datenformate, Protokolle, Kommunikationstechnologien etc. erschweren die Integration von Kontextinformationen auf Konsumentenseite. Um diese Informationen nutzen zu können, müssten sich die Konsumenten an die jeweiligen Datenquellen anpassen, was neben einem hohen Aufwand auch eine Reihe von Einschränkungen mit sich bringt. Um Kontextdaten heterogener Quellen einer möglichst großen Menge an Konsumenten unter Berücksichtigung ihrer jeweiligen Anforderungen zur Verfügung stellen zu können, bedarf es entsprechend einer Instanz, die zwischen diesen Rollen vermitteln kann. Hierfür wurde in diesem Kapitel ein Vermittlungsprozess vorgestellt, welcher Produzenten und Konsumenten entkoppeln und entlasten, den Entwicklungsaufwand für Anwendungen reduzieren, die Wiederverwendbarkeit von Funktionen unterstützen und somit auch komplexe kontextsensitive Anwendungen (auf mobilen Geräten) ermöglichen kann.

Dieser Prozess umfasst eine Reihe von Aufgabenbereichen, die zusammengenommen alle notwendigen Belange einer Vermittlung abdecken. Hierzu gehören unter anderem die Erhebung von Daten auf Produzentenseite und die Übertragung über verschiedenartige Kanäle an einen Vermittler. Dieser hat die Aufgaben die Daten aufzubereiten und ggf. zu persistieren, wobei je nach Art der Daten und Anwendungserfordernisse unterschiedliche Arten von Datenhaltungssystemen zum Einsatz kommen können. Außerdem muss ein

Vermittler Abfragen der Konsumenten auf den Datenströmen bzw. Zeitreihen ausführen und Aufträge zur Beeinflussung der Datenerhebung verwalten. Hierbei bedarf es insbesondere einer Unterstützung der Konsumenten durch Werkzeuge zur Formulierung und Validierung von Abfragen, da diese ein potenzielles Akzeptanz- und Benutzbarkeitsproblem darstellen. Hierfür wurden unterschiedliche Ansätze, z.B. das Bereitstellen von Metadaten für Werkzeuge, semantische Abbildungen und Ableitungen sowie das Testen von Abfragen mittels Sensorsimulationen skizziert.

Die grundlegende Idee der Vermittlung ist, dass Konsumenten zusammen mit einer Abfrage nach Informationen auch gleich formulieren, in welcher Form sie die Ergebnisse präsentiert bekommen möchten. Auf diese Weise ist es möglich, den unter Umständen beträchtlichen Aufwand zur Aufbereitung der Informationen in die Verantwortung eines Vermittlers als Leistungserbringer zu stellen. Dies entlastet die Konsumenten und ermöglicht eine aufwandsarme Anwendungsintegration der Kontextinformationen. Um dies zu ermöglichen, wird im nächsten Kapitel das Konzept der *kontextbasierten Sichten* eingeführt und ausgehend von einer Anforderungsanalyse die Herausforderungen sowie potentielle Lösungsansätze für einen späteren Systementwurf identifiziert.

5. Aufbereitung und Vermittlung kontextbasierter Sichten

Im vorigen Kapitel wurde der Vermittlungsbegriff eingeführt und die zu dem Vermittlungsprozess gehörigen Aufgabenbereiche und Aufgaben detailliert. Motiviert wurde die Vermittlung unter anderem durch das Erfordernis einer einfacheren Integration von Kontextinformationen in bestehende Anwendungen auf Konsumentenseite. Um dies zu ermöglichen, kann der Konsument zusammen mit einer Angabe für welche Kontextinformationen er sich interessiert, auch Verarbeitungsanweisungen in Form einer Orchestrierungsbeschreibung für Verarbeitungsdienste übergeben. Mit Hilfe dieser Beschreibung kann ein Vermittler die gewünschten Kontextdaten dahingehend aufbereiten und ggf. in eine neue Repräsentation überführen, dass Konsumenten die Daten ohne weiteren Aufwand in den eigenen Programmlauf integrieren können. Somit ist ein Konsument unabhängig von etwaigen Produzenten und muss seine Programmlogik bei einem Wechsel der Kontextdatenquelle nicht anpassen.

In diesem Kapitel soll diese Idee aufgegriffen und weiter detailliert werden. Hierzu wird zunächst in Abschnitt 5.1 der Begriff der *kontextbasierten Sicht* definiert und mit dem Vermittlungsbegriff in Beziehung gesetzt werden. Im Anschluss daran werden in Abschnitt 5.2 Anwendungsbeispiele aufgeführt, die den Nutzen anhand praktischer Szenarien demonstrieren. Auf Basis dieser Anwendungsszenarien wird darauffolgend in Abschnitt 5.3 eine Anforderungsanalyse durchgeführt und in Abschnitt 5.4 werden schließlich die sich ergebenden Teilprobleme sowie geeignete Lösungsansätze für den Vermittlungsprozess identifiziert, welche die Grundlage für eine spätere Bewertung verwandter Arbeiten und den darauffolgenden Entwurf und die Implementierung eines exemplarischen, prototypischen Vermittlers darstellen.

5.1. Einführung und Begriffsklärung

Im Rahmen dieser Arbeit soll der Begriff einer Sicht (engl. *View*) in Anlehnung an die Verwendung im Bereich der Datenbanken bzw. Datenbanksysteme aufgefasst werden. Dort wird der Begriff als Teilmenge von Informationen über einem logischen Schema verstanden und zur Vereinfachung der Abfragestellung sowie zur Optimierung der Abfragen verwendet [KE11]. Eine Sicht auf Daten kann im Allgemeinen mehrere Formen annehmen [SSH10]:

- Auf Basis einer größeren Datenmenge kann eine Sicht einen begrenzten Ausschnitt auf diese Datenmenge repräsentieren. Das bedeutet, es werden Elemente der Ursprungsmenge gefiltert und nur diejenigen Elemente, die den Bedingungen einer Sicht genügen, als Teil dieser

berücksichtigt. Diejenigen Temperatursensoren, die ein Messintervall von weniger als 1 Minute haben, könnten z.B. eine Sicht auf die Menge aller Temperatursensoren konstituieren. Dies bezeichnet man als eine *Selektionssicht*.

- Während die Selektionssicht Elemente filtert, betrifft die *Projektionssicht* die Attribute von Elementen. Eine solche Sicht könnte z.B. den Fokus auf lediglich die Positionsangaben aller Temperatursensoren in Hamburg legen.

- Eine *Aggregationssicht* fasst die ursprüngliche Datenmenge anhand einer bestimmten Funktion zu einem oder wenigen Elementen zusammen. Die Durchschnittstemperatur des letztes Jahres in Hamburg oder die Anzahl an Temperatursensoren sind Beispiele für eine solche Form von Sicht.

- Verknüpft man zwei unabhängige Datenmengen miteinander, so spricht man von einer *Verbundsicht*. Alle Besitzer von Temperatursensoren in Hamburg stellt zum Beispiel eine Sicht auf der Menge der Temperatursensoren in Hamburg sowie der Menge der Personen, die einen solchen Sensor besitzen, dar.

- Eine Sicht kann auch mehreren der obigen Formen gleichzeitig zuzuordnen sein. So stellt zum Beispiel das Durchschnittsalter aller Besitzer von Temperatursensoren in Hamburg eine Aggregationssicht über einer Projektionssicht über einer Verbundsicht dar.

Kemper und Eickler betonen in [KE11], dass Sichten immer auf die Bedürfnisse der jeweiligen Benutzer zugeschnitten sind. Allerdings bezieht sich das im Wesentlichen auf eine einfache Auswahl relevanter Daten. Weitergehende Bedürfnisse, z.B. bezüglich der Repräsentation von Daten oder der Anreicherung von Daten mit zusätzlichen Informationen aus externen Quellen, werden nicht bedient, da dies über den intendierten Zweck von Datenbanken hinaus geht. So könnte ein Benutzer beispielsweise statt eines XML-kodierten Ergebnisses ein JSON-Objekt, statt der hinterlegten Längen- und Breitengrade eines Sensors, eine postalische Adresse bevorzugen oder anhand der Modellbezeichnung eines Sensors zusätzlich auch dessen durchschnittliche Messabweichung vom Hersteller erfragen wollen.

Es lassen sich nahezu beliebige Beispiele finden, die zwar eine Sicht darstellen, aber nicht in eine der obigen Sichtklassen einzuordnen sind. Dabei wurde in Abschnitt 4.1 durch die Analogie mit „Vermittlung von Wissen“ und „Vermittlung von Eindrücken“ hervorgehoben, dass die Kontextdatenvermittlung explizit auch die Aufbereitung der Kontextdaten nach Maßgabe des Konsumenten umfasst, sodass diese verständlich und aufgabenadäquat präsentiert werden können. Eine ähnliche Auffassung einer Sicht findet sich auch in [Luc02], wo eine Sicht als eine Auswahl von Informationen verstanden wird, die verarbeitet werden, um lediglich diejenigen Aspekte zu abstrahieren oder

zu extrahieren, die für das Verständnis eines Problems von Interesse sind. Auch [MS06] spricht von einer „Sicht auf Daten“, welche lediglich diejenigen Kontextinformationen exponiert, die einen Benutzer im Umgang mit seinen Anwendungen unterstützt.

In Anlehnung daran soll im Rahmen dieser Arbeit daher ein weiterer Sichtbegriff eingeführt werden: *kontextbasierte Sicht*. Diese Art von Sicht umfasst und ergänzt oben genannte Sichten um eben solche Aspekte wie bevorzugte Repräsentation der Ergebnisse, Anreicherung der Ergebnisse mit Informationen externer Quellen etc. Um dies zu erreichen, müssen bei der Generierung einer Ergebnismenge weitergehende Verarbeitungsschritte, z.B. eine umgekehrte Geokodierung oder das Abrufen weiterer Informationen, erfolgen. Da beliebige Verarbeitungsschritte vorstellbar sind, ließe sich der Begriff der kontextbasierten Sicht leicht zu *benutzerdefinierte Sicht* erweitern. Um jedoch den Anwendungsbereich und damit die zur Verfügung stehenden Verarbeitungsfunktionen (vgl. Abschnitt 2.3) eingrenzen zu können, beschränkt sich diese Arbeit auf den Begriff der kontextbasierten Sicht, der wie folgt definiert wird[1]:

> *Eine **kontextbasierte Sicht** beschreibt eine Sicht auf eine Menge von Kontextdaten, die alle für einen konkreten Anwendungszweck relevanten Elemente in einer geeigneten Repräsentation enthält.*

Das bedeutet, eine kontextbasierte Sicht kann von einem Konsumenten direkt in den eigenen Programmablauf integriert und für den eigentlich intendierten Zweck genutzt werden, ohne dass konsumentenseitig weitere Verarbeitungsschritte erforderlich sind. Hierdurch reduziert sich der Integrationsaufwand und der Konsument wird entlastet. Außerdem können so auch Verarbeitungsfunktionen auf den Kontextdaten angewendet werden, für deren Ausführung der (mobile) Konsument nicht über ausreichend Ressourcen verfügt. Jedoch erhöht sich für diesen der Aufwand für die Abfragestellung, da bei einer Abfrage von Informationen nunmehr nicht nur relevante Abfragekriterien beschrieben werden müssen (wie z.B. bei einer herkömmlichen Datenbankabfrage), sondern zudem auch die auszuführenden Verarbeitungsschritte zur Erzeugung einer kontextbasierten Sicht angegeben werden müssen.

Sowohl die Formulierung der Abfrage als auch die Ausführung der Verarbeitungsschritte sowie aller damit zusammenhängenden und im vorigen Kapitel detaillierten Aufgaben sind Teil des Vermittlungsprozesses. Während eine kontextbasierte Sicht also das endgültige Resultat beschreibt, repräsentiert der Vermittlungsprozess den Weg zur Erzeugung dieses Resultats. Diese Zusammenhänge werden in den folgenden Abschnitten näher betrachtet.

[1] Eine alternative Definition könnte eine kontextbasierte Sicht auch als eine Sicht beschreiben, die in einem bestimmten Kontext überhaupt erst entsteht. Zum Beispiel könnte eine solche Sicht beliebige Informationen über einen Sensor enthalten, bei dem soeben eine Fehlfunktion gemeldet wurde (d.h. eine Änderung des Kontextes eintrat). Im Rahmen dieser Arbeit soll der Fokus jedoch bei der Aufbereitung der Informationen sowie deren adäquater Präsentation liegen.

5.2. Anwendungsbeispiele

Um die Aufbereitung und Vermittlung kontextbasierter Sichten weitergehend zu veranschaulichen und mögliche Anwendungsszenarien in der Praxis aufzuzeigen, sollen im Folgenden drei Anwendungsbeispiele etwas detaillierter vorgestellt sowie weitere Einsatzmöglichkeiten kurz skizziert werden.

5.2.1. Intelligente Wohnumgebungen

Mit dem Begriff *Smart Home* werden intelligente Wohnumgebungen bezeichnet, welche die Fähigkeiten besitzen, sich autonom den Bedürfnissen ihrer Bewohner und den Erfordernissen der Umwelt anzupassen. Die Ziele dabei sind eine höhere Wirtschaftlichkeit, ökologisch sinnvoller Umgang mit Ressourcen, ein gesteigerter Unterhaltungswert, größerer Komfort und mehr Sicherheit [CYD06]. So kann die intelligente Wohnung beispielsweise den Stromverbrauch durch automatisches Abschalten von Lampen optimieren, sobald alle Personen einen Raum verlassen haben, oder die Heizung einschalten, sobald sich der erste Bewohner von der Arbeit auf den Heimweg macht. Der intelligente Kühlschrank warnt von sich aus vor dem Verzehr abgelaufener Lebensmittel, der Fernseher nimmt automatisch das laufende Programm auf, sobald der Zuschauer durch Telefon- oder Türklingeln gestört wird und bei Unfällen werden automatisch entsprechende Notrufstellen benachrichtigt [CYD06, GB00, HLFP99, Moz98]. Solche und ähnliche Beispiele zeigen die Möglichkeiten für intelligentes Verhalten eines Smart Homes.

Um diese Beispiele realisieren und allgemein die Ziele erfüllen zu können, muss die Wohnung den Zustand der Umwelt sowie die Aktivitäten der Bewohner erfassen können. Die Ziele können dabei mittels Regeln beschrieben werden, die festlegen, wie die Wohnung in bestimmten Situationen, d.h. unter welchen Kontextbedingungen, reagieren soll. Sobald eine solche Regel erfüllt wird, können beispielsweise Aktuatoren befehligt werden, auf die Umwelt einzuwirken, um die definierten Zielvorgaben zu erfüllen. Doch da der Bereich der Heimautomatisierung noch relativ jung ist, hat sich bisher kein Standard für die Schnittstellen und die Kommunikationsprotokolle der einzelnen Komponenten wie Sensoren und Aktuatoren sowie die Regelformulierung durchgesetzt [BH13].

Beispiel Alice hat ihre Wohnung vor wenigen Jahren aufgerüstet und eine Reihe von Sensor- und Aktuatorinstallationen vorgenommen, welche jedoch mittels eines proprietären Kommunikationsprotokolls kommunizieren. Nun möchte Alice in ihre Wohnung zusätzlich noch Aktuatoren einbauen, mit Hilfe derer die Fenster automatisch gekippt bzw. wieder geschlossen werden können. Jedoch sind die Steuerbefehle für die neuen Aktuatoren nicht zu dem Kommunikationsprotokoll der alten Komponenten kompatibel. Dies muss bei Definition von Verhaltensregeln berücksichtigt werden.

Alice möchte eine neue Regel für die Kippaktuatoren definieren: „Sofern die Fenster mindestens 10 Minuten offenstanden, schließe alle Fenster, sobald die Außentemperatur unter 20 Grad sinkt.“ Hierbei vertreten ein oder mehrere Temperatursensoren die Rolle der Produzenten und alle Kippaktuatoren die Rolle der Konsumenten. Da Konsument und Produzent aufgrund unterschiedlicher Kommunikationsprotokolle und Schnittstellen nicht direkt miteinander kommunizieren können, muss zwischen ihnen vermittelt werden. Im Rahmen der Vermittlung müssen die unterschiedlichen Kommunikationsprotokolle berücksichtigt sowie das proprietäre Datenformat des Temperatursensors konvertiert werden, damit der Wert später als Teil der Verhaltensregel ausgewertet werden kann (vgl. Abbildung 5.1).

Der Auswertungsteil der Verhaltensregel entspricht einer ereignisbasierten Abfrage, die kontinuierlich auf Datenströme angewendet wird. Der andere Teil der Verhaltensregel bezieht sich auf die Aktion, die bei Eintreten eines bestimmten Ereignisses ausgeführt werden soll: die Steuerung der Aktuatoren. Diese bedürfen eines speziellen Befehls um ein Fenster zu schließen. Dieser Befehl muss als Resultat einer erfolgreichen Regelauswertung an die Konsumenten geschickt werden.

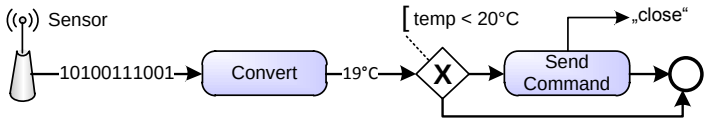


Abbildung 5.1.: Anwendungsbeispiel Smart Home

Sobald Alice die Verhaltensregel bei einem Vermittler registriert hat, wird deren Auswertungsteil fortan bei neu eintreffenden Sensormesswerten evaluiert. Bei erfolgreicher Evaluation führt der Vermittler den Aktionsteil der Verhaltensregel aus und konstruiert somit ein Ergebnis, das schließlich an die Aktuatoren übermittelt wird. Dieses Ergebnis repräsentiert den Befehl zum Schließen des Fensters, kann jedoch auch als eine, wenn auch sehr einfache und abstrakte, kontextbasierte Sicht verstanden werden, da sie das Ergebnis einer Reihe von Verarbeitungsschritten nach Maßgabe von Alice bzw. der Kippaktuatoren auf Kontextdaten ist und genau die für die Aktuatoren relevanten Informationen in einer adäquaten Repräsentation enthält.

5.2.2. Aktivitätsüberwachung

Um älteren, alleinstehenden Menschen ein möglichst langes und selbstbestimmtes Leben im eigenen Zuhause zu ermöglichen, bedarf es gewisser Überlegungen, damit z.B. bei einem Unfall automatisch ein Notruf abgesetzt wird, um Angehörige sowie Ersthelfer zu benachrichtigen.

Hierfür ist es erforderlich, die Aktivität einer Person zu überwachen, um z.B. einen Sturz zu registrieren. Es gibt eine Reihe von Möglichkeiten dieses zu realisieren. Beispielsweise existieren druckempfindliche Fußböden mit Hilfe derer die Lage von Personen detektiert werden kann. Auch können mit Hilfe

spezieller Vitalsensoren Blutdruck, Atmung, der Glukosegehalt des Bluts etc. überwacht werden. Auf diese Weise ist es möglich, einerseits auf die Aktivität des Bewohners und andererseits auf das körperliche Befinden zu schließen. Allerdings sind derartige Sensoren teuer und haben daher noch keine weite Verbreitung gefunden.

Im Rahmen dieses Anwendungsbeispiels soll daher eine kostengünstige Lösung für die Mutter von Alice Verwendung finden, die im Wesentlichen aus einem kleinen Computer (oder einem Telefon) mit Internet-Zugang sowie einem Armband besteht, in welches ein Beschleunigungssensor und ein damit verbundener WLAN-Adapter eingewebt sind. Das Armband erhebt kontinuierlich Beschleunigungsdaten und sendet diese zusammen mit allen zur Zeit vom WLAN-Adapter empfangenen Signalstärkeindikatoren der umliegenden WLAN-Zugangspunkte zur Auswertung an den Computer.

Dort werden die Beschleunigungsdaten zunächst geglättet und anschließend klassifiziert. Deutet das Beschleunigungsmuster dabei auf einen Sturz hin, ohne dass innerhalb weniger Sekunden das Aufstehen der Person (eine vertikale Beschleunigung) festgestellt wird, soll anhand der WLAN-Signalstärkeindikatoren der aktuelle Aufenthaltsort des Armbands bestimmt und anschließend die Rettungszentrale sowie die Angehörigen mittels eines synthetisierten Sprachanrufs über Voice-over-IP alarmiert werden (vgl. Abbildung 5.2).

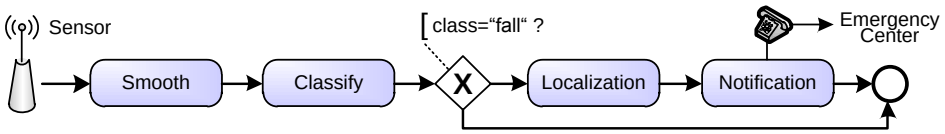


Abbildung 5.2.: Anwendungsbeispiel Aktivitätsüberwachung (aus (BL12))

In diesem Anwendungsszenario stellt das Armband den Produzenten dar. Um die Herstellungskosten niedrig zu halten, enthält dieser keine eigene Programmlogik, sondern lediglich Standardbauelemente. Die Verarbeitungslogik liegt auf der Seite eines Vermittlers, welcher durch den kleinen Computer (oder das Telefon) repräsentiert wird. Dort erfolgt das Glätten und Klassifizieren der Beschleunigungsdaten sowie ggf. die Positionsbestimmung anhand der WLAN-Signalstärkeindikatoren und das Benachrichtigen der Rettungszentrale und Angehörigen, welche die Konsumenten darstellen. Da keine besonderen Anforderungen an diese gestellt werden sollen, bedarf es einer Möglichkeit über herkömmliche Kommunikationswege, z.B. per Telefon, mit diesen in Verbindung zu treten. Eine automatisch generierte Nachricht, z.B. „Erna Müller ist in der Seilerstrasse 42 in ihrer Küche gestürzt und bittet um Hilfe“, welche mittels Sprachsynthese vom Computer vorgelesen werden kann, stellt eine kontextbasierte Sicht dar, da sie den Konsumenten in geeigneter Art und Weise relevante Informationen über eine Situation präsentiert.

5.2.3. Transportketten

Ein Schiff auf dem Weg von China nach Hamburg hat eine Ladung moderner Elektrofahrräder an Bord. Aufgrund der neuen Bauart sind deren Batterien relativ empfindlich gegenüber extremen Umwelteinflüssen. Insbesondere hohe Temperaturen können die Akkus nicht nur beschädigen, sondern bergen auch Explosions- und Verätzungsgefahr durch auslaufende Batteriesäure. Aus diesem Grund reisen die Fahrräder in klimatisierten *Smart Containern* [CS05], welche kontinuierlich die Temperatur im Inneren überwachen und in stündlichen Abständen an die Brücke des Schiffs melden.

Der Auftraggeber in Hamburg möchte sofort benachrichtigt werden, sobald ein Container ausfällt und die Akkus der Fahrräder überhitzen, da aufgrund der großen Nachfrage und langer Lieferzeiten rechtzeitig eine Ersatzlieferung bestellt und die Kunden der entsprechenden Räder über die Verzögerung informiert werden müssen. Daher registriert der Auftraggeber eine ereignisbasierte Abfrage beim System, welche bei Überschreiten eines festgelegten Temperaturgrenzwertes ein Ergebnis liefern soll. Dieses soll nach Maßgabe des Auftraggebers einerseits eine Visualisierung des Temperaturverlaufs der letzten sieben Tage zur Abschätzung des Handlungsbedarfs enthalten sowie andererseits die Stammblätter aller betroffenen Kunden.

Die Informationen, die der Smart Container liefert, sind die aktuelle Temperatur sowie die eindeutige Container-ID. Um nun die gewünschte kontextbasierte Sicht zu erzeugen, müssen vermittlerseitig mehrere Verarbeitungsschritte durchlaufen werden (siehe Abbildung 5.3):

- Anhand der Container-ID müssen die Temperaturmessungen der letzten sieben Tage aus einer Datenbank abgerufen und anschließend in einem Graphen visualisiert werden.

- Ebenfalls anhand der Container-ID müssen die Adressen von EPC Information Services (EPCIS) [EPC07] über den Object Naming Service (ONS) [EPC08] ermittelt werden[2], in denen weitere Informationen über die im Container enthaltenen Fahrräder gespeichert sind.

- Anhand jeder einzelnen Fahrrad-ID muss das Stammblatt des jeweiligen Kunden aus dem EPCIS des Auftraggebers abgerufen werden.

- Alle aggregierten Informationen müssen schließlich dem Auftraggeber übermittelt werden.

Hier wird deutlich, dass ausgehend von der ursprünglichen Information „Container X meldet Temperatur Y“, eine Reihe von Verarbeitungsschritten notwendig sind, um dem Kunden eine geeignete kontextbasierte Sicht zu präsentieren.

[2] EPCIS und ONS sind Verzeichnisdienste aus den EPCglobal bzw. GS1-Standards für das Internet der Dinge (vgl. Abschnitt 7.1.4.2).

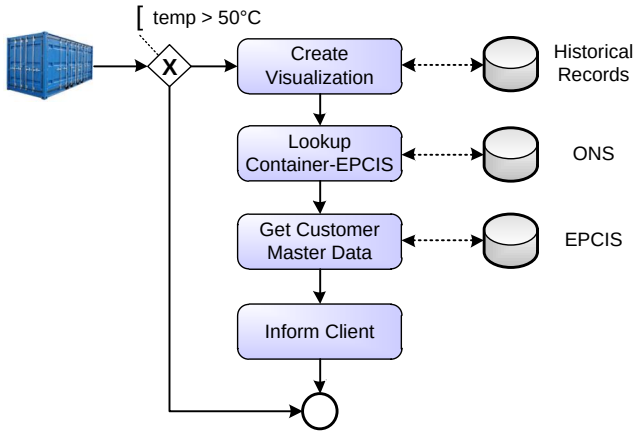


Abbildung 5.3.: Anwendungsbeispiel Transportkette

5.2.4. Weitere Anwendungsdomänen

Obige Anwendungsbeispiele stellen nur einen Ausschnitt der Einsatzmöglichkeiten dar und sollen im Rahmen dieser Arbeit das Konzept der Vermittlung kontextbasierter Sichten veranschaulichen. Für den Einsatz in der Praxis empfehlen sich, neben obigen Szenarien, noch eine Reihe weiterer Domänen, die an dieser Stelle nur kurz Erwähnung finden sollen.

Emergency Response Bei Naturkatastrophen oder schwerwiegenden Unfällen bedarf es der Koordination einer Vielzahl Beteiligter, deren Aufgaben entsprechend der Situation priorisiert und untereinander abgestimmt werden müssen. Dabei spielen insbesondere die Erfassung der Situation sowie in Abhängigkeit davon die Ausführung von Notfallreaktionsplänen (engl. *Emergency Response Plan*) eine wichtige Rolle. Dies lässt sich durch Vermittlung kontextbasierter Sichten dahingehend unterstützen, dass Sensordaten (und auch manuelle Eingaben) von einem Vermittler erfasst und in Bezug auf situationsabhängige Verhaltensregeln ausgewertet werden. Bei Eintreten bestimmter Situationen, z.B. Herzversagen eines Unfallopfers, muss das System proaktiv eine Reihe von Aktivitäten anstoßen und entsprechende Ersthelfer mit relevanten Informationen versorgen. Im Gegensatz zu obigem Anwendungsbeispiel der Aktivitätsüberwachung sind hierbei sehr viel mehr Produzenten und Konsumenten beteiligt, was eine höhere Systemlast auf Seiten eines Vermittlers erzeugt. Hieraus folgt, dass einzelne Ausführungsschritte ggf. vom Vermittler priorisiert werden müssen. Außerdem bedarf es Gruppen- und Rollenmodellen, um von einzelnen Personen bezüglich ihrer Profession abstrahieren zu können.

Umweltbeobachtung In zunehmendem Maße werden Sensornetze auch zur großflächigen Umweltbeobachtung eingesetzt. Einerseits, um langfristige Trends erkennen, andererseits auch, um rechtzeitig vor Katastrophen warnen zu können. Zu den bereits realisierten Anwendungen gehören

u.a. Tsunami-, Lawinen-, Waldbrand-, Erdbeben- und Vulkanausbruch-Frühwarnsysteme, sowie beispielsweise Systeme zur Beobachtung von Flussläufen und Gewässern, Luft- und Bodenverschmutzung sowie Besucher- und Verkehrsströmen [PSM+03, MG04, APM05, MRT06, FMF09, LCT05, MTS+06, DS08, BEJ+11]. Auch hier bedarf es der Erfassung und ggf. Persistierung großer Mengen an Sensordaten, die unterschiedliche Arten von Produzenten erheben, der Suche nach komplexen Mustern in diesen Daten sowie der situationsabhängigen Ausführung von Reaktionsplänen.

Gefahrenabwehr in Rechnernetzen In Firmen und Privathaushalten finden sich heutzutage eine Vielzahl vernetzter Geräte (Computer, Smartphones, Telefonanlagen, Mediengeräte und zunehmend auch Haushaltsgeräte wie Stromzähler, Waschmaschinen und Küchengeräte). Dies erlaubt potentiellen Angreifern (Viren, Würmern, Trojanern und Kriminellen) weitreichende Möglichkeiten über Schwachstellen einzelner Geräte in das Netz einzudringen. Sogenannte *Intrusion Detection*-Systeme sollen Eindringlinge aufspüren und verhindern, dass größerer Schaden entsteht [Bac00]. Allerdings sind je nach Netzkomplexität und Anzahl zu überwachender Geräte häufig besondere Infrastrukturkomponenten (spezielle Hardware oder zusätzliche Verarbeitungsressourcen) erforderlich, um die Menge an Informationen nach schädigenden Zugriffsmustern zu durchsuchen. Insbesondere für private Haushalte und kleine Firmen ist diese Investition einen möglichen Nutzen nicht wert. Auch hierfür eignet sich das Konzept der Vermittlung kontextbasierter Sichten dahingehend, dass ein Vermittler seine Dienste nach Bedarf über mehrere Knoten im Netzwerk verteilen können soll, so eine dezentrale Verarbeitung und Überwachung erlaubt und im Falle eines erkannten Eindringlings einerseits durch Ausführung entsprechender Aktionen eine zeitnahe Begrenzung des Schadens erlaubt und andererseits den Benutzer durch entsprechend aufbereitete Sichten über den Systemzustand informieren kann.

Smart Factory Als *Smart Factory* werden intelligente Produktionsumgebungen bezeichnet, in denen eine höhere Flexibilität und damit einhergehend die Reduzierung von Komplexität, Kosten und Produktionsdauer erzielt werden sollen [WJ03]. Dies soll durch den Einsatz intelligenter Maschinen und Kommunikationsinfrastrukturen erfolgen, mit Hilfe derer eine Produktionsüberwachung und ggf. ein Eingriff in die Produktion in nahezu Echtzeit ermöglicht wird. Die so gewonnenen Daten und Informationen dienen weiterhin als Entscheidungsgrundlage zur Optimierung der Produktions- und Geschäftsprozesse. Auch hier kann die Vermittlung kontextbasierter Sichten unterstützen: Auf der einen Seite stehen eine Vielzahl heterogener Sensoren, die große Mengen an Kontextdaten erheben. Auf der anderen Seite gibt es unterschiedliche Arten von Konsumenten (z.B. Maschinen, Techniker, Manager), welche mit jeweils eige-

nen Anforderungen an die Präsentation des Kontextes unterschiedliche Reaktionen auf bestimmte Situationen veranlassen können möchten.

Internet des Wissens Versteht man Daten als rein symbolische Abbildungen von Sachverhalten und Informationen als organisierte und strukturierte Daten, dann entsteht Wissen z.B. durch die logisch-funktionale Verknüpfung von Informationen mehrerer Quellen, durch Kontextualisierung, Erfahrung oder Lernprozesse [RH08]. Die Vision des *Internet des Wissens* skizziert entsprechend eine Verknüpfung des World Wide Webs, des Internet der Dinge, des Sensor Webs und des Semantic Webs [JS12, Tan13]. Es ist ein Überbegriff für riesige Datenmengen aus denen in intelligenter Art und Weise Wissen geschlussfolgert werden kann. In dieser Vision liegt es nicht fern, einen Dienst zu nutzen, der einem zu einer gestellten Frage bei der Verknüpfung der richtigen Informationsquellen unterstützt, also eine Art „Suchmaschine“. Unter Verwendung einer solchen Maschine könnte beispielsweise ein Weinliebhaber auf Idee kommen eine Antwort auf die Frage zu finden, wie der Temperaturverlauf zwei Wochen vor der Lese seines Eisweins war. Hier müssen mehrere Quellen miteinander verknüpft werden, i) die Einkaufshistorie, um herauszufinden welcher Wein überhaupt gemeint ist, ii) das Internet der Dinge, um Hersteller und damit das Weinanbaugebiet ausfindig zu machen, und schließlich iii) das Sensor Web, welches Informationen über die Wetterhistorie des Weinanbaugebiets vorhält. Die Vermittlung kontextbasierter Sichten kann dahingehend unterstützen, als dass sie die einzelnen Quellen auffinden und die gewünschten Daten abrufen kann. Mittels semantischer Abbildungen und Ableitungen werden idealerweise die den Daten anhaftenden Konzepte geschlussfolgert, um schließlich eine sinnhafte Verknüpfung der Informationen zu ermöglichen und dem Weinliebhaber zu präsentieren.

Quantified Self Unter dem Begriff *Quantified Self* versteht man eine Bewegung von Menschen, die aus unterschiedlichsten Interessen ihren Alltag durch eine Reihe von Sensoren aufzeichnen lassen. Insbesondere der Aspekt der Bewusstmachung von Verhalten und ggf. eine Veränderung desselben stellt für viele Menschen eine wesentliche Motivation dar [Wik12d]. Hierfür werden beispielsweise Aktivitätssensoren (z.B. Schrittzähler), Schlafmonitore, digitale Waagen, Blutdruck- und Herzfrequenzmessgeräte etc. verwendet. Die gesammelten Messwerte werden gespeichert, ausgewertet und ggf. kann auch in bestimmten Situationen eine Systemreaktion erfolgen, z.B. beim Lauftraining ein Warnhinweis, dass die Belastung zu hoch wird. Die Vermittlung kontextbasierter Sichten kann helfen, den Prozess der Erhebung, Speicherung und Aufbereitung zu automatisieren und in bestimmten Situationen kontextabhängige Reaktionen auszulösen.

5.3. Anforderungsanalyse

Nachdem nun anhand von Anwendungsbeispielen der praktische Nutzen der Vermittlung kontextbasierter Sichten aufgezeigt wurde, soll im Folgenden eine Anforderungsanalyse für den Prozess der Aufbereitung und Vermittlung kontextbasierter Sichten durchgeführt werden. Diese Anforderungsanalyse bildet die Grundlage für die Identifikation von Teilproblemen und Lösungsansätzen in Abschnitt 5.4 und soll aus zwei Perspektiven erfolgen:

Top Down Bei der *Top Down-Analyse* werden Anwendungsbeispiele betrachtet und anhand der Anforderungen dieser konkreten Beispiele wird ein Katalog aufgestellt. Auf diese Weise kann sichergestellt werden, dass nur praktisch sinnvolle Anforderungen erhoben werden, jedoch besteht die Gefahr, dass diese zu sehr auf eine Domäne beschränkt und nicht auf andere Anwendungsszenarien übertragbar sind.

Bottom Up Die *Bottom Up-Analyse* betrachtet keine konkreten Anwendungen, sondern stattdessen den Gegenstandsbereich. Hierbei steht die Frage im Mittelpunkt, was alles möglich ist. Dieser Ansatz ist also in gewissem Maße visionär, da er sich nicht am konkreten Bedarf existierender Szenarien orientiert, sondern neue Szenarien ermöglichen soll. Allerdings besteht die Gefahr eine Reihe von Anforderungen zu erheben, die keinen praktischen Nutzen aufweisen.

Die im Folgenden aufgeführten Anforderungen sind das Resultat einer Top Down-Analyse anhand konkreter Anwendungsbeispiele, mit Hilfe derer verpflichtende Anforderungen zur Realisierung eines minimalen Vermittlungsprozesses erhoben wurden, sowie einer anschließenden Bottom Up-Analyse, welche zusätzliche, optionale Anforderungen aufzeigt. Die Bottom Up-Analyse gründet sich einerseits auf einer durchgeführten Literaturrecherche sowie andererseits auf eigenen Überlegungen und Schlussfolgerungen.

5.3.1. Funktionale Anforderungen

Die folgenden funktionalen Anforderungen beschreiben die Funktionen, die notwendig bzw. wünschenswert sind, um einen Vermittlungsprozess zu realisieren. Sie gehen auf die Frage ein, *was* das System leisten soll [RR06]. Dabei lassen sich Kernfunktionen von optionalen Funktionen unterscheiden. Letztere sind entsprechend gekennzeichnet und stellen die wünschenswerten Funktionalitäten dar, die für eine Reihe von Anwendungen einen Mehrwert bringen, jedoch zum Beispiel aufgrund geringer Ausführungsressourcen verzichtbar sind. In Anlehnung an die Dekomposition des Gesamtprozesses in verschiedene Aufgabenbereiche (siehe Abschnitt 4.4ff.) werden auch die funktionalen Anforderungen im Folgenden entsprechend untergliedert.

5.3.1.1. Datenübermittlung

Die funktionalen Anforderungen an die Datenübermittlung beziehen sich auf die Unterstützung unterschiedlicher Kommunikationstechnologien und damit einhergehend auf eine geeignete, technologieneutrale Adressierung sowie unterschiedliche Arten der Nachrichtenzustellung und den Umgang insbesondere mit mobilen Klienten. Dabei sollen sowohl Produzenten und Konsumenten als auch Vermittler berücksichtigt werden.

Unterschiedliche Kommunikationstechnologien Wie bereits in Abschnitt 4.4.2 erwähnt, existieren eine Vielzahl unterschiedlicher Standards für die Datenkommunikation. Insbesondere im Bereich der Sensoren konkurrieren mehrere Standards und Technologien auf unterschiedlichen Ebenen des ISO/OSI-Modells (z.B. *IEEE 802.15* [Ins13], *6loWPAN* [Int10c], *ZigBee* [Zig12] etc.). Da diese oftmals unterschiedliche Aspekte und Anforderungen adressieren, wird sich in naher Zukunft auch keine der Technologien als Standard für alle Einsatzbereiche durchsetzen können [HKS08]. Auch sind die meisten Technologien auf die lokale Kommunikation beschränkt. In Netzen größerer Skalen haben sich gänzlich andere Technologien (z.B. *(Mobile) IPv4*, *IPv6* [Com06, Rot05], *HTTP* [Int99], *SOAP* [W3C07] etc.) etabliert. Daher muss im Rahmen des Vermittlungsprozesses zwischen unterschiedlichen Kommunikationstechnologien übersetzt werden (z.B. von ZigBee nach IP). Zusätzlich muss noch bedacht werden, dass die meisten heutzutage verwendeten Technologien eine Maschine-zu-Maschine-Kommunikation fokussieren. Unter Berücksichtigung von Menschen als Kommunikationsendpunkte müssen daher noch weitere Technologien, z.B. der *Short Message Service* (SMS) oder *Text-2-Speech*-basierte Benachrichtigungen per Telefon, unterstützt werden.

Logische Adressierung Einher mit der Unterstützung unterschiedlicher Kommunikationstechnologien geht die Forderung einer einheitlichen Adressierung, die nicht von einer bestimmten Kommunikationstechnologie abhängig ist. Während Bluetooth beispielsweise eine Adressierung mittels der *Medium Access Control* (MAC) -Adressen der Geräte vornimmt [Blu10], abstrahieren normale Desktop-Anwendungen von dieser technischen Ebene und verwenden stattdessen meist IP-Adressen sowie eine Port-Nummer (also TCP-Adressen) [Com06]. Möchte man unabhängig von den jeweils verwendeten Kommunikationstechnologien der Produzenten und Konsumenten sein, bedarf es entsprechend einer rein logischen Adressierung [HKS08]. Hierbei erhalten alle Endpunkte (Produzenten, Konsumenten und Vermittlungsdienste) zusätzlich zu ihrer physischen Adresse noch eine universell eindeutige Kennung (engl. *Universally Unique Identifier*, UUID) mittels derer sie adressiert werden können. Außerdem kann eine logische Adressierung den Grad der Kopplung verringern, da man z.B. zur Ausführung von Verarbeitungsdiensten nicht die konkrete ausführende Plattform, sondern nur noch den Dienst adressieren muss [HKS08]. Hierfür bedarf es wiederum eines Verzeichnisses, welches eine Um-

setzung von logischen in konkrete (physische) Adressen vornimmt (und umgekehrt).

Push- und Pull-basierte Kommunikation (optional) Unter push-basierter Kommunikation wird im Allgemeinen die proaktive Nachrichtenzustellung verstanden [DGD05, TS07]. Ein Produzent initiiert beispielsweise einen Kommunikationsakt und übermittelt von sich aus seine Daten an einen Vermittler, sobald ein neuer Messwert erhoben wurde. Gleichsam kann ein Vermittler proaktiv einen Konsumenten informieren, sobald das Ergebnis einer Abfrage vorliegt. Eine solche push-basierte Nachrichtenübermittlung erfolgt also zeitnah nach Auftreten eines Ereignisses und ist somit in den meisten Fällen zu bevorzugen. Jedoch erfordert dies, dass die aktuelle physische Adresse des Kommunikationsendpunkt bekannt und dieser erreichbar ist. Insbesondere in Systemen mit mobilen Teilnehmern ist dies jedoch nicht immer der Fall, da den Teilnehmern oftmals nur temporäre Endpunktadressen zugeordnet werden, die zudem einem privaten Adressbereich entstammen können, sodass eine Verbindungsaufnahme aus öffentlichen Netzen (z.B. dem Internet) technisch nicht ohne Weiteres möglich ist [CDK12, VHP06]. Gleiches gilt auch für Endpunkte, die durch Schutzmechanismen (z.B. Firewalls) vor unaufgeforderten Verbindungsversuchen geschützt sind. Unter solchen Bedingungen bietet sich eine pull-basierte Nachrichtenübermittlung an. Hierbei fragen die eigentlichen Empfänger einer Nachricht periodisch bei potentiellen Absendern an, ob Nachrichten für sie vorliegen. Ein Konsument z.B. fragt in regelmäßigen Abständen bei einem Vermittler an, ob bereits ein Ergebnis für seine Abfrage vorliegt. Da in diesem Fall der eigentliche Empfänger einer Nachricht den Kommunikationsakt initiiert, umgeht man die oben beschriebenen, technischen Restriktionen. Allerdings führt dies zu einem erhöhtem Nachrichtenaufkommen durch wiederholtes Abfragen und einer insgesamt höheren Latenz, da Nachrichten auf Seite des Senders solange zwischengespeichert werden müssen, bis der Empfänger sie abholt. Da je nach Anwendungsfall eine der Kommunikationsarten zu bevorzugen ist, sollte ein Vermittler daher beide Möglichkeiten anbieten.

Mobilitätsunterstützung (optional) Mobile Endgeräte auf Seiten der Produzenten und Konsumenten bedürfen einer besonderen Unterstützung, da sie oftmals nur temporär mit der Infrastruktur konnektiert sind, die Verbindungen unvorhersehbar und jederzeit abbrechen können und häufige Wechsel der physischen Endpunktadressen berücksichtigt werden müssen [Rot05]. Mit Einführung einer logischen Adressierung ist es möglich, die Kommunikation zwischen Produzent und Vermittler sowie zwischen Vermittler und Konsument über eine zwischengeschaltete Mobilitätsverwaltung laufen zu lassen, welche die jeweils aktuelle physische Endpunktadresse eines Adressaten nachschlagen und bei dessen Nicht-Erreichbarkeit die zu versendenden Nachrichten zwischenspeichern kann. So können auch plötzliche Verbindungsabbrüche bei der Übertragung kompensiert und eine *At-Least-Once-Semantik* [CDK12] bei der

Nachrichtenzustellung realisiert werden, sodass eine Nachricht im Laufe der Zeit mindestens einmal erfolgreich übertragen wird, sofern der Endpunkt erreichbar ist.

5.3.1.2. Datenverarbeitung

Die funktionalen Anforderungen an die Datenverarbeitung betreffen vornehmlich den Vermittler. Klienten, also Produzenten und Konsumenten, sollen nach eigener Maßgabe die Verarbeitung von Sensordaten bzw. die Aufbereitung kontextbasierter Sichten bestimmen können. Hierzu können sie auf vermittlerseitig angebotene Verarbeitungsfunktionen zurückgreifen oder eigene, benutzerdefinierte Dienste spezifizieren. Komplexe Verarbeitungsketten können durch Orchestrierung von Diensten realisiert werden. Das Auffinden angebotener Dienste über ein entsprechendes Verzeichnis hilft den Klienten, eine Übersicht über das Dienstangebot eines Vermittlers zu erlangen.

Klienten-bestimmte Kontextdatenverarbeitung Zur Erstellung kontextbasierter Sichten, welche ein Konsument direkt weiterverarbeiten kann, bedarf es in den meisten Fällen einer Vorverarbeitung der Kontextdaten, um sie den Erfordernissen des Konsumenten entsprechend aufzubereiten. Da ein Vermittler per se keine Kenntnis der Erfordernisse eines Konsumenten hat und darüber hinaus verschiedene Konsumenten unterschiedliche Anforderungen an dieselben Daten haben können, muss jeder Konsument einem Vermittler mitteilen, wie die Kontextdaten als eine kontextbasierte Sicht aufzubereiten sind. Doch die Verarbeitung von Kontextdaten kann in vielen Anwendungsfällen nicht nur nach Maßgabe der Konsumenten sinnvoll sein, auch Produzenten können ein Interesse an der Verarbeitung der von ihnen übermittelten Daten haben, bevor sie zum Beispiel in einer Datenbank gespeichert oder dem Datenstrommanagementsystem zum Abgleich mit ereignisbasierten Abfragen übergeben werden. So kann zum Beispiel vermieden werden, dass Verarbeitungsschritte, die ohnehin von Konsumenten ausgeführt werden müssten, z.B. die Umwandlung von Formaten oder eine Dekompression, mehrfach ausgeführt werden. Auch können Produzenten durch Vorverarbeitung oder Anreicherung ihrer Daten einen Mehrwert für die Konsumenten erzeugen, den sie, z.B. aufgrund fehlender Ressourcen, selbst nicht erbringen können.

Angebot typischer Verarbeitungsfunktionen (optional) Zur Erstellung kontextbasierter Sichten kann es notwendig sein, auf den von Sensoren oder Datenbanken gelieferten Kontextdaten weitere Verarbeitungsschritte auszuführen, welche die ursprünglichen Daten zum Beispiel mit weiteren Informationen anreichern und in eine geeignete Präsentation überführen. Natürlich lassen sich hier, je nach Anforderungen von Produzenten und Konsumenten, beliebige Verarbeitungsschritte vorstellen. Für die Verarbeitung speziell von Kontextdaten gibt es jedoch eine Reihe typischer Verarbeitungsfunktionen, die

häufig Verwendung finden [CKDB06, FJK+05] und daher als integraler Bestandteil des Vermittlungsprozesses angesehen werden können. Exemplarisch sollen folgende Funktionen genannt werden (vgl. auch Abschnitt 2.3):

Reduktion Zu diesen Funktionen gehören z.B. das Filtern syntaktisch inkorrekter Nachrichten, die Duplikaterkennung und die verlustfreie Kompression.

Aggregation Derartige Funktionen fassen mehrere Messwerte oder Zeitreihen zusammen, hierzu gehören z.B. Funktionen zur Mittelwertbildung, Min/Max-Funktionen sowie einfache Summen- und Zählfunktionen.

Konversion Konversionsfunktionen (auch Konvertierungsfunktionen) wandeln Daten sinnerhaltend um. Beispiele hierfür sind die *Einheitenkonversion* (z.B. Grad Celsius in Grad Fahrenheit), *Formatkonversion* (JSON nach XML oder GIF nach PNG), *Transkodierung* (verlustbehaftete Umwandlung, z.B. von WAV nach MP3 oder von MPEG-2 nach MPEG-4) oder *Transkriptionsfunktionen* (z.B. Umwandlung von Sprache in Text, die Umschreibung von Notenschriften in der Musik oder die Übertragung eines Textes in andere Alphabete).

Translation Zu diesen Funktionen gehören beispielsweise die Übersetzung von einer Sprache in eine andere Sprache (z.B. vom Deutschen ins Englische) oder Funktionen zur mathematischen Verschiebung bzw. Abbildung (z.B. zur Berechnung der zurückgelegten Distanz aus einer Anzahl von Schritten).

Augmentation Funktionen zur Anreicherung von Daten mit weiteren Informationen, z.B. aus externen Quellen (beispielsweise die Abfrage von Produktinformationen zu einer gegebenen ISBN-Nummer oder von Wetterinformationen zu einem bestimmten Geopunkt).

Ableitung Funktionen zur Ableitung von (weiteren) Informationen aus gegebenen Daten mittels Inferenzmechanismen (z.B. das Schlussfolgern der Aktivität eines Nutzers anhand gemessener Beschleunigungsdaten).

Archivierung Funktionen zum Festschreiben von Daten, z.B. in einer Datenbank oder einem Dateisystem.

Kommunikation In diese Klasse fallen ganz allgemein Funktionen zur Kommunikation. Beispielsweise könnte ein Klient mit Hilfe solcher Funktionen Zwischenergebnisse eines Vermittlungsprozesses beziehen oder im Rahmen des Prozesses Nachrichten (z.B. Anweisungen) an andere Entitäten verschicken wollen.

Sonstige Funktionen Zu dieser Klasse gehören eine Reihe weiterer, typischer Funktionen, die sich nicht in obige Klassifizierung einordnen lassen. Hierzu gehören beispielsweise das Ver- und Entschlüsseln von Daten, das

Sortieren oder höherwertige Funktionen zur Gesichts- oder Objekterkennung in Bildern, das Erstellen akustischer Fingerabdrücke, das Erzeugen von Visualisierungen aus Zeitreihen etc.

Diese Verarbeitungsfunktionen stellen lediglich eine Auswahl dar, die sinnvollerweise einen integralen Bestandteil des Systems darstellen, da sie sowohl von Produzenten als auch Konsumenten häufig nachgefragt werden.

Benutzerdefinierte Verarbeitungsfunktionen (optional) Da, wie oben erwähnt, nur eine Auswahl sinnvoller Verarbeitungsfunktionen durch einen Vermittler bereitgestellt werden kann, bedarf es einer Möglichkeit benutzerdefinierte Verarbeitungsfunktionen in den Vermittlungsprozess integrieren zu können. Dies kann im Wesentlichen auf zwei Arten erfolgen:

Externe Dienstreferenzen Die Erstellung einer kontextbasierten Sicht setzt sich aus einer Sequenz einzelner Verarbeitungsschritte zusammen. In jedem Verarbeitungsschritt ruft ein Vermittler einen entsprechenden Dienst zur Ausführung des Schrittes auf. Sofern er selbst entsprechende Dienste anbietet, können die Verarbeitungsschritte auch durch diesen ausgeführt werden. Durch Angabe von Dienstreferenzen kann ein Klient einen Vermittler jedoch auch anweisen, für die Ausführung eines Schrittes einen bestimmten externen Dienst aufzurufen, der die entsprechende Ausführungslogik bereitstellt.

Mobiler Code Die Referenzierung externer Dienste in einem Vermittlungsprozess hat den Nachteil, dass man sicherstellen muss, dass der entsprechende Dienst zum gewünschten Ausführungszeitpunkt erreichbar ist. Auch ist es in vielen Fällen nicht effektiv, große Datenmengen (z.B. Bilder) an einen externen Dienst zu übermitteln. Stattdessen kann es sinnvoll sein, einen benutzerdefinierten Dienst bzw. den Programmcode zur Ausführung an einen Vermittler zu übergeben, sodass dieser zum geeigneten Zeitpunkt den entsprechenden Dienst selbst instanziieren kann. Dies kann entweder direkt durch Übermittlung aller benötigten Bibliotheken erfolgen oder aber indirekt durch Angabe eines entsprechenden *Repositories*, welches Bibliotheken bei Bedarf zum Abruf bereitstellt.

Eine solche Injektion benutzerdefinierter Dienste erlaubt die Ausführung beliebiger Verarbeitungsschritte und kann somit die (möglicherweise mobilen oder ressourcenbeschränkten) Produzenten und Konsumenten weitergehend entlasten.

Orchestrierung von Verarbeitungsfunktionen Da die Verarbeitungsfunktionen eine starke Kohäsion aufweisen sollten, d.h. dass sie speziell für die Erbringung einer relativ eng abgegrenzten Aufgabe bestimmt sind, ist es in den meisten Fällen sinnvoll, komplexe Verarbeitungsketten (sogenannte *Workflows*) aus mehreren simplen Verarbeitungsschritten aufzubauen, welche in

festgelegter Reihenfolge ausgeführt werden. Eine solche Verknüpfung oder Komposition bezeichnet man auch als *Orchestrierung* [Pel03]. Diese ermöglicht es Produzenten und Konsumenten, die Verarbeitungsfunktionen nach ihren eigenen Maßgaben miteinander zu kombinieren. Beispielsweise könnte ein Konsument für das Ergebnis einer Wetteranfrage eine Einheitenkonversion von Grad Fahrenheit nach Grad Celsius, die Anreicherung der Sensordaten mit einer Prognose der nächsten zwei Tage aus einer externen Quelle und schließlich eine Konvertierung des Gesamtergebnisses in das XML-Format wünschen. Diese Verarbeitungskette umfasst nur drei sequentiell auszuführende Schritte, wobei eine Orchestrierung durchaus auch komplexere Ausführungsfolgen mit parallelen, konditionalen und asynchronen Dienstaufrufen enthalten kann. Hierfür ist eine entsprechende Beschreibungssprache zur Definition einer Orchestrierung erforderlich.

Auffinden angebotener Verarbeitungsfunktionen (optional) Wie bereits oben erwähnt, kann ein Vermittler typische Verarbeitungsfunktionen für Kontextdaten selbst anbieten. Damit ein Klient sich eine Übersicht über angebotene Funktionen bzw. Dienste verschaffen und ggf. entscheiden kann, ob eigene, benutzerdefinierte Dienste im Rahmen des Prozesses referenziert und ausgeführt werden müssen, bedarf es eines Verzeichnisses, welches alle angebotenen Funktionen führt. Ein solches Verzeichnis erlaubt es dem Klienten nicht nur, den korrekten Bezeichner eines Dienstes in seiner Workflow-Beschreibung zu verwenden, sondern zudem idealerweise auch die entsprechenden Ein- und Ausgabeparameter sowie ggf. Vor- und Nachbedingungen des Dienstaufrufs einzusehen.

Zustandslose/-behaftete Verarbeitungsdienste (optional) Für eine Reihe von Anwendungsfällen muss zwischen zustandslosen und zustandsbehafteten Diensten unterschieden werden. Der Aufruf eines zustandslosen Dienstes ist deterministisch, bei gleicher Eingabe erfolgt dieselbe Ausgabe. Für Dienste, die von unterschiedlichen Parteien genutzt werden, stellt dies den Normalfall dar. Bei zustandsbehafteten Diensten kann ein Aufruf hingegen den Zustand ändern und ein nachfolgender Aufruf kann bei gleicher Eingabe eine unter Umständen andere Ausgabe hervorrufen. Hierbei ist zu unterscheiden, ob ein zustandsbehafteter Dienst der Allgemeinheit oder dediziert nur einzelnen Klienten zur Verfügung steht, da hiervon die Anzahl prinzipiell verfügbarer Instanzen abhängt.

Ein typischer Anwendungsfall für Ersteres sind Vorhersage- oder Klassifizierungsdienste, die anhand der Eingaben lernen, um ihr internes Modell zu optimieren. Solche Dienste können in den meisten Fällen ohne Weiteres von unterschiedlichen Klienten genutzt werden, da die einzelnen Aufrufe nicht interferieren. Ein Beispiel hierfür wäre ein Dienst zur Wetterprognose in Hamburg, der anhand der Wetterdaten aller Sensoren in Hamburg sein internes Prognosemodell optimiert.

In anderen Anwendungsfällen ist es erforderlich, dass ein zustandsbehafteter Dienst dediziert einem bestimmten Klienten, und nur diesem, zur Verfügung steht. Ein entsprechender Anwendungsfall wäre ein Aggregationsdienst, welcher RFID-Leseereignisse räumlich benachbarter Lesegeräte sammelt, Duplikate filtert und die Ergebnisse aggregiert, bevor sie weiterverarbeitet werden. Von solchen Diensten können mehrere Instanzen mit jeweils eigenen Zuständen existieren.

5.3.1.3. Datenverwaltung

Die funktionalen Anforderungen bezüglich der Datenverwaltung betreffen ausschließlich den Vermittler. Einen Großteil möglicher funktionaler Anforderungen, z.B. Unterstützung von Transaktionen oder räumlich-zeitlicher Abfragen, zielen direkt auf die Wahl eines entsprechenden Datenmodells bzw. Datenbanksystems. Diese werden durch die Anforderung der Unterstützung unterschiedlicher Datenbanksysteme zusammengefasst bzw. abstrahiert. Zwar gibt es auch Datenbanksysteme, die explizit die Semantik von Daten unterstützen, jedoch wiegt diese Forderung so groß, dass sie ausdrücklich auch für (beliebige) andere Datenbanksysteme gelten soll.

Unterschiedliche Datenbanksysteme (optional) In Abschnitt 4.4.4 wurde erläutert, dass die Unterstützung verschiedener Datenmodelle und damit einhergehend unterschiedlicher Datenbanksysteme wünschenswert ist, da ein einzelnes Datenbanksystem niemals den Anforderungen aller Anwendungsszenarien gerecht werden kann. Neben dem weit verbreiteten relationalen Modell sollen so auch z.B. objektrelationale und nicht-relationale Modelle sowie räumlich-zeitliche oder probabilistische Datenbanksysteme eingesetzt werden können. Daraus ergibt sich direkt die Anforderung, dass Datenbanken zur Verwaltung von Prozessdaten (z.B. Zugriffs- und Abrechnungsdaten) und Metadaten (von Konsumenten, Produzenten, Messwerten etc.), die von nicht-optionalen Diensten im Rahmen des Vermittlungsprozesses zugegriffen werden, über eine generische Schnittstelle gekapselt werden müssen. Andernfalls wäre es notwendig Wissen über die Art der Datenbank und die von dieser verwendeten Abfragesprache in die Vermittlungsdienste zu integrieren, was eine engere Kopplung nach sich ziehen und bei Wechsel des Datenbanksystems eine Anpassung der Dienste erfordern würde. Abfragen von Konsumenten nach Zeitreihen hingegen sollen in einer beliebigen Abfragesprache formuliert werden können, sofern ein entsprechendes Datenbanksystem unterstützt wird.

Semantische Abbildungen und Ableitungen (optional) Datenbankabfragen bedürfen in den meisten Fällen einer genauen Kenntnis der Bezeichner von Attributen, welche den Daten implizit eine Bedeutung verleihen. Allerdings gibt es keine allgemeingültige Norm der Benennung, sodass die Bezeichner beim Anlegen von z.B. Schemata oder der Eintragung von Datensätzen prinzipiell beliebig gewählt werden können. Dies erschwert die Formulierung von Abfra-

gen seitens der Konsumenten. Die Abfrageformulierung sollte daher durch semantische Abbildungen und Ableitungen unterstützt werden, da sie eine Abstraktion von technischen und formalen Details erlauben. Mit Hilfe semantischer Abbildungen können beispielsweise Terme wie „Stunde“, „Freitags“ oder „Temperatur“ auf eine formale Syntax, z.B. von Zeitstempeln, abgebildet werden. Bei semantischen Ableitungen werden zudem Generalisierungen und Spezialisierungen von Konzepten berücksichtigt, sodass z.B. der Begriff „Hamburg“ zu einem geografischen Fixpunkt mit Längen- und Breitengradangaben und einem bestimmten Radius abgeleitet werden kann. Semantische Abbildungen und Ableitungen bedürfen eines semantischen Modells, welches die Abbildungs- und Ableitungsregeln festschreibt und eines Inferenzmechanismus, welcher die Abbildung bzw. Ableitung durchführt.

5.3.1.4. Prozessverwaltung

Die Prozessverwaltung umfasst im Wesentlichen administrative Funktionen, die im Rahmen des Vermittlungsprozesses der Überwachung, Fehlersuche, Optimierung und Sicherheit dienen. Alle nachfolgenden Anforderungen sind optional, da sie nicht in allen Szenarien oder in allen Phasen des Lebenszyklus eines Vermittlers sinnvoll genutzt werden können. Teils bedeuten sie zudem einen zusätzlichen Ressourcenaufwand für den Austausch von Kontroll- und Statusnachrichten, was in einer frühen Phase des Lebenszyklus, z.B. für das Testen und Debuggen von Diensten, hilfreich, in einem Produktionssystem jedoch unnötig sein mag.

Logging & Accounting (optional) Unter *Logging & Accounting* versteht man das Aufzeichnen von Vorgängen innerhalb eines Systems zur späteren Analyse [ED06, SHG+08]. Hierzu gehören beispielsweise die Aufzeichnung von Dienstaufrufen sowie der Zugriff auf Daten, aber auch Änderungen an der Systemkonfiguration. Die aufgezeichneten Daten bilden die Grundlage für eine Vielzahl an Aufgaben:

- Die Aufzeichnungen helfen bei der Suche nach Fehlern. Sie sind unverzichtbar für komplexere, verteilte Systeme in denen verschiedene Aktivitäten parallel auf unterschiedlichen Knoten ausgeführt werden.
- Die Analyse der Aufzeichnungen größerer Zeiträume bietet Entscheidungshilfen für administrative Maßnahmen. So lassen sich zum Beispiel Flaschenhälse erkennen, Stoßzeiten ermitteln, regelmäßige Verstöße gegen QoS-Vereinbarungen ausfindig machen etc.
- Durch Aufzeichnung der Zugriffe auf Daten und Dienste lassen sich Missbräuche aufdecken, welche entweder die Systemsicherheit gefährden (z.B. durch *Denial-of-Service*-Attacken) oder sensible Daten betreffen (z.B. wiederholter Versuch unbefugt auf persönliche Informationen zuzugreifen).

- Mit Hilfe der Zugriffsaufzeichnungen lassen sich auch Abrechnungen für den Fall, dass der Zugriff auf bestimmte Daten und Dienste kostenpflichtig angeboten wird, erstellen.

Logging-Daten müssen persistiert werden. Eine Schnittstelle zum Hinzufügen und Abrufen muss seitens eines entsprechenden Dienstes angeboten werden. Höhere Funktionen zum Erstellen von Abrechnungen oder der Analyse missbräuchlichen Zugriffs können auf diesem Dienst aufsetzen, sind jedoch optional.

Testing & Debugging (optional) Mit Hilfe von Test- und Debugging-Werkzeugen kann das Risiko reduziert werden, dass ein System nicht fehlerfrei funktioniert bzw. nicht den Anforderungen und Spezifikationen genügt [Cop04]. Hierfür können Testfälle für unterschiedliche Ebenen spezifiziert werden: angefangen auf der Ebene einzelner Methoden von Software-Bausteinen bis hin zu kompletten Systemtests, die das korrekte Zusammenspiel von Software-Bausteinen überprüfen. Während das Ziel eines Tests das Auffinden vorher unbekannter Fehler ist, versucht man mit Hilfe des Debugging fehlerverdächtige Programmabschnitte genauer zu analysieren bzw. detaillierte Informationen zur Analyse zusammenzutragen [MSB11]. Insbesondere in verteilten Systemen bedarf es einer besonderen Unterstützung durch geeignete Werkzeuge sowohl zum Testen als auch zum Debuggen, da die Generierung einer globalen Sicht auf das Gesamtsystem sehr großen Aufwand bedeutet [TS07].

Monitoring & Reporting (optional) Der Vermittlungsprozess erfordert das Zusammenspiel einer Reihe von Diensten, welche unter Umständen verteilt und parallel ausgeführt werden. Um eine optimale Prozessausführung gewährleisten zu können, bedarf es einer kontinuierlichen Beobachtung des Gesamtsystems (engl. *Monitoring*). Die Beobachtung kann sich dabei auf unterschiedliche Aspekte beziehen, beispielsweise auf das Auftreten von Fehlern, die Auslastung von Ressourcen, die Detektion von Angriffen auf das System oder aber auch auf die Einhaltung von QoS-Anforderungen. Werden Auffälligkeiten entdeckt, kann unter Umständen ein zeitnaher Eingriff erforderlich sein, weshalb eine Berichterstattung (engl. *Reporting*), z.B. an einen Administrator oder eine bestimmte Systemkomponente, geschehen muss, um ggf. geeignete administrative Maßnahmen einzuleiten. Zum Beispiel könnte ein Monitoring-Werkzeug eine wiederholte Verletzung von QoS-Vereinbarungen, bedingt durch eine zu hohe Auslastung einzelner Systemkomponenten, erkennen und daraufhin veranlassen, dass Dienste auf andere Ressourcen umverteilt oder weitere Ressourcen zur Verfügung gestellt werden.

Ressourcenverwaltung (optional) Damit ein Vermittlungsprozess effektiv durchgeführt werden kann, bedarf es ausreichender Ressourcen (insbesondere Verarbeitungs- und Speicherressourcen). Um die ausreichende Verfügbarkeit

von Ressourcen in einem verteilten System sicherzustellen, bedarf es der kontinuierlichen Beobachtung der Ressourcen (siehe *Monitoring & Reporting*) und einer entsprechenden Reaktion, sobald die Auslastung der Ressourcen zu groß oder zu gering ist. Bei zu geringer Last kann es aus verschiedenen Gründen (z.B. Kosten, Umweltschutz) sinnvoll sein, Ressourcen aus dem System zu entfernen. Bei zu hoher Last hingegen können - sofern der Vermittlungsprozess horizontal skalierbar ist (vgl. Abschnitt 5.3.2) - weitere Ressourcen alloziert oder Dienste zwischen existierenden Ressourcen umverteilt werden. Dies kann entweder manuell durch einen Administrator nach entsprechender Benachrichtigung oder automatisch durch das System selbst geschehen.

Zugriffskontrolle (optional) Sowohl Kontextdaten als auch Metadaten müssen vor unberechtigtem Zugriff geschützt werden. Auf der einen Seite können sie vertrauliche Informationen repräsentieren (z.B. den Aufenthaltsort einer Person), auf der anderen Seite können die Daten wertvoll sein und ggf. nur gegen ein Entgelt zur Verfügung gestellt werden. Das Gleiche gilt auch für Verarbeitungs- und Vermittlungsdienste. Auch diese können vertrauliche Informationen (z.B. einen besonderen Algorithmus) kapseln oder ggf. nur gegen ein Entgelt genutzt werden. Der Schutz vor unberechtigtem Zugriff muss daher sowohl für die (Meta-)Datenverwaltung als auch für die Datenverarbeitung und Prozessverwaltung durchgesetzt werden. Idealerweise erlaubt die Zugriffskontrolle zudem eine feingranulare Bestimmung zulässiger Zugriffe auf Basis von Rollen, Gruppen und einzelnen Entitäten.

5.3.1.5. Metadatenverwaltung

Metadaten werden für die Prozessausführung, z.B. zur Abrechnung einer Diensterbringung, das Testen und Debuggen, Leistungsoptimierungen etc. benötigt, aber vor allem unterstützen sie den Konsumenten bei der Abfrageformulierung. Mit ihrer Hilfe kann der Konsument einen Überblick einerseits über verfügbare Produzenten und deren Eigenschaften (z.B. deren Position oder die Namen der gemessenen Kontextattribute) und andererseits über zur Verfügung stehende Verarbeitungsdienste zur Erstellung kontextbasierter Sichten und deren Schnittstellenbeschreibungen bekommen.

Metadatenermittlung (optional) Wie in Abschnitt 4.4.6 beschrieben, gibt es zwei Möglichkeiten die Metadaten von Produzenten, Konsumenten und Diensten zu ermitteln: Die Metadaten werden entweder proaktiv von diesen Entitäten an einen Vermittler übermittelt oder ein Vermittler sucht im Netzwerk nach verfügbaren Produzenten, Konsumenten und angebotenen Diensten und fragt deren Metadaten ab. Während Ersteres durch eine einfache Registrierung geschehen kann, bedarf Letzteres, das sogenannte *Metadaten-Harvesting* (vgl. Abschnitt 4.4.6), einerseits geeigneter Methoden zum Auffinden von Klienten und Diensten und andererseits entsprechender Abfrageschnittstellen über welche die Daten abgerufen werden können.

Metadatenverwaltung In Abschnitt 4.4.6 wurde auch die Notwendigkeit einer Metadatenverwaltung aufgezeigt. Diese ist zuständig für das Persistieren und den Zugriff auf Metadaten der Konsumenten, Produzenten und deren Messwerte, Verarbeitungsdienste sowie auf Prozessmetadaten. Da sich Metadaten in der Regel relativ selten ändern, kann durch das Persistieren der Daten auf Vermittlerseite ein *Caching* realisiert werden, sodass wiederholte Abfragen nicht die Produzenten, Konsumenten und Dienste belasten. Das Hinzufügen neuer Datensätze, Änderungen an bestehenden Datensätzen und die Abfrage von Datensätzen müssen über eine generische Schnittstelle möglich sein. Das Hinzufügen oder Ändern kann entweder direkt von den betroffenen Entitäten (Produzenten, Konsumenten, Diensten etc.) erledigt werden oder automatisch durch die Metadatenermittlung erfolgen (siehe oben).

5.3.1.6. Datenabfrage

Die Datenabfrage gehört im Rahmen des Vermittlungsprozesses zu den wichtigsten Aufgabenbereichen, da hier die Schnittstelle zwischen Konsumenten und Vermittler liegt. Da die Anforderung verschiedenartige Abfragen zu ermöglichen, ggf. zu Benutzbarkeitsproblemen bei der Abfrageformulierung führen kann, bedarf es verschiedenster Unterstützungsformen um Konsumenten die Abfragestellung zu erleichtern.

Verschiedenartige Abfragen Wie in Abschnitt 4.4.7 erläutert, lassen sich im Wesentlichen drei verschiedene Abfragearten unterscheiden: i) ad-hoc Abfragen, ii) historische Abfragen und iii) ereignisbasierte Abfragen. Historische Abfragen müssen im Rahmen des Vermittlungsprozesses an ein Datenbankmanagementsystem (DBMS), welches entsprechende Daten vorliegen hat, delegiert werden. Ereignisbasierte Abfragen werden bei einem Datenstrommanagementsystem (DSMS) subskribiert und bleiben für gewöhnlich solange aktiv, bis sie explizit de-subskribiert werden. Und für ad-hoc Abfragen nach aktuellen Datensätzen gibt es verschiedene Strategien (siehe unten). Da Abfragen sowohl an ein DBMS als auch an ein DSMS durchaus komplex werden können, ist es nicht ohne Weiteres möglich eine generische Schnittstelle, wie sie weiter oben für den Zugriff auf Prozess- und Metadaten gefordert wurde, für die Abfrageverarbeitung zu fordern. Aus diesem Grund sollen die Abfragen konsumentenseitig in entsprechender Syntax des DBMS / DSMS formuliert werden. Hierfür bedarf es einer weitreichenden Unterstützung durch zum Beispiel Werkzeuge zur graphischen Abfrageformulierung oder das Bereitstellen von Metadaten. Zudem existieren Anwendungsfälle, in denen eine Kombination dieser Abfragearten sinnvoll ist. Hierzu gehört insbesondere die Einbettung historischer in ereignisbasierte Abfragen, z.B. eine Benachrichtigung, sobald die Durchschnittstemperatur im Mai über der Durchschnittstemperatur der vergangenen Jahre liegt. Eine derartige Kombination sollte ebenfalls durch einen Vermittler bzw. die Datenverwaltung ermöglicht werden.

Verschiedene ad-hoc-Abfragestrategien (optional) Ad-hoc Abfragen haben das Ziel, möglichst aktuelle Kontextinformationen als Ergebnis zu liefern. In Systemen, in denen die Daten weitestgehend aktiv von den Produzenten publiziert werden, ist dies mit einigen Herausforderungen verbunden, da mit den Produzenten häufig nicht direkt kommuniziert werden kann bzw. soll. Für die Beantwortung von ad-hoc Abfragen können daher verschiedene Strategien angewandt werden:

- Abfrage des letzten in einer Datenbank gespeicherten Datenwertes. Je nach Zeitpunkt der Erhebung der Datenwerte kann dieser Wert jedoch ungenügend aktuell sein, bzw. falls die Daten nicht in der Datenbank gespeichert, sondern nur dem DSMS zugeführt wurden, wird eine solche Abfrage immer ein leeres Ergebnis liefern.
- Subskribieren einer entsprechend generischen Abfrage beim DSMS, um sofort bei Erhebung des nächsten Datenwertes informiert zu werden. Je nach Erhebungsintervall des Produzenten kann das Ergebnis jedoch unter Umständen erst weit in der Zukunft zurückgeliefert werden.
- Das direkte Abfragen beim Produzenten erlaubt die Erhebung eines aktuellen Wertes. Dies jedoch erfordert, dass der Produzent derartige Aufträge entgegennehmen kann (siehe Abschnitt 5.3.1.7, *Delegation von Aufträgen*), was nicht unbedingt gegeben sein muss.

Da es einerseits vom Anwendungsfall und andererseits vom Produzenten abhängig ist, welche dieser Strategien geeignet bzw. überhaupt anwendbar ist, sollte ein Vermittler grundsätzlich alle Strategien ermöglichen.

Multimediale Abfragen (optional) Im Normalfall werden Abfragen in Textform an einen Vermittler übermittelt. Es gibt jedoch eine Reihe von Anwendungsbeispielen, in denen andere Formen zu bevorzugen sind. Beispiele hierfür sind Abfragen in gesprochener Sprache (vgl. *Siri* [Wik12e]), Abfragen in Form von Bildern (vgl. *Google Goggles* [Goo12c]) oder in Form akustischer Fingerabdrücke (vgl. *Soundhound* [Sou12]). Derartige Abfragen bedürfen vermittlerseitig einer speziellen Vorverarbeitung (z.B. Spracherkennung oder Objekterkennung in Bildern), welche die Abfragen in Textform übersetzt und dabei die wesentlichen Informationen aus dem Abfragemedium extrahiert.

Graphische Abfrageformulierung (optional) Wie bereits erwähnt, bedarf die Formulierung von Abfragen besonderer Unterstützung, da dies die wesentliche Schnittstelle zur Nutzung des Systems darstellt. Mit Hilfe der Metadaten ist es dem Nutzer bereits möglich, Abfragen sowie die dazugehörigen Verarbeitungsschritte syntaktisch korrekt zu formulieren. Jedoch führt die textuelle Formulierung unter Umständen zu Benutzbarkeitsproblemen, da sie einerseits zeitaufwändig ist und andererseits eine genaue Kenntnis der verwendeten Syntax erforderlich ist und sich leicht Fehler einschleichen können. Aus

diesen Gründen ist eine graphische Unterstützung der Abfrageformulierung bzw. der Formulierung auszuführender Verarbeitungsschritte sinnvoll.

Visuelle Exploration zur Abfrageunterstützung (optional) Neben der syntaktischen Korrektheit einer Abfrage ist darüber hinaus auch die semantische Korrektheit bzw. die Sinnhaftigkeit einer Abfrage essenziell. Beispielsweise könnte ein Abfragender die Werteskalen eines Sensors oder die eine Abfrage betreffende Datenbasis falsch einschätzen. Eine Abfrage nach der Durchschnittstemperatur in Hamburg über die letzten 50 Jahre liefert kein sinnvolles Ergebnis, falls die Datenbasis lediglich zwei Tage umfasst. Eine syntaktisch korrekte Abfrage liefert natürlich ein Ergebnis, jedoch wird der Abfragende nicht feststellen können, dass es unter Umständen dem Gewünschten nicht entspricht. Aus diesem Grund sollte die Abfrageformulierung auch dahingehend unterstützt werden, dass der Abfragende sich einen Eindruck von der Datenbasis verschaffen kann, ohne gezielt Abfragen stellen zu müssen. Hierfür eignen sich Methoden der visuellen Datenexploration mit Hilfe derer auch große Datenmengen kompakt dargestellt werden können [Le11]. So lassen sich beispielsweise Werteskalen bestimmen, Trends erkennen, Muster entdecken etc. und auf dieser Grundlage sinnvolle Abfragen stellen. Im Rahmen des Vermittlungsprozesses sollten daher seitens eines Vermittlers geeignete Funktionen zur Erstellung visueller Repräsentationen von Zeitreihen angeboten und einem Konsumenten auf Abfrage zur Verfügung gestellt werden.

Testunterstützung für ereignisbasierte Abfragen (optional) Eine weitere Unterstützung sollte speziell für die Formulierung ereignisbasierter Abfragen geboten werden. Während man bei historischen Abfragen ein sofortiges Ergebnis erhält, anhand derer man oft die Sinnhaftigkeit der Abfrage beurteilen kann, ist dies bei ereignisbasierten Abfragen schwierig, da sie in die Zukunft gerichtet sind. Bei einer fehlerhaft formulierten Abfrage, z.B. durch Verwendung ungültiger Attributnamen, kann es zudem passieren, dass man nie eine passende Antwort erhält. Aus diesem Grund bedarf es einer Testunterstützung für ereignisbasierte Abfragen, mittels derer man, ohne auf reale Ereignisse warten zu müssen, Situationen simulieren und anhand derer die Semantik von Abfragen überprüfen kann. Eine Abfrage beispielsweise, die ein Ergebnis liefert, sobald Sensoren einen Waldbrand melden, wird idealerweise vorher mit Hilfe eines Simulators getestet. Ein solcher Simulator ersetzt physische Sensoren durch virtuelle Sensoren, die jedoch die gleichen Ein- und Ausgabeparameter aufweisen. Solche virtuellen Sensoren können vom Konsumenten „programmiert“ werden, eine bestimmte Situation abzuspielen, um somit eine speziell subskribierte Testabfrage auf ihre Sinnhaftigkeit zu überprüfen.

5.3.1.7. Auftragsverwaltung

Neben der Abfrage von Daten können Konsumenten auch Aufträge an die Produzenten übermitteln wollen, beispielsweise um einen Produzenten anzuwei-

sen, seinen Standort zu ändern oder um etwaige Konfigurationseinstellungen neuen Anforderungen anzupassen.

Delegation von Aufträgen (optional) Im Rahmen des Vermittlungsprozesses müssen die Aufträge der Konsumenten seitens eines Vermittlers lediglich an den Produzenten weitergeleitet werden. Da die Logik zum Ausführen eines Auftrags auf Produzentenseite angesiedelt ist, hat ein Vermittler nur dafür Sorge zu tragen, dass ein Auftrag dem Produzenten übermittelt wird. Hierbei ist es jedoch erforderlich die Verfügbarkeit des Produzenten zu berücksichtigen und Aufträge ggf. bis zur erfolgreichen Zustellung zwischenzuspeichern.

Durchführbarkeitsanalyse von Aufträgen (optional) Nicht alle Aufträge sind zu jedem Zeitpunkt von einem Produzenten sinnvoll auszuführen. Einerseits muss die Autonomie des Produzenten respektiert werden, da dieser einen Auftrag auch ablehnen kann, andererseits kann ein Auftrag auch die Fähigkeiten eines Produzenten übersteigen (z.B. das Setzen eines neuen Erhebungsintervalls, welches nicht durch einen Sensor unterstützt wird). Eine solche Durchführbarkeitsanalyse wird in den meisten Fällen vom Produzenten erledigt, sodass ein Vermittler hier, genau wie bei der Delegation von Aufträgen, lediglich eine Weiterleitung der Analyseaufforderung vornimmt. Für den Fall, dass der Produzent z.B. selbst keine Durchführbarkeitsanalyse machen kann, kann ein Vermittler versuchen, anhand der Metadaten des Produzenten zumindest eine Abschätzung der prinzipiellen Durchführbarkeit vorzunehmen.

Statusüberwachung von Aufträgen (optional) Da Aufträge unter Umständen eine längere Zeit zur Ausführung bedürfen, kann es für einen Konsumenten sinnvoll sein, zwischenzeitliche Statusmeldungen abzufragen. Entweder unterstützt ein Produzent von sich aus die Abfrage von Statusmeldungen, sodass einem Vermittler wieder lediglich eine delegierende Funktion zukommt, oder ein Vermittler übermittelt dem Konsumenten Hinweise, z.B. bezüglich der Position des Auftrags in der Warteschlange des Produzenten, anhand derer der Konsument selbst den Status des Auftrags abschätzen kann.

5.3.2. Nicht-funktionale Anforderungen

Die folgenden nicht-funktionalen Anforderungen beschreiben die Qualitätsziele des Vermittlungsprozesses, deren Einhaltung notwendig bzw. wünschenswert ist, um diesen Prozess geeignet zu realisieren bzw. zu nutzen. Sie gehen auf die Frage ein, *wie* das System die im vorigen Abschnitt aufgeführten Funktionen erbringen soll [RR06].

Offenheit Offene Systeme sind dadurch charakterisiert, dass ihre Schlüsselschnittstellen und damit einhergehend die Datenaustauschformate dokumentiert und veröffentlicht sind [CDK12]. Das Ziel hierbei ist im Wesentlichen die Erweiterbarkeit des Systems dadurch, dass Dritten die

Möglichkeit gegeben wird, vorhandene Dienste nicht nur direkt aufzurufen, sondern auch als Teil eigener höherwertiger Verarbeitungsfunktionen zu verwenden und ggf. auch neue, eigene Dienste in das System einzubinden. Auch wird durch Veröffentlichung der Schnittstellen ein Austausch von Komponenten ermöglicht, sodass für einzelne Dienste neue Implementierungen erarbeitet und verwendet werden können. Idealerweise beruht die Offenheit eines Systems auf der Berücksichtigung etablierter Standards [CDK12]. Dies umfasst beispielsweise die Verwendung weit verbreiteter Kommunikationsprotokolle wie TCP/IP, HTTP und SOAP, die Verwendung gängiger Beschreibungssprachen bzw. Notationen wie XML oder JSON sowie auf den höheren Ebenen die Berücksichtigung standardisierter Dienstschnittstellen. Im Rahmen des Sensor Webs wären dies beispielsweise die Schnittstellen- und Datenmodelle des *Open Geospatial Consortiums* (vgl. Abschnitt 6.1.2) und für das Internet der Dinge zeichnet sich *EPCglobal* bzw. *Global Standards One* (vgl. Abschnitt 6.2.1) für die Standardisierung entsprechender Dienste und deren Schnittstellen verantwortlich.

Heterogenität Heterogenität im Allgemeinen bedeutet soviel wie Vielfalt oder Unterschiedlichkeit [CDK12]. Sie umfasst im Rahmen des Vermittlungsprozesses mehrere Dimensionen:

Informationen Kontextdaten repräsentieren jedwede Information, die zur Beschreibung der Situation einer Entität herangezogen werden können [DA99]. Entsprechend mannigfaltig können die Inhalte sein, die Kontextdaten darstellen, beispielsweise Temperaturwerte, persönliche Präferenzen oder der Ausführungsstatus von Geschäftsprozessen. Diese Inhalte können zudem mittels unterschiedlicher Medien repräsentiert werden, zum Beispiel als Text, als Bild oder in akustischer Form (z.B. Sprache). Abhängig vom Medium können zudem eine Vielfalt von Formaten zur Kodierung der Inhalte Verwendung finden: Für textuelle Repräsentation bieten sich z.B. kommaseparierte Listen, XML oder JSON an, Bilder können beispielsweise im JPEG-, PNG- oder GIF-Format vorliegen und akustische Aufzeichnungen können z.B. im WAV- oder MP3-Format Verwendung finden.

Geräte Da die globale Vermittlung kontextbasierter Sichten einigen Aufwand bedeutet, sind im Kern eines Vermittlers vorzugsweise leistungsstarke Server oder Desktop-Computer zu finden. An die Produzenten und Konsumenten werden keinerlei Anforderungen bezüglich ihrer Ressourcenausstattung gestellt. Entsprechend können hier auch (sehr) leistungsschwache Geräte, z.B. Tablets, Mobiltelefone oder Sensoren eingesetzt werden.

Netzwerke Kontextdaten werden auf unterster Ebene von Sensoren erhoben und in der Praxis lokal über ein sogenanntes *Personal Area Network* (PAN) an eine Basisstation übertragen. Von dort aus können die Daten, ggf. über weitere Zwischenstationen in einem *Local Area Network*

(LAN), schließlich an einen Vermittler, z.B. über das Internet als *Wide Area Network* (WAN), kommuniziert werden. Kommunikationstechnologien und -protokolle unterscheiden sich stark in diesen verschiedenen Arten von Netzen, da sie häufig unterschiedlichen Anwendungszwecken dienen (vgl. Abschnitt 4.4.2). Eine Vielfalt weiterer Technologien und Protokolle müssen zudem berücksichtigt werden, sofern mobile Teilnehmer in beliebigen Rollen im Rahmen des Vermittlungsprozesses eingesetzt werden.

Teilnehmer Wie in Abschnitt 4.2 bereits vorgestellt, wirken Teilnehmer in unterschiedlichen Rollen am Vermittlungsprozess mit. So gibt es Produzenten, die Kontextdaten erheben und publizieren, Konsumenten, welche die Daten nachfragen, einen Vermittler, der zwischen einzelnen Rollen vermittelt und Stellvertreter, welche im Auftrag von Teilnehmern mit einem Vermittler interagieren. Nicht nur haben die unterschiedlichen Rollen heterogene Ziele, auch die Teilnehmer, welche diese Rollen ausfüllen, können beliebige Anforderungen und Wünsche an den Vermittlungsprozess stellen.

Dieser Vielfalt muss im Rahmen des Vermittlungsprozesses Rechnung getragen werden, indem Informationen jedweder Art zwischen beliebigen Teilnehmern mit heterogenen Geräten über unterschiedliche Netzwerke ausgetauscht werden können.

Flexibilität Der Vermittlungsprozess soll dahingehend flexibel sein, dass er in verschiedensten Anwendungsszenarien mit unterschiedlichen Anforderungen eingesetzt werden kann. Auch die Flexibilität umfasst hierbei mehrere Dimensionen. Einerseits soll der Vermittlungsprozess beispielsweise zwischen einem lokalen Sensor und einer lokalen Anwendung vermitteln können und darf daher nicht darauf angewiesen sein, dass Teile seiner Funktionalität ausschließlich verteilt im Netzwerk ausgeführt werden. Andererseits soll der Prozess auch auf einer globalen Skala zwischen geographisch separierten Entitäten vermitteln können. Je nach Anwendungsszenario können auch einzelne Bestandteile des Vermittlungsprozesses obsolet sein (z.B. eine Zugriffskontrolle oder die Auftragsverwaltung). Dies setzt ein entsprechend flexibles Deployment voraus, bei dem nicht benötigte Komponenten ignoriert werden können. Auf diese Weise lässt sich ein Vermittler auch an unterschiedliche Ausführungsumgebungen (z.B. leistungsstarke Server oder ressourcenschwache Mobilgeräte) anpassen. Aus der Perspektive der Softwareentwicklung muss der Prozess zudem einerseits offen sein, sodass einzelne Komponenten einfach gegen andere Implementierungen ausgetauscht werden können, sowie andererseits erweiterbar, damit neue Funktionalitäten hinzugefügt werden können.

Erweiterbarkeit Mit der immer weiter voranschreitenden Verbreitung von Sensoren werden Kontextinformationen zur Erbringung von Diensten eine im-

mer größere Rolle spielen. Damit einhergehend werden auch die Möglichkeiten und Anforderungen kontextbasierter Anwendungen steigen. Erfahrungen mit dem praktischen Einsatz von Kontextinformationen führten zudem bereits zu einer Evolution der Standards, die für den Bereich der Vermittlung kontextbasierter Sichten relevant sind [BL09]. Zusammen mit der geforderten Flexibilität soll die Anforderung der Erweiterbarkeit sicherstellen, dass ein bestehendes System nicht nur flexibel an aktuelle Bedürfnisse angepasst werden kann, sondern zudem auch zukünftigen Anforderungen gerecht wird. Hierzu ist es unumgänglich, dass die Funktionalität eines Vermittlers einfach erweitert werden kann, beispielsweise durch Hinzufügen neuer Verarbeitungsfunktionen, Datenbankmanagementsysteme oder Kommunikationsadapter.

Verteilung Auch wenn alle an einer Vermittlung beteiligten Rollen lokal auf demselben Gerät angesiedelt sein können und im Rahmen des Vermittlungsprozesses lediglich lokal erhobene Kontextdaten an lokale Anwendungen vermittelt werden, erstreckt sich die Vermittlung im Normalfall über Gerätegrenzen hinweg. Produzenten, Konsumenten und Vermittler stellen somit unabhängige Knoten dar, die über ein Netzwerk miteinander verbunden sind. Aber insbesondere ein Vermittler selbst kann zudem die Leistungserbringung über mehrere Knoten aufteilen. So kann beispielsweise die Datenverwaltung auf einer verteilten Datenbank aufsetzen oder die Verarbeitungsfunktionen für Kontextdaten auf unterschiedlichen Computern ausgeführt werden. Direkte Konsequenzen daraus sind beispielsweise ein unabhängiges Fehlerverhalten und echte Parallelität [CDK12], aber auch die Möglichkeit durch Hinzufügen weiterer Ressourcen die Effektivität zu erhöhen.

Skalierbarkeit Laut Vorhersagen wird die Anzahl an Geräten oder allgemein Objekten im Internet in den nächsten Jahren drastisch ansteigen. Damit einher geht auch ein starker Anstieg der Menge an Kontextdaten, die über das Internet ausgetauscht werden [IDC12]. Ein System gilt dann als skalierbar, wenn es weiterhin effektiv arbeitet, auch wenn die Anzahl an Nutzern, Verarbeitungsressourcen und Daten steigt [CDK12]. Das führt zu der impliziten Forderung, dass es auf Seiten eines Vermittlers keinen Flaschenhals geben darf. Das wiederum bedeutet, dass jede nicht-optionale Aufgabe über mehrere Knoten (im Fall eines verteilten Systems) bzw. mehrere Betriebssystemprozesse verteilt ausgeführt werden können muss. Man muss hier zwischen horizontaler (engl. *scale-out*) und vertikaler (engl. *scale-up*) Skalierung unterscheiden. Bei vertikaler Skalierung werden einer lokalen Installation zusätzliche Hardware (z.B. weitere CPUs, mehr Speicher) zur Verfügung gestellt. Bei horizontaler Skalierung werden in einem verteilten System zusätzliche Knoten eingebunden [Sch07]. Da der Vermittlungsprozess einerseits darauf ausgelegt ist, auf verschiedenen Skalen (lokal, regional und global) zu operieren, und andererseits dynamisch Ressourcen zu (de-)allozieren, ist die Anforderung einer guten Skalierbarkeit obligatorisch.

Robustheit Falls im Rahmen des Vermittlungsprozesses ein Dienst oder die Datenhaltung ausfällt, sollte dies nicht die Verfügbarkeit des Gesamtsystems einschränken. In einem verteilten System ohne sogenannten *Single Point of Failure* und mit unabhängigem Fehlerverhalten können hierfür Dienste repliziert werden, sofern diese entweder zustandslos sind, der aktuelle Zustand eines Dienstes jederzeit in einem Persistenzspeicher hinterlegt ist und von dort wiederhergestellt werden kann oder sich die Replikate bei Zustandsänderung über entsprechende Protokolle synchronisieren. Zur Detektion eines Ausfalls müssen alle Komponenten kontinuierlich beobachtet werden, damit ggf. auf einen Ausfall reagiert werden kann. Da ein gänzlich robustes System schwer zu realisieren ist und der Aufwand, je nach Anforderung an den Grad der Robustheit, immens sein kann, sollten zumindest elementare Aufgaben möglichst robust gestaltet werden.

Transparenz Der Transparenzbegriff steht für das Verbergen technischer Details vor dem Benutzer bzw. dem Anwendungsprogrammierer, sodass dieser ein (verteiltes) System nicht als ein Konglomerat unabhängiger Komponenten, sondern als ein Ganzes wahrnimmt [CDK12]. Man unterscheidet im Wesentlichen acht verschiedene Arten von Transparenz, von denen im Kontext dieser Arbeit die folgenden fünf von besonderer Bedeutung sind [CDK12]:

Netzwerktransparenz Diese Art der Transparenz umfasst sowohl die *Zugriffs*- als auch die *Ortstransparenz*. Erstere erlaubt den Zugriff auf lokale sowie entfernte Ressourcen unter Verwendung identischer Operationen. Auf diese Weise muss beispielsweise ein Konsument kein Wissen darüber haben, ob abgefragte Informationen von lokalen oder entfernten Sensoren erhoben wurden. Eng daran angelehnt ist die Ortstransparenz, die einen Zugriff auf Ressourcen unabhängig von ihrem physischen Ort oder ihrer konkreten Endpunktadresse im Netzwerk erlaubt. Dieses bedarf der Verwendung logischer Adressen (vgl. Abschnitt 5.3.1).

Fehlertransparenz Falls im Rahmen der Vermittlung unerwartete Fehler, z.B. durch den Ausfall eines Dienstes, auftreten, sollen diese vor den Produzenten und Konsumenten verborgen bleiben. Stattdessen muss ein Vermittler dafür sorgen, dass der Fehler kompensiert wird, beispielsweise durch Wiederholung der Operation mittels eines äquivalenten Dienstes.

Mobilitätstransparenz Die Mobilitätstransparenz ermöglicht die Mobilität von Produzenten und Konsumenten, ohne dass diese bedeutende funktionale Einschränkungen bei der Vermittlung kontextbasierter Sichten hinnehmen müssen. Hierfür bedarf es einer entsprechenden Instanz zur Mobilitätsverwaltung (vgl. Abschnitt 5.3.1).

Leistungstransparenz Unter Leistungstransparenz versteht man die Möglichkeit ein System zur Laufzeit zu rekonfigurieren, um dessen Effektivität der aktuellen Last anzupassen. Beispielsweise könnten häufig aufgerufene Dienste von stark ausgelasteten Netzwerkknoten

auf weniger ausgelastete Knoten verschoben, neue Knoten hinzugefügt oder vorhandene entfernt sowie Ausführungsparameter der Dienste angepasst werden, ohne dass das Gesamtsystem, zum Beispiel durch einen erforderlichen Neustart, für ein längeres Zeitintervall nicht verfügbar ist.

Skalierungstransparenz Diese Art der Transparenz ermöglicht die Skalierung eines Systems, ohne dass Änderungen an der allgemeinen Struktur oder den Diensten vorgenommen werden müssen. Im Zusammenspiel mit der Leistungstransparenz ermöglicht dies eine Adaption zur Laufzeit, bei der abhängig von der aktuellen Gesamtsystemlast neue Ressourcen hinzugefügt oder nicht mehr verwendete Ressourcen entfernt werden können.

Ein System welches obige Transparenzen realisiert, vereinfacht nicht nur die Nutzung für (mobile) Produzenten und Konsumenten, sondern kann sich darüber hinaus selbstorganisierend an aktuelle Leistungsanforderungen adaptieren.

Sicherheit Die Schutzziele im Rahmen des Vermittlungsprozesses beziehen sich in erster Linie auf die Datensicherheit sowie auf die Dienstsicherheit. Bei der Datensicherheit muss insbesondere der Zugriff auf Kontextdaten und Metadaten auf berechtigte Instanzen eingeschränkt werden können, da diese Daten sensible Informationen repräsentieren können (siehe auch *Zugriffskontrolle* in Abschnitt 5.3.1). Der Zugriff umfasst hierbei sowohl das Lesen als auch das Schreiben von Daten, um nicht nur die Vertraulichkeit, sondern auch die Integrität der Daten sicherzustellen. Des Weiteren muss die Verfügbarkeit der Daten gewährleistet werden, damit jederzeit benötigte Daten aufgefunden und zugegriffen werden können. Zudem muss jeder Zugriff geeignet protokolliert werden, sodass ein Zugriff seitens eines Klienten von diesem nicht abgestritten werden kann. Dies dient einerseits dem Nachvollziehen von Änderungen, andererseits kann ein Zugriff auch kostenpflichtig sein und muss daher dem Klienten in Rechnung gestellt werden können.

Ebenso wie die Daten müssen auch die Dienste vor unberechtigtem Zugriff geschützt werden. Dies umfasst alle Dienste zur Verarbeitung von Kontextdaten, die durch Produzenten und Konsumenten orchestriert werden können, jedoch insbesondere auch Dienste, die im Rahmen der Prozessverwaltung ausgeführt und nicht direkt durch Produzenten und Konsumenten zugegriffen werden sollen. Außerdem muss die Verfügbarkeit aller Dienste, die durch einen Vermittler realisiert werden, dahingehend gewährleistet sein, dass alle im Rahmen der Vermittlung nicht-optionalen Dienste, die an sie gestellten Ausführungsanforderungen jederzeit adäquat erfüllen. Dienste, die im Rahmen der Orchestrierung von Klienten extern referenziert werden (siehe *Benutzerdefinierte Verarbeitungsfunktionen* in Abschnitt 5.3.1), betrifft dies nicht. Außerdem muss, ebenso wie bei der Datensicherheit, der Dienstzugriff protokolliert werden, um das Kriterium der Nichtabstreitbarkeit zu realisieren und Abrechnungsfunktionen zu ermöglichen.

Zur Realisierung der obigen Schutzziele bedarf es zudem sicherer Kommunikationskanäle, da alle Zugriffe im Normalfall per Nachrichtenaustausch (über ein Netzwerk) erfolgen. Sichere Kanäle für die Kommunikation zwischen Vermittler und Produzenten bzw. Konsumenten sind optional, da nicht sichergestellt werden kann, dass Klienten entsprechende Protokolle beherrschen.

Lose Kopplung Unter Kopplung versteht man den Grad der Abhängigkeit zwischen Komponenten eines Systems [CDK12, Win05]. Eine enge Kopplung bezeichnet eine hohe Abhängigkeit, wohingegen eine lose Kopplung für eine weniger enge Bindung steht. Bei eng miteinander gekoppelten Komponenten ziehen Änderungen in einer Komponente häufig auch Anpassungen an anderen, gekoppelten Komponenten nach sich, was softwaretechnisch für qualitativ mindere Systeme steht [YC79]. Bei Systemen, in denen einzelne Komponenten mittels Nachrichten kommunizieren, lassen sich eine lose Kopplung z.B. durch einfache, wohldefinierte Schnittstellen erreichen, welche die Implementierung der Funktionen vor dem Klienten verbergen, durch asynchronen Nachrichtenaustausch sowie durch Verwendung von Verzeichnisdiensten, mit Hilfe derer geeignete Dienste zur Laufzeit aufgefunden und gebunden werden können [CDK12].

Starke Kohäsion Neben der Kopplung ist die Kohäsion eine weitere Metrik zur Bestimmung der Softwarequalität. Kohäsion bezeichnet hierbei den Grad der inhaltlichen Verbundenheit von Funktionen einzelner Komponenten. Hierbei spricht man von starker Kohäsion, falls eine Komponente sehr fokussiert auf ihre eng abgegrenzte Verantwortlichkeit ist. Komponenten mit einer schwachen Kohäsion hingegen offerieren meist Funktionen, die funktional nicht miteinander in Beziehung stehen und keinen ähnlichen Zweck erfüllen. Eine starke Kohäsion ist prinzipiell zu bevorzugen, da sie die Verständlichkeit, die Wartbarkeit und die Wiederverwendbarkeit einzelner Komponenten erhöht [YC79, Win05].

5.4. Identifikation von Teilproblemen und Lösungsansätzen

In der vorangegangenen Analyse funktionaler und nicht-funktionaler Anforderungen wurden bereits auf einer abstrakten Ebene allgemeine Anforderungen an den Vermittlungsprozess identifiziert. Um diese Anforderungen adäquat umsetzen zu können, sollen in diesem Abschnitt geeignete *Unterstützungspunkte* bestimmt werden. Unterstützungspunkte konkretisieren die aufgestellten Anforderungen, indem sie Teilprobleme identifizieren und mit existierenden Lösungen zusammenführen. Sie repräsentieren einzelne Aspekte eines Systems, die im Entwurf eines Vermittlers (siehe Abschnitt 7.1) berücksichtigt werden sollen. Eine solche weitergehende Dekomposition der Anforderungen ist hilfreich um existierende Lösungen für einzelne Aspekte zu finden. Da der Vermittlungsprozess eine Reihe von Aufgaben umfasst, für die bereits adäquate Lösungen existieren, müssen im späteren

Entwurf insbesondere die Fragen beantwortet werden, wie sich existierende Lösungen integrieren und offene Unterstützungspunkte geeignet realisieren lassen. Außerdem lassen sich anhand von Unterstützungspunkten verwandte Arbeiten diskutieren und bewerten (siehe Kapitel 6). Beginnend mit Unterstützungspunkten, die die Systemarchitektur betreffen, werden im Anschluss weitere Unterstützungspunkte, unterteilt nach den Aufgabenbereichen, die in Abschnitt 4.4 identifiziert wurden, vorgestellt.

5.4.1. Systemarchitektur

Bevor in den folgenden Abschnitten konstituierende Komponenten und Schnittstellen eines Vermittlers aus den identifizierten Anforderungen und Herausforderungen des Vermittlungsprozesses abgeleitet werden, sollen zunächst allgemeine Aspekte, welche die Systemarchitektur eines Vermittlers betreffen, betrachtet werden. Hierzu gehören die Ausgestaltung von Systemkomponenten, Kommunikationsaspekte in verteilten Systemen, die Gestaltung technologieneutraler Schnittstellen sowie Möglichkeiten zur adaptiven Ressourcenverwaltung.

Lose gekoppelte und stark kohäsive Systemkomponenten

Wie in der Analyse nicht-funktionaler Anforderungen in Abschnitt 5.3.2 bereits geschildert, sind Kopplung und Kohäsion zwei Metriken zur Bewertung der Qualität von Software. Kopplung bezeichnet dabei den Grad der Abhängigkeit zwischen Komponenten und Kohäsion die inhaltliche Verbundenheit der Funktionen einer Komponente. Um eine lose Kopplung von Komponenten bzw. Diensten zu erreichen, gibt es eine Reihe unterschiedlicher Möglichkeiten [CDK12]:

Schnittstellen Eine Separation von Schnittstellen und den dazugehörigen Implementierungen ermöglicht Änderungen an einer Komponente, ohne dass eine Anpassung abhängiger Komponenten erforderlich ist (sofern sich die Schnittstelle selbst nicht ändert). Die Schnittstellen sollten zudem einfach und allgemein gehalten sein, da diese einerseits robuster gegenüber Änderungen der Implementierung sind und andererseits eine weitere Abhängigkeit von Operationsnamen verringert wird. Beispiele wie die REST-Schnittstellen im World Wide Web [PZL08] zeigen, dass diese lose Kopplung durchaus auch komplexe Anwendungsszenarien unterstützt. Dies führt auch zu einer eher datenorientierten Sicht, da anhand der Semantik der Daten über die Art der Verarbeitung entschieden werden kann.

Kommunikation Eine synchrone Kommunikation nach dem Anfrage/Antwort-Prinzip führt zu einer höheren Kopplung von Komponenten, da diese einerseits gleichzeitig verfügbar sein und sich andererseits miteinander synchronisieren müssen. Durch asynchrone Kommunikation kann zumindest die Synchronisierungskopplung (engl. *Synchronization*

Coupling) vermieden werden. Durch Verwendung von Nachrichtenwarteschlangen als Indirektion zwischen Sender und Empfänger können darüber hinaus auch eine Entkopplung in Raum und Zeit ermöglicht werden, da nicht beide Kommunikationsteilnehmer gleichzeitig verfügbar sein müssen, sondern sich nach Bedarf über die Nachrichtenwarteschlagen austauschen können.

Verzeichnisse Bevor ein Kommunikationsakt initiiert werden kann, muss der Sender den Empfänger der Nachricht spezifizieren. In einfachen Systemen geschieht dies entweder bereits zur Entwurfszeit oder manuell zur Laufzeit. Auch hierdurch entsteht eine enge Kopplung zwischen Sender und Empfänger. Falls möglich, sucht sich der Sender einen geeigneten Empfänger erst zur Laufzeit. Auf diese Weise lässt sich ein passender Endpunkt unter mehreren potentiellen Empfängern dynamisch auswählen und man ist weitestgehend unabhängig von der Kenntnis über dessen aktuelle physische Endpunktadresse, was insbesondere auch bei Berücksichtigung mobiler Teilnehmer bedeutsam ist.

Starke Kohäsion korreliert häufig mit loser Kopplung, jedoch sind beides im Grunde konkurrierende Kriterien [Win05]. Eine starke Kohäsion verlangt, dass die Aufgaben einer Komponente auf einen funktional zusammenhängenden Bereich begrenzt sind. Dies führt dazu, dass man in einem System eine Vielzahl leichtgewichtiger Komponenten mit spezifischen Aufgaben hat. Je größer die Anzahl der Komponenten, je höher ist jedoch auch die Wahrscheinlichkeit, dass eine Komponente mit vielen anderen assoziiert ist, wodurch die Kopplung enger werden kann. Es gilt also einerseits durch oben erwähnte Maßnahmen die Kopplung so lose wie möglich zu halten, andererseits durch einen guten Systementwurf einen geeigneten Grad an Kohäsion zu finden. Entsprechende Entwurfsprinzipien sollen an dieser Stelle nicht weiter detailliert werden und es sei hierfür auf [YC79, Win05] sowie Abschnitt 7.1 verwiesen.

Verteilte Bearbeitung

Wie bereits erwähnt, soll die Verarbeitung des Vermittlungsprozesses verteilt in einem Netzwerk erfolgen können. Hierfür müssen Informationen zwischen einzelnen Komponenten mittels Nachrichten übertragen werden, welche die auszutauschenden Informationen in serialisierter Form kapseln. Der Nachrichtenaustausch kann explizit oder implizit geschehen. Beim expliziten Austausch muss der Sender eine Nachricht erstellen, die notwendigen Informationen einbetten, den Adressaten festlegen und anschließend die Nachrichten an einen Transportdienst zur Überlieferung übergeben. Im Gegensatz dazu abstrahiert der implizite Nachrichtenaustausch von den Details des Austauschs. Der Sender ruft lokal eine Methode eines sogenannten *Stubs* oder *Proxys* auf und übergibt die benötigten Informationen als Parameter. Das Erstellen und der Versand der Nachricht erfolgen dann im Hintergrund und transparent

für den Sender. Ein solches Verfahren wird auch als entfernter Methodenaufruf (engl. *Remote Procedure Call* bzw. *Remote Method Invocation*) bezeichnet [CDK12].

Vorteil des expliziten Nachrichtenaustauschs ist die Möglichkeit auch komplexere Interaktionsprotokolle (z.B. Ausschreibungen und Verhandlungen) zwischen den Komponenten realisieren zu können. Jedoch sind die Funktionsschnittstellen meist nur implizit in Form einer Dokumentation festgehalten. Eine Überprüfung z.B. der Korrektheit von Parametern erfolgt somit erst zur Laufzeit. Beim entfernten Methodenaufruf kann dies meist bereits zur Entwicklungszeit erfolgen, was eine frühzeitige Erkennung von Fehlern erlaubt. Allerdings beschränkt sich die Interaktion meist auf das Anfrage/Antwort-Muster. Komplexere Protokolle müssen durch eine Sequenz von Methodenaufrufen nachgebildet werden. Durch das Verbergen der Komplexität des Nachrichtenaustauschs sorgt jedoch der implizite Nachrichtenaustausch für eine wesentlich einfachere Programmierung und erlaubt vollständige Zugriffstransparenz.

Da beide Verfahren ihre Vor- und Nachteile haben und für jeweils unterschiedliche Einsatzzwecke geeignet sind, sollten beide Verfahren im Rahmen des Vermittlungsprozesses unterstützt werden. Grundsätzlich bedarf es hierfür eines Nachrichtentransportsystems, welches den Austausch von Nachrichten zwischen zwei Kommunikationspartnern ermöglicht. Zur Realisierung des entfernten Methodenaufrufs müssen zudem explizite Schnittstellen definiert werden, auf Basis derer klientenseitig die Stubs bzw. Proxies generiert werden können. Darüber hinaus unterstützen explizite Schnittstellen auch die lose Kopplung (siehe oben).

Technologieneutrale Schnittstellen

Man muss grundlegend zwischen unterschiedlichen Arten von Schnittstellen unterscheiden. Um eine lose Kopplung zu erreichen und einen entfernten Methodenaufruf zu realisieren, bedarf es, wie oben detailliert, der Definition von Schnittstellen im programmiersprachlichen Sinne. Also einer abstrakten Beschreibung einer oder mehrerer Funktionen, die eine oder mehrere Klassen (oder allgemein Komponenten) implementieren und mit Hilfe derer entfernte Komponenten interagieren und kommunizieren können [Cle03]. Ein Aufruf an solch einer Schnittstelle erfolgt immer lokal, das heißt aus demselben Adressraum eines Programms (auch bei entfernten Methodenaufrufen erfolgt der Aufruf an der Schnittstelle zunächst lokal). Sofern alle im Vermittlungsprozess involvierten Knoten dieselben Technologien (z.B. Programmiersprachen) verwenden, würde diese Art von Schnittstelle ausreichend sein. Allerdings sind solche Schnittstellen spezifisch für eine Programmiersprache und in der Anforderungsanalyse wurde gefordert, dass der Vermittlungsprozess offen sein und heterogene Klienten unterstützen soll, dass heißt er muss existierende und idealerweise technologieneutrale Standards berücksichtigen. Dies gilt mindestens für alle Funktionen, die von Produzenten und Konsumenten

in Anspruch genommen werden, z.B. das Subskribieren von Abfragen oder das Übermitteln von Messwerten. Möchte man also Funktionen nach außen hin sichtbar machen, bedarf es einer anderen, technologieneutralen Art der Schnittstellenbeschreibung.

Verschiedene de-facto und de-jure Standards haben sich in diesem Bereich etabliert. Zu den Prominentesten gehören beispielsweise:

OMG IDL Diese Sprache zur Schnittstellendefinition (engl. *Interface Definition Language*, IDL) der *Object Management Group* wurde im Rahmen des *CORBA*-Standards veröffentlicht und erlaubt eine technologieneutrale Definition von Schnittstellen, welche dann sowohl für den lokalen als auch entfernten Zugriff auf Programmkomponenten verwendet werden können [OMG13].

WSDL Die *Web Services Description Language* (WSDL) [W3C06] des *World Wide Web Consortiums* (W3C) gehört mit zu den am meisten genutzten Beschreibungssprachen für Dienstschnittstellen. Sie ähnelt der OMG IDL, erfüllt den gleichen Zweck, basiert jedoch auf XML.

Protocol Buffers, Thrift Auch unter anderem diese Technologien stellen Beschreibungssprachen für Schnittstellen zur Verfügung. Allerdings wurden sie nicht standardisiert. Die *Protocol Buffers* von Google [Goo12d] sind insbesondere für die effiziente Übertragung von Nachrichten zwischen entfernten Diensten entwickelt worden und bieten objektorientierte Schnittstellenbeschreibungen. Das Gleiche gilt auch für das ursprünglich von Facebook entwickelte und nun von der Apache Software Foundation betreute *Thrift* [Apa12b].

Für alle Beschreibungssprachen existieren Werkzeuge zur Generierung von *Stubs* und *Skeletons*, welche bei einem entfernten Aufruf das Bindeglied zwischen den Schnittstellen und deren Implementierung darstellen. Auch existieren Werkzeuge zum Überführen einer Schnittstellenbeschreibung in eine nahezu beliebige andere. Daher ist die Systemarchitektur lediglich so zu gestalten, dass die Implementierungen der wesentlichen Dienste von ihren lokalen Schnittstellenbeschreibungen getrennt werden. Beschreibungen für den externen Aufruf können beliebig auch im Nachhinein ergänzt werden und haben keinen weiteren Einfluss auf die Gestaltung der Systemarchitektur.

Adaptive Ressourcenverwaltung

Bei einer verteilten Ausführung des Vermittlungsprozesses interagieren mehrere Knoten innerhalb eines Netzwerks miteinander, um den Prozess kollaborativ auszuführen. Dabei soll die Anzahl beteiligter Knoten dahingehend variabel sein, dass sowohl während des Deployments als auch zur Laufzeit beliebige Knoten aus dem Prozess entfernt oder neue Knoten hinzugefügt werden können, damit die Menge zur Verfügung stehender Ressourcen an die aktuelle Last angepasst werden kann.

Das Hinzufügen eines Knotens kann auf unterschiedliche Weise erfolgen: Im einfachsten Fall laufen auf dem Knoten bereits Vermittlungsdienste, deren Präsenz durch Publikation in einem Verzeichnis angekündigt oder die mittels Broadcasts in einem lokalen Netz aufgefunden und in den Vermittlungsprozess einbezogen werden können (siehe Abschnitt 5.4.2, *Auffinden konkreter Endpunkte*). Laufen auf einem neuen Knoten noch keine Vermittlungsdienste, können auf diesem entweder einfach neue Dienstinstanzen erzeugt oder vorhandene Instanzen migriert werden. In [Bad07] wurde jedoch gezeigt, dass die starke Migration von Programmkomponenten, d.h. inklusive ihres momentanen Ausführungszustandes, zwar prinzipiell möglich, in der Praxis aber in gängigen Programmiersprachen nicht vorgesehen und daher schwierig umzusetzen ist. In Ausführung befindliche Dienste müssen ihre Aufträge entsprechend zunächst beenden, um einen Datenverlust zu verhindern. Ist diese Forderung erfüllt, lässt sich auch die Migration eines Dienstes realisieren, indem eine laufende Instanz des Dienstes auf einem Knoten beendet und eine neue Instanz auf einem anderen Knoten erzeugt wird. Da Dienste in der Konsequenz entweder zustandslos sein oder ihren Zustand untereinander synchronisieren oder in einem Persistenzspeicher verwalten müssen, muss lediglich sichergestellt werden, dass der Programmcode auf dem neuen Knoten zu Verfügung steht. Dies kann entweder bereits im Rahmen des Deployments erfolgen oder aber zur Laufzeit durch das Nachladen von Code bzw. Code-Bibliotheken aus speziellen sogenannten *Repositories*. Durch Unterstützung sowohl einer Metadatenverwaltung, über die verfügbare Dienste veröffentlicht werden können, als auch von Repositories ließen sich dem System somit beliebige Knoten zur Laufzeit hinzufügen.

Um einen Knoten schließlich zu entfernen, muss dieser ordentlich beim Vermittler bzw. dessen Metadatenverwaltung abgemeldet werden, damit keine neuen Aufträge an diesen Knoten übergeben werden.

5.4.2. Datenübermittlung

Unter der Annahme, dass im Rahmen des Vermittlungsprozesses Daten zwischen in einem Netzwerkwerk verteilten Entitäten ausgetauscht werden, muss die Kommunikation zwischen den Beteiligten mittels Nachrichten erfolgen. Insbesondere aufgrund der großen Heterogenität zwischen allen Beteiligten bedarf die Kommunikation daher einer besonderen Unterstützung. Hierzu gehören die Verwendung von Adaptern zur protokoll- und technologieunabhängigen Kommunikation, damit einhergehend ein logisches Adressierungsschema zur Kennzeichnung sowie Mechanismen zum Auffinden von Endpunkten und zur Publikation angebotener Dienste und letztlich eine Mobilitätsunterstützung für Endpunkte in besonders dynamischen Umgebungen.

Kommunikationsadapter

Der Vermittlungsprozess erstreckt sich unter Umständen über Netzwerke verschiedener Skalen. Von *Personal Area Networks* bis hin zu *Wide Area Networks*

existieren eine Vielzahl an Kommunikationstechnologien und -standards (vgl. Abschnitt 4.4.2), die im Rahmen der Vermittlung überbrückt werden müssen. Insbesondere auf dem Kommunikationsweg zwischen Produzenten und Vermittler sowie zwischen Vermittler und Konsumenten herrscht eine große Heterogenität der eingesetzten Technologien. An diesem Punkt, in der Peripherie eines Vermittlers, bedarf es daher der Unterstützung durch sogenannte Kommunikationsadapter. Ein Adapter ermöglicht die Vermittlung zwischen unterschiedlichen Technologien und Standards, indem er intern eine Übersetzung vornimmt. Auf Produzentenseite werden beispielsweise Technologien und Protokolle wie *Bluetooth* [Blu10], *ZigBee* [Zig12] und die *IEEE 1451*-Standards [Nat12b] eingesetzt, auf Seite eines Vermittlers kommen etablierte Technologien aus dem Internet und dem World Wide Web zum Einsatz. Ein Adapter auf Seite des Vermittlers muss die Nachrichten eines Produzenten annehmen und an die Schnittstellen der internen Vermittlungsdienste weiterleiten. Hierbei ist es erforderlich, dass der Produzent den Empfänger entweder mittels einer konkreten physischen Adresse oder logisch adressiert (siehe unten). Möchte ein Vermittler eine Nachricht an den Produzenten übermitteln, verläuft die Kommunikation über den Adapter in umgekehrter Richtung, wobei auch hier wieder eine entsprechende Adressierung berücksichtigt werden muss.

Der Einsatz von Kommunikationsadaptern ist gängige Praxis, verlangt aber einen vergleichsweise hohen Aufwand, da für jede Kombination an Technologien entsprechende Adapter entwickelt werden müssen. Bei weiter fallenden Sensorpreisen kann dieser Aufwand in größeren Projekten zu einem der wesentlichen Kostenschlüssel werden [BFJP10]. Entsprechend gibt es eine Vielzahl an Realisierungen, die zwischen unterschiedlichen Technologien und Protokollen vermitteln können. Beispiele hierfür sind das *Sensor Bus*-Projekt [BFJP10], welches einen Nachrichten-Bus realisiert, an den sich einerseits proprietäre Sensoren über *Gateways* und andererseits Dienste auf Anwendungsebene über entsprechende Schnittstellen binden können. Auch *SOCRADES* [DSSG+08] verwendet Gateways, um Sensoren und andere Geräte mit proprietären Kommunikationsschnittstellen an die Infrastruktur anzubinden. In [ZBH+09] wird eine Gateway-Architektur vorgestellt, welche im Rahmen der *Web Services for Devices*-Initiative (WS4D) einen Ansatz zur Kommunikation beliebiger Geräte über technologiespezifische Adapter mit den Standarddiensten der Sensor Web Enablement-Initiative erlaubt. Ein ähnliches Ziel verfolgt auch das *Smart Transducer Web Services*-Projekt [SL08], welches sich speziell auf Geräte, die dem IEEE 1451-Standard entsprechen, konzentriert und diese mittels Web Services über das Internet zugreifen kann.

Das Ziel all dieser Projekte ist die Anbindung von Geräten, die nicht mittels der Standard-Internetprotokolle kommunizieren, an Web-basierte Infrastrukturen. Hierzu werden auf der einen Seite technologiespezifische Adapter bzw. Gateways zur Kommunikation mit den Geräten bereitgestellt, auf der anderen Seite bieten technologieneutrale Schnittstellen etwaigen Klienten einen standardisierten Zugriff über wohldefinierte Schnittstellen. Auch im Rahmen des

Vermittlungsprozesses muss eine solche Anbindung unterstützt werden, wobei ggf. auf vorhandene Arbeiten zurückgegriffen werden kann.

Logische Adressierung

Die Grundidee einer logischen Adressierung ist die Unabhängigkeit von physischen Endpunktadressen, um eine technologie- und protokollübergreifende Kommunikation zu ermöglichen [HKS08]. Im Rahmen des Vermittlungsprozesses kommunizieren Produzenten und Konsumenten (sowie ggf. ihre Stellvertreter) mit Diensten eines Vermittlers (und umgekehrt) sowie Dienste eines Vermittlers untereinander. Gegenstand einer logischen Adressierung sind entsprechend die Geräte und Dienste. Um eine logische Adressierung zu realisieren, bedarf es einerseits einer (eindeutigen) Kennzeichnung von Entitäten und andererseits einer Möglichkeit anhand einer logischen Kennung einen konkreten Endpunkt aufzufinden.

Kennzeichnung für Endpunkte Produzenten und Konsumenten repräsentieren einmalige Instanzen und bedürfen daher einer eindeutigen Kennung. Im Gegensatz dazu kann ein Dienst, der von einem Vermittler angeboten wird, in einem verteilten System mehrfach instanziiert werden, um z.B. eine Lastverteilung zu ermöglichen. Hierbei spielt es unter Umständen keine Rolle, welche Instanz einen Auftrag verarbeitet und somit bedarf nicht jede einer eindeutigen Kennung, sondern nur der Diensttyp, bzw. die auszuführende Funktion [KC03]. Dem Problem der eindeutigen Kennzeichnung wird in der Praxis verschiedenartig begegnet:

TCP/IP-Adressen Geräte im Internet werden heutzutage durch eindeutige 32-Bit (IPv4) bzw. 128-Bit (IPv6) IP-Adressen identifiziert, welche durch die *International Corporation for Assigned Names and Numbers* (ICANN) vergeben werden. Auf der Transportschicht kommt z.B. mit dem TCP-Protokoll und der damit eingeführten *Port*-Nummer eine weitere Konkretisierung einzelner Dienstendpunkte, die auf den Geräten laufen, hinzu. Da die numerischen IP-Adressen für Menschen schlecht lesbar sind, wurde zudem mit dem *Domain Name System* (DNS) ein globaler Dienst etabliert, welcher Namen auf numerische IP-Adressen abbildet [CDK12, TS07, AIM10].

UUID Ein *Universally Unique Identifier* (UUID) wurde von der IETF zur eindeutigen Kennzeichnung beliebiger Entitäten in (verteilten) Software-Produkten standardisiert [Int05b]. Hierbei werden eindeutige Kennzeichner u.a. mittels einer lokalen Systemzeit, der eindeutigen Netzwerkkennung sowie weiteren (zufälligen) Werten oder Namen gebildet. Auf diese Weise lassen sich auch in verteilten Systemen sehr viele (bis zu 2^{122}) eindeutige, jedoch für Menschen schwer lesbare Kennungen erzeugen.

EPC Mit Hilfe eines eindeutigen *elektronischen Produktcodes* (EPC) können Objektinstanzen weltweit eindeutig identifiziert werden. Beispielsweise

enthalten RFID Tags solche Codes häufig als Seriennummern. Es gibt verschiedene Formen eines EPCs, welche je nach Länge Informationen über das Objekt, den Typ und dessen Hersteller tragen [EPC10]. *Global Standards One* (GS1) zeichnet sich für die Vergabe der Codes bzw. einzelner Namensräume verantwortlich, damit die Eindeutigkeit sichergestellt werden kann.

URI/IRI Ein *Uniform Resource Identifier* (URI) oder ein *Internationalized Resource Identifier* (IRI) dient der Identifikation einer beliebigen Art von Ressource. Die Struktur einer URI ist durch einen IETF-Standard geregelt und schreibt für eine Kennung ein *Schema*-Präfix zur Typidentifikation sowie eine Pfadangabe vor [Int05a]. Beispielsweise können sowohl WWW-Adressen als auch EPCs mittels einer URI repräsentiert werden.

Eine eindeutige Kennzeichnung von Endpunkten sollte technologieneutral erfolgen. Abgesehen von der sehr begrenzten Anzahl verfügbarer IP-Adressen, besitzt nicht jedes Gerät eine IP-kompatible Kommunikationsschnittstelle. Somit entfällt diese Möglichkeit der eindeutigen Identifizierung. Da eindeutige elektronische Produktcodes von einer zentralen Vergabestelle nur gegen Gebühr veräußert werden, entfällt auch diese Möglichkeit. URIs stellen für sich genommen keine eindeutigen Kennungen dar, bieten jedoch eine praktikable Möglichkeit diese zu repräsentieren, sind universal einsetzbar, leicht lesbar, verständlich und lassen sich nahezu beliebig erweitern [GBMP12].

Um dem Punkt der Eindeutigkeit zu entsprechen, soll daher im Rahmen des Vermittlungsprozesses eine logische Adressierung mittels als URI-kodierter UUIDs zum Einsatz kommen. Auf diese Weise lassen sich beliebige Entitäten logisch adressieren und durch Verwendung des URI-Schemas lassen sich zudem Geräte von beliebigen Diensttypen unterscheiden, was eine flexible Suche nach physischen Endpunkten zu gegebenen logischen Adressen ermöglicht. Für die Vergabe solcher eindeutigen Kennungen bedarf es einer Unterstützung seitens eines Vermittlers. Für die Konsumenten und Produzenten können Kennungen im Rahmen einer Registrierung von Metadaten erzeugt und diesen zugewiesen werden. Inwiefern diese Kennung als logische Adresse dem Endpunkt mitgeteilt werden muss, hängt vom jeweiligen Anwendungsfall ab. Lediglich mobile Endpunkte bedürfen der Kenntnis ihrer logischen Kennung, da diese im Falle eines Wechsels der physischen Endpunktadresse im Rahmen einer Aktualisierungsmeldung als Schlüssel einem Vermittler mitgeteilt werden muss. Andernfalls könnte dieser Vermittler lediglich versuchen einen mobilen Endpunkt anhand seiner Metadaten zu identifizieren, was nicht unbedingt eindeutig sein muss.

Auffinden konkreter Endpunkte Um einen Endpunkt zu kontaktieren werden, wie oben beschrieben, logische Adressen verwendet. Diese Adressen müssen vor Kontaktaufnahme auf die aktuelle physische Adressen abgebildet werden. Hierfür gibt es zwei wesentliche Ansätze:

Verzeichnis Ein Verzeichnis repräsentiert im einfachsten Fall eine Tabelle, welche logische und physische Adressen beinhaltet. Die logischen Adressen dienen dabei als Schlüssel bei einer Abfrage, deren Ergebnis die aktuelle physische Adresse liefert. Bei Änderung der physischen Adresse (z.B. bei mobilen Endpunkten), wird diese einfach als neuer Wert für die logische Adresse als Schlüssel gespeichert. Derartige Verzeichnisse lassen sich einfach auch verteilt realisieren, beispielsweise als verteilte Hash-Tabelle (siehe zum Beispiel *Chord* [SMK+01] oder *Pastry* [RD01]). Sollen zusätzlich zur physischen Adresse weitere (Meta-)Daten vorgehalten werden, zum Beispiel um eine attributbasierte Suche nach Endpunkten zu ermöglichen, können sogenannte *Verzeichnisdienste* Verwendung finden. Hier unterscheidet man zwischen sogenannten *Gelbe Seiten-*, *Weiße Seiten-* und *Grüne Seiten*-Diensten. Gelbe Seiten werden zum Auffinden von Diensten verwendet, sie enthalten konkretere Dienstinformationen, wie z.B. den Diensttyp. Weiße Seiten enthalten Informationen über das Objekt oder den Dienst selbst, z.B. die zugehörige Organisation, und Grüne Seiten stellen Informationen über den Zugriff auf das Objekt bzw. dessen Dienste zur Verfügung. Beispiele für solche Verzeichnisdienste sind unter anderem das *Domain Name System* [PM87], der *Object Naming Service* [EPC10], der *JADE Directory Facilitator* [BCTR10], der *Apache River Lookup Service* [Apa12a] und die *OASIS UDDI Registry* [Org12] sowie vom Open Geospatial Consortium der *Catalogue Service* [Ope07a], die *Sensor Instance Registry* [BEJ+11] und die *Sensor Observation Registry* [BEJ+11].

Broadcast Anstatt die Abbildung logischer in physische Adressen über ein Verzeichnis vorzunehmen, kann in bestimmten Fällen auch eine ad-hoc Suche nach diesen Daten sinnvoll sein. Dies trifft vor allem für die Dienste eines Vermittlers zu, da deren Verfügbarkeit einer hohen Dynamik unterliegen kann, was ein ständiges Aktualisieren der Verzeichnisse erfordern würde. Bei einer ad-hoc Suche werden im Bedarfsfall Broadcast- oder Multicast-Nachrichten [Com06, DGD05] im Netzwerk versendet. Empfänger einer solchen Nachricht können auf diese antworten oder die Anfrage einfach verwerfen. Um einen Endpunkt zu einer logischen Adresse ausfindig zu machen, kann man einfach eine Multicast-Anfrage mit der logischen Adresse versenden und alle Endpunkte (im Falle replizierter Dienste können das ggf. mehrere Endpunkte sein) antworten auf diese Anfrage mit ihrer physischen Adresse. Auf diese Weise erhält man nicht nur die aktuelle Endpunktadresse, sondern kann auch annehmen, dass der Endpunkt zur Zeit erreichbar ist.

Der Vermittlungsprozesses soll eine Unterstützung für beide Möglichkeiten des Auffindens konkreter Endpunkte bieten. Ein Verzeichnis ist unumgänglich, da Multicast-Nachrichten nur innerhalb lokaler Netze versandt werden können [Com06] und sich somit Konsumenten und Produzenten in den meisten Fällen nicht pull-basiert anfragen lassen. Auf der anderen Seite bietet

sich die ad-hoc Suche mittels Multicast-Nachrichten zum Auffinden konkreter Endpunkte auf Vermittlerseite an, sofern sich diese in einem Multicast-fähigen Netz befinden, da die kontinuierliche Aktualisierung eines Verzeichnisses entfällt, ein Nachrichtenaustausch nur bei konkretem Bedarf notwendig ist und die Verfügbarkeit des Endpunktes bei einer entsprechenden Antwort angenommen werden kann.

Pull- und Push-basierte Ereignis-Benachrichtigung

Ein Konsument kann auf zwei verschiedene Arten über Ereignisse informiert werden: Einerseits kann er selbst aktiv nachfragen, ob ein Ereignis (z.B. das Resultat einer Abfrage) für ihn vorliegt (*pull-basiert*), andererseits kann er ohne ständiges Nachfragen warten, bis er von einem Vermittler das Ereignis zugestellt bekommt (*push-basiert*) [DGD05, TS07]. Bei einer pull-basierten Abfrage bedarf es der Verwendung eines einfachen Anfrage/Antwort-Protokolls, was Konsument und Vermittler unterstützen müssen. Bei einer push-basierten Kommunikation muss der Konsument zum einen ständig bereit sein, eine Verbindungsanforderung seitens eines Vermittlers anzunehmen, zum anderen muss der Konsument bei einem Wechsel seiner physischen Endpunktadresse dies dem Vermittler mitteilen. Ersteres ist häufig problematisch, falls sich der Konsument entweder hinter einer Firewall oder in einem privaten Netz befindet (siehe auch Abschnitt 4.4.2), während Letzteres insbesondere bei mobilen Geräten mit häufigen Adresswechseln ein höheres Nachrichtenaufkommen für Aktualisierungsmeldungen zur Folge hat.

Gerade im Bereich der mobilen Geräte ist jedoch die push-basierte Kommunikation eine verbreitete Anforderung, weshalb sich hier von unterschiedlichen Seiten proprietäre Lösungen entwickelt haben. Hierzu gehören beispielsweise:

GCM *Google Cloud Messaging for Android* (GCM) ist ein Dienst der Firma Google Inc. speziell für Android-basierte Telefone [Goo12a]. Der Dienst macht sich zu Nutze, dass Android-Geräte von sich aus eine Verbindung zu Google-Diensten aufbauen und halten, über welche Nachrichten an das Gerät versendet und von diesem an die jeweilig adressierten lokalen Applikationen weitergeleitet werden.

APNs Der *Apple Push Notification Service* arbeitet nach dem gleichen Prinzip wie GCM [App12]. Im Gegensatz zu GCM ist es jedoch speziell auf Produkte der Firma Apple zugeschnitten und funktioniert nur mit iPhones, iPods und iPads.

BPS Auch der *BlackBerry Push Service* der Firma Research In Motion leistet wie obige Dienste eine push-basierte Zustellung von Nachrichten, allerdings ausschließlich für BlackBerry-Geräte [Res12].

Alle obigen Dienste sind auf spezielle Geräte zugeschnitten und setzen einen kontinuierlich verfügbaren externen Dienst voraus, welcher Nachrichten entgegennimmt, ggf. zwischenspeichert und bei vorhandener Verbindung zu einem Mobilgerät übermittelt. Da die einzelnen Geräte ohnehin eine mehr oder

weniger kontinuierliche Verbindung zu den Dienstanbietern (Google, Apple, Research in Motion) halten, eignet sich dieser Kanal für die Zustellung von Push-Nachrichten. Eine Technik, die dabei häufig eingesetzt wird, bezeichnet man auch als *Long Polling*. Wie der Name andeutet, handelt es sich hierbei nicht direkt um einen Push-Mechanismus. Stattdessen fragt ein Klient bei einem Dienst an, ob Nachrichten vorliegen (pull-basiert), die Antwort des Dienstes wird jedoch solange verzögert, bis wirklich eine Nachricht vorliegt oder ein Timeout eingetreten ist [CN10]. Ein Nachteil hierbei ist der Ressourcenverbrauch auf Seiten des Dienstes, da dieser für jeden Klienten eine Verbindung offen halten muss.

Im Rahmen des Vermittlungsprozesses bedarf es entsprechend einer Möglichkeit für Klienten auszuwählen, in welcher Weise sie Antworten bzw. Ereignisse von einem Vermittler zugestellt bekommen möchten: pull-basiert, per Long Polling oder push-basiert. Auf Seite eines Vermittlers muss ein entsprechender Nachrichtendienst existieren, welcher die Verwaltung der Kommunikationsbeziehungen sowie das Zwischenspeichern der Nachrichten übernimmt.

Mobilitätsverwaltung

Die Mobilitätsverwaltung eines Vermittlers stellt sicher, dass Nachrichten an mobile Klienten trotz wechselnder physischer Endpunktadressen zugestellt werden können [AGRS05], sofern eine push-basierte Nachrichtenübermittlung gefordert ist. Je nach Anforderung sind verschiedene Verfahren zur Mobilitätsverwaltung denkbar und werden in der Praxis angewandt. Nachfolgend sollen zwei Verfahren vorgestellt und im Anschluss daran ein Unterstützungspunkt, der den Anforderungen des Vermittlungsprozesses gerecht wird, detailliert werden.

- Die Mobilitätsverwaltung in zellulären Systemen, z.B. Mobilfunknetzen, beruht auf zwei verschiedenen Arten von Registern, welche Informationen über einzelne Teilnehmer führen: Das sogenannte *Home Location Register* (HLR) enthält grundlegende Informationen über alle Teilnehmer, die bei dem jeweiligen Anbieter registriert sind, während das *Visitor Location Register* (VLR) Informationen über alle in einem bestimmten Areal eingebuchten Teilnehmer führt [AGRS05]. Sobald ein Gerät eingeschaltet wird oder die aktuelle Basisstation bzw. Funkzelle wechselt, werden Informationen des aktuellen VLRs unter der Kennung des Mobilgerätes im HLR des Teilnehmers hinterlegt. Möchte man ein Mobilgerät erreichen, so wird eine Abfrage an das HLR des Teilnehmers gestellt, welches mit der Adresse des aktuellen VLR antwortet. Daraufhin wendet man sich an das VLR, welches die Abfrage an das eingebuchte Gerät weiterleitet. Kann ein Gerät nicht erreicht werde, z.B. weil die Abfrage während eines Handovers zwischen zwei Basisstationen erfolgte, so kann eine Suche bei allen VLRs der benachbarten Zellen durchgeführt werden.

- In IP-basierten Netzen wurde *Mobile IP* von der Internet Engineering Task Force (IETF) standardisiert [Int13]. Dieses Protokoll adressiert das Problem des Adresswechsels, sobald sich ein mobiles Gerät in anderen Netzen einbucht. Zu diesem Zweck werden einem mobilen Gerät zwei IP-Adressen zugewiesen: eine statische *Home Address* (HA) und eine wechselnde *Care-Of Address* (CA). Ein spezieller Dienst im Heimnetz eines Gerätes, der sogenannte *Home Agent*, führt eine Abbildung von HA auf CA, welche von dem mobilen Gerät bei Wechsel seiner CA aktualisiert wird. Um eine Nachricht an ein mobiles Gerät zu schicken, versieht man diese mit der permanenten HA des Gerätes. Die Nachricht wird im Heimnetz an den Home Agent übergeben, welcher anhand der Abbildung die entsprechende CA nachschlägt und die Nachricht dorthin weiterleitet [AGRS05].

Da für Produzenten und Konsumenten weder gefordert ist, dass sie einem zellulären Netz angehören, noch, dass sie das Mobile IP-Protokoll unterstützen, können existierende Mechanismen nicht direkt verwendet werden. Um mobile Geräte im Rahmen der Push-Kommunikation geeignet zu unterstützen, bedarf es daher ebenfalls eines Registers auf Seite eines Vermittlers, welches, ähnlich dem Home Location Register, Informationen über alle Klienten, unter anderem auch über deren aktuelle physische Endpunktadresse, führt. Dies muss nicht zwangsläufig ein eigenständiges Register sein, sondern kann im Rahmen der Metadatenverwaltung (siehe Abschnitt 5.4.5) erfolgen. Um eine eindeutige und statische Adressierung zu gewährleisten, bekommt jeder Teilnehmer, neben seiner (dynamischen) physischen Endpunktadresse eine statische logische Adresse (siehe oben) zugeteilt, ähnlich der Home Address und der Care-Of Address im Mobile IP-Protokoll.

Kann eine Nachricht, z.B. wegen eines gerade erfolgten Wechsels der physischen Endpunktadresse, nicht zugestellt werden, kann diese zwischengespeichert und nach einer bestimmten Zeit erneut zugestellt werden. Hierbei können dieselben Nachrichtenspeicher verwendet werden, die auch für das Bereitstellen der Nachrichten bei der pull-basierten Kommunikation erforderlich sind.

5.4.3. Datenverarbeitung

Nachdem Sensordaten erhoben wurden, bedürfen sie einer weiteren Verarbeitung. In der Literatur wird argumentiert, dass die Verarbeitung möglichst früh, also bereits auf den Sensorknoten selbst erfolgen sollte [SMZ07]. Sofern es sich um Verarbeitungsschritte handelt, die eine Reduktion der Datenmenge zum Ziel haben, ist dies auch sinnvoll, da hierdurch Energie der Sensorknoten gespart und somit die Lebensdauer eines Sensors verlängert werden kann [SMZ07, AV10]. Auch wenn Sensoren immer preisgünstiger und leistungsfähiger werden, sprechen jedoch einige Argumente dafür, die weitere Verarbeitung von Sensordaten in die Infrastruktur zu verlagern. Hier lassen sich Verarbeitungsfunktionen einfacher aufsetzen und warten, leistungsfähige

Geräte können die Verarbeitungsschritte schneller ausführen und zudem spielt die Verfügbarkeit von Ressourcen, vor allem Energie, nur eine untergeordnete Rolle.

Daher soll im Rahmen des Vermittlungsprozesses die Aufbereitung und weitergehende Verarbeitung von Sensordaten von einem Vermittler übernommen werden. Hierzu bedarf es einerseits entsprechender Komponenten, welche die erforderlichen Funktionen ausführen, und andererseits einer Möglichkeit der Beschreibung von Verarbeitungsketten, da Klienten eines Vermittlers unterschiedliche Anforderungen an die Verarbeitung von Sensordaten bzw. kontextbasierten Sichten stellen. Im Folgenden sollen daher die wesentlichen Unterstützungspunkte für die Datenverarbeitung erläutert werden.

Orchestrierung von Verarbeitungsfunktionen

Das Verarbeiten von Sensordaten bzw. das Erstellen kontextbasierter Sichten auf Basis von Sensordaten kann aus verschiedenen Verarbeitungsschritten bestehen, z.B. können Daten zuerst gefiltert, dann aggregiert und schließlich mit weiteren Daten externer Quellen angereichert werden. Da die Produzenten unabhängig von etwaigen Konsumenten sind, d.h. sie sind nicht auf einen bestimmten Anwendungsfall spezialisiert, kann die Art und Reihenfolge der Verarbeitungsschritte prinzipiell beliebig sein. Das wiederum impliziert, dass die Klienten explizit beschreiben müssen, wie ein Vermittler die Daten für sie aufbereiten soll. Eine solche Beschreibung wird auch als *Orchestrierung* bezeichnet, da der Ablauf einzelner Aktivitäten aus Sicht eines Akteurs repräsentiert wird [Pel03]. Hierfür ist eine Beschreibungssprache erforderlich, die folgende Möglichkeiten bieten muss:

- Benennung der auszuführenden Verarbeitungsschritte,
- Parametrisierung der Verarbeitungsschritte,
- Festlegung der Verarbeitungsreihenfolge,
- Konditionale Verzweigungen in der Verarbeitungsreihenfolge,
- Berücksichtigung von Ereignissen bei der Ausführung,
- Angabe möglicher Parallelisierungen und
- Ausnahmebehandlung bei der Verarbeitung

In der Praxis existieren eine Vielzahl verschiedener Beschreibungssprachen für die Orchestrierung von Diensten. Prominente und weit verbreitete Beispiele sind[3]:

BPMN Mit der *Business Process Model and Notation* (BPMN) der Object Management Group lassen sich Prozesse graphisch darstellen. Hauptziel ist die Verständlichkeit der Darstellung, daher abstrahiert diese Notation von

[3] Eine Übersicht weiterer Sprachen findet sich z.B. in [MTJ+10].

technischen Details und besitzt erst ab der 2011 veröffentlichten Version 2.0 eine Ausführungssemantik [Obj09, Obj11a].

WS-BPEL Die *Web Services Business Process Execution Language* (WS-BPEL), standardisiert durch die OASIS, ist eine Beschreibungssprache speziell für die Orchestrierung von Web Services. Das Hauptaugenmerk bei dieser Sprache liegt auf der Prozessausführung und entsprechend einer wohldefinierten Ausführungssemantik, sodass die Sprache von entsprechenden Workflow Engines interpretiert und die Prozessausführung automatisiert werden kann [Org07].

XPDL Die *XML Process Definition Language* (XPDL) der Workflow Management Coalition kann als eine Art Austauschformat für verschiedene Beschreibungssprachen verstanden werden. Sie enthält sowohl Elemente zur Repräsentation graphischer Informationen als auch Elemente für die Ablaufsteuerung [Wor12].

UML Die *Unified Modeling Language* (UML) der Object Management Group versteht sich als allgemeine Sprache zur Modellierung verschiedenster Aspekte von Softwaresystemen. Hierzu gehören auch die Aktivitätsdiagramme, mit Hilfe derer Abfolgen von Aktivitäten modelliert werden können [Obj11b].

Da alle obigen Beschreibungssprachen den Anforderungen zur Beschreibung von Verarbeitungsketten für Sensordaten genügen, hängt die Wahl einer Sprache einerseits von der Werkzeugunterstützung zur Erstellung von Beschreibungen und andererseits von den Plattformen zur Ausführung und Überwachung der Verarbeitungsketten ab. Da alle Sprachen in der Praxis weit verbreitet sind, finden sich auch für alle Sprachen ausgereifte Modellierungswerkzeuge, jedoch sind lediglich BPMN und BPEL ausführbar und entsprechend gibt es auch nur für diese Sprachen Ausführungsplattformen (für Übersichten siehe z.B. [KLL09, KH07, Obj12, Wik12b]).

Als Unterstützungspunkt im Rahmen der Datenverarbeitung im Vermittlungsprozess bedarf es daher lediglich einer Komponente, welche die Beschreibungen von Verarbeitungsketten entgegennehmen kann, diese an eine entsprechende Ausführungsplattform weiterleitet und das Ergebnis einer Ausführung an die nächste Instanz im Vermittlungsprozess weiterdelegiert.

Auffinden, Binden und Ausführen von Verarbeitungsdiensten

Wie oben bereits erwähnt, besteht eine Verarbeitungskette aus einem oder mehreren Verarbeitungsschritten, die in einer bestimmten Reihenfolge ausgeführt werden sollen. Da die Ausführung für den Klienten netzwerktransparent sein soll und eine lose Kopplung der dabei aufzurufenden Dienste gefordert ist (vgl. Abschnitt 5.3.2), wird vom Klienten lediglich die Angabe der Art des auszuführenden Dienstes sowie etwaiger Ausführungsparameter verlangt.

Der Aufruf bzw. das Erzeugen einer konkreten Dienstinstanz obliegt dann einem Vermittler. Hierfür bedarf es seitens des Vermittlers eines Mechanismus zum Auffinden und Binden eines konkreten Endpunktes, der den gewünschten Dienst erbringt. Erst dann kann der Aufruf des Dienstes erfolgen.

Dies erfordert als weiteren Unterstützungspunkt das Vorhandensein einer Art von Dienstverzeichnis, bei dem alle Knoten die von ihnen angebotenen Dienste registrieren. Bei Ausführung der Verarbeitungskette wird für jeden aufzurufenden Dienst eine Abfrage an dieses Verzeichnis gestellt, um einen konkreten Endpunkt zur Ausführung aufzufinden. Diesem werden dann alle zur Ausführung benötigten Informationen übergeben. Zudem unterstützt ein solches Verzeichnis den Klienten bei der Abfragestellung, da dieser hierüber Informationen über angebotene Dienste sowie deren Metadaten abfragen kann. Entsprechend kann sich ein solches Verzeichnis auch eng an die Metadatenverwaltung anlehnen bzw. durch diese realisiert werden.

Angebot typischer Verarbeitungsfunktionen

In Abschnitt 2.3 wurden bereits eine Reihe von Verarbeitungsfunktionen vorgestellt. Diese stellen allerdings nur einen Ausschnitt aller möglichen Funktionen zur Verarbeitung von Sensordaten dar, da die konkrete Auswahl immer domänen- und anwendungsabhängig ist. Allerdings lassen sich eine Reihe gängiger Funktionen identifizieren, welche in einer Vielzahl von Anwendungsszenarien benötigt werden. Zu solchen typischen Verarbeitungsfunktionen gehören beispielsweise Reduktions-, Aggregations- und Konversionsfunktionen, das Archivieren von Kontextdaten mittels unterschiedlicher Persistenzmechanismen, das Anreichern der Daten mit zusätzlichen Informationen aus externen Quellen sowie das Ableiten von Informationen mittels Inferenzmechanismen. Hinzu kommen sonstige mathematische und kryptographische Funktionen, das Sortieren von Daten oder das Extrahieren von Merkmalen aus Datenmengen, die Vorhersage von Trends sowie das Erzeugen von Visualisierungen etc. Für die meisten gängigen Funktionen existieren frei verfügbare Programmbibliotheken, für welche lediglich eine Schnittstelle bereitgestellt werden muss, sodass diese im Rahmen von Verarbeitungsketten referenziert und ausgeführt werden können.

Da, wie unten weiter ausgeführt, auch beliebige benutzerdefinierte Funktionen unterstützt werden sollen, stellt das Anbieten typischer Funktionen lediglich eine Erleichterung der Benutzung dar. Hierbei gilt: je größer das Angebot eines Vermittlers, desto größer die Wahrscheinlichkeit, dass eine benötigte Funktion nicht durch den Nutzer bereitgestellt werden muss.

Benutzerdefinierte Verarbeitungsfunktionen

Ein Vermittler kann zwar eine Reihe gängiger Verarbeitungsfunktionen anbieten, da derlei Funktionen jedoch nicht alle denkbaren Anwendungsszenarien und deren Anforderungen abdecken können, müssen im Rahmen der Datenverarbeitung auch benutzerdefinierte Verarbeitungsfunktionen aufgerufen wer-

den können. Solche Funktionen können entweder durch einen externen Dienst des Klienten realisiert oder einem Vermittler in Form einer Programmbibliothek bereitgestellt werden.

In ersterem Fall bedarf es einer Möglichkeit die externe Referenz in der Beschreibung der Verarbeitungskette unterzubringen. Bei der Ausführung der Verarbeitungskette stellt ein Vermittler dann keine Abfrage an das Dienstverzeichnis bzw. die Metadatenverwaltung, sondern ruft den vom Klienten angegebenen externen Dienst direkt auf. Im zweiten Fall muss der Klient einem Vermittler entweder den Programmcode zusammen mit der Beschreibung der Verarbeitungskette übermitteln oder einen Vermittler anweisen zur Laufzeit den Code aus einem Repository nachzuladen. Der Vermittler instanziiert dann bei der Ausführung der Verarbeitungskette den vom Klienten referenzierten Dienst und ruft diesen schließlich auf.

Zustandslose/-behaftete Verarbeitungsdienste

Im Rahmen der funktionalen Anforderungsanalyse wurde gefordert, dass neben zustandslosen Diensten auch Dienste, die einen internen Zustand fortschreiben, unterstützt werden sollen. Zustandslose Dienste sind einfach zu realisieren, da sie bei jedem Aufruf ggf. neu instanziiert werden können, sofern keine bereits laufende Instanz aufgefunden werden kann und der Programmcode vorliegt. Im Gegensatz dazu bringen zustandsbehaftete Verarbeitungsdienste im Rahmen des Vermittlungsprozesses einige Herausforderungen mit sich: Sofern der Prozess verteilt ausgeführt wird, kann ein Vermittler bestimmte Aufgaben, z.B. zur (Meta-)Daten- oder Prozessverwaltung, durch eine Vielzahl von Dienstinstanzen auf mehreren Knoten verteilt ausführen lassen, um zur Laufzeit einen Lastausgleich bei hohem Verarbeitungsaufkommen zu ermöglichen. Dasselbe gilt grundsätzlich auch für die Ausführung von Verarbeitungsdiensten nach Maßgabe der Produzenten und Konsumenten. Bei zustandslosen Diensten ist der Knoten, auf dem dieser letztlich ausgeführt wird, unerheblich und bei wiederholten Aufrufen des Dienstes kann dieser durchaus auf unterschiedlichen Knoten instanziiert werden. Bei zustandsbehafteten Diensten ist dies hingegen nicht ohne Weiteres möglich. Zwei Lösungsmöglichkeiten bieten sich für diesen Fall an:

1. Ein zustandsbehafteter Verarbeitungsdienst darf systemweit nur auf einem Knoten instanziiert werden und alle Aufrufe werden an diesen delegiert. Entweder muss der Dienst hierfür mit einem entsprechenden Hinweis in der Metadatenverwaltung versehen werden. Bei Aufruf einer Verarbeitungsfunktion wird dann in diesem zunächst überprüft, ob eine Dienstinstanz bereits existiert und ggf. eine neue Instanz erzeugt. Oder der Klient muss im Rahmen der Orchestrierungsbeschreibung, ähnlich der Referenzierung benutzerdefinierter Dienste (siehe oben), dem Dienst eine entsprechende Markierung hinzufügen. Durch Letzteres ließen sich auch dedizierte Instanzen für einzelne Klienten realisieren.

2. Der Zustand eines Dienstes wird nach jedem Aufruf in der Datenverwaltung (siehe Abschnitt 5.4.4) persistiert und von dort vor einem erneuten Aufruf wiederhergestellt. Auf diese Weise ist der Ausführungsort eines zustandsbehafteten Dienstes unerheblich.

In beiden Fällen muss eine geeignete Synchronisierung der Aufrufe sichergestellt werden, damit kritische Abschnitte, wie das Nachschlagen eines Dienstes in der Metadatenverwaltung sowie das Lesen und Schreiben eines persistierten Zustands, atomar erfolgen können. Andernfalls könnten Dienstinstanzen ungewollt mehrfach instanziiert werden oder inkonsistente Zustände auftreten.

5.4.4. Datenverwaltung

Aufgabe der Datenverwaltung ist das Persistieren und Bereitstellen von Kontext- bzw. Sensordaten. Während für die Datenhaltung von Metadaten (siehe Abschnitt 5.4.5) meist ein festes Schema verwendet werden kann, ist dies bei der Heterogenität von Kontextdaten nicht praktikabel. Auch müssen Kontextdaten nicht zwangsläufig persistiert werden, da häufig nur bestimmte Änderungen oder Abweichungen von einem Normalzustand für einen Klienten von Interesse sind. Aufgrund dieser Unterschiede gibt es auch verschiedene Unterstützungspunkte für den Umgang mit Kontext- bzw. Metadaten.

Verteiltes Persistieren von Zeitreihen

Ein Teil der Kontextdaten, hierzu gehören alle Daten, die von Sensoren erhoben werden, wird in Form kontinuierlicher Datenströme an einen Vermittler übergeben. Schon bei einer relativ geringen Anzahl an Sensoren entstehen so große Mengen an Zeitreihen, deren Handhabung durch zentralisierte Datenbanksysteme zumindest schwierig ist [PHPS10]. Es bedarf daher einer verteilten Speicherung der Daten. Bei der Verteilung können unterschiedliche Strategien angewendet werden, die Einfluss darauf haben, wie die Daten verteilt werden, ob sie repliziert oder fragmentiert werden und wie gut die Datenbanksysteme skalieren können.

Da verschiedenste Anforderungen an die Daten gestellt werden, sollen auch unterschiedliche Systeme zur Datenhaltung unterstützt werden (siehe Abschnitt 5.3.1.3). Die Umsetzung der Verteilungsstrategien obliegt den Datenbanksystemen und muss entsprechend von einem Vermittler nicht weitergehend unterstützt werden. Es bedarf daher lediglich einer generischen Schnittstelle für grundlegende Funktionen zum Speichern, welche von ein oder mehreren Knoten angeboten wird und die unabhängig vom dahinterliegenden Datenbanksystem sind.

Bereitstellen von Zeitreihen

Wie bereits erwähnt, erfolgt das Persistieren von Zeitreihen semi-strukturiert bzw. schemafrei. Das bedeutet, es ist nicht vorgeschrieben, welche Datenwer-

te ein Datensatz beinhalten muss und die Interpretation der Daten obliegt allein dem Abfragesteller. Daraus folgt, dass auch für das Bereitstellen der Daten eine generische Schnittstelle geboten werden muss. Hierbei muss zwischen Schnittstellen zur technologiespezifischen und zur technologieneutralen Abfrage unterschieden werden. Eine generische Schnittstelle zur technologiespezifischen Abfrage erwartet die Abfrage in einer zum verwendeten Datenhaltungssystem passenden Form, z.B. im Fall einer SQL-basierten Datenbank als Text. Das Formulieren syntaktisch korrekter Abfragen fällt hierbei komplett in den Verantwortungsbereich des Abfragenden. Eine generische, technologieneutrale Schnittstelle beruht, wie in [Gue11] beschrieben, auf einem kleinsten gemeinsamen Nenner gängiger Abfragesprachen und erlaubt für Abfragen lediglich die Spezifikation von Eingabe-, Filter- und Ausgabedefinitionen. Ungeachtet der Art der Schnittstelle sollten zudem auch Funktionen zur Abfrage von Metainformationen über die verwendeten Datenbanksysteme bereitstellen.

Semantische Abbildungen und Ableitungen

In Abschnitt 4.4.6 wurde auf mögliche Benutzbarkeitsprobleme bei der Formulierung syntaktisch korrekter Abfragen hingewiesen. Unter anderem um die Abfrageformulierung zu unterstützen, bedarf es daher einer Möglichkeit, semantische Abbildungen und Ableitungen durchzuführen, welche den Umgang mit Mehrdeutigkeiten, Generalisierungen und Spezialisierungen erlauben. Die langjährige Forschung in diesem Bereich hat eine Vielzahl an Möglichkeiten zur Integration semantischer Annotationen, Konzeptualisierungen und Inferenzmechanismen in bestehende Systeme hervorgebracht. Insbesondere auch für Kontextdaten existieren eine Reihe von Arbeiten (z.B. [JSB+09, PHPS10]), auf denen aufgebaut werden kann.

Grundlegend müssen Meta- und Kontextdaten sowie Abfragen semantisch annotiert werden, damit Abbildungen und Ableitungen vorgenommen werden können. Da dies nicht von allen Klienten verlangt werden kann, bedarf es eines Verzeichnisses, welches zu eindeutigen Stichwörtern (z.B. „Heute“, „Celsius“ etc.) entsprechende semantische Annotationen nachschlägt. Derartige Annotationen referenzieren Konzepte, d.h. eine allgemeine Übereinkunft über die Bedeutung eines Terms. Die Unterstützung von Konzeptualisierungen und Inferenzmechanismen beschränkt sich lediglich auf jene Daten, die annotiert wurden. Mit Hilfe von z.B. Ontologien, einer entsprechenden Abfragesprache, eines Inferenzmechanismus sowie ggf. einer speziellen Faktenbasis können schließlich semantische Abbildungen und Ableitungen vorgenommen werden.

Um dies transparent für den Klienten durchzuführen, bestehen die Unterstützungspunkte einerseits aus einem Annotationsverzeichnis, welches Stichwörter auf Annotationen abbilden kann, und andererseits aus einer Abfragevorverarbeitung, welche zwischen der Abfrageschnittstelle und den dahinter liegenden Datenbanksystemen anzusiedeln ist und ankommende Abfragen entgegennimmt, semantische Abbildungen oder Ableitungen durchführt

und dann an die Datenbanksysteme weiterleitet. Ein konkretes methodisches Vorgehen wird in [JSB+09] beschrieben. Eine alternative, praktische Umsetzung wird z.B. in [PHPS10] dargestellt.

5.4.5. Metadatenverwaltung

Die Verwaltung von Metadaten unterscheidet sich in mehrfacher Hinsicht von der Verwaltung von Zeitreihen. Metadaten repräsentieren meist strukturierte Informationen, sie können im Laufe der Zeit geändert bzw. aktualisiert werden und es wird häufiger lesend als schreibend auf diese Daten zugegriffen. Zudem ist das Datenvolumen, das verwaltet werden muss, weitaus geringer. Entsprechend eignen sich unter Umständen andere Technologien für deren Persistierung.

Verteiltes Persistieren von Metadaten

Obwohl das Gesamtvolumen an Metadaten weitaus geringer ausfällt als bei Zeitreihen, bietet sich auch hier eine verteilte Lösung zum Persistieren an. Der Grund hierfür ist, neben einer besseren Skalierbarkeit, vor allem die Robustheit verteilter Lösungen, sofern die Daten nicht fragmentiert, sondern redundant über die Knoten verteilt werden.

Während Zeitreihen beliebige Kontextdaten enthalten können, lassen sich bei Metadaten gängige Attribute ausmachen. Beispielsweise hat ein Sensor für gewöhnlich einen Bezeichner, einen Ort, ein Messintervall etc. Ein Dienst hat ebenfalls einen Bezeichner, eine Endpunktadresse, und Ein- und Ausgabeparameter, welche aus einer festgelegten Menge an Datentypen bestehen. Derlei Standardattribute erlauben zumindest eine eingeschränkte Validierung von Metadaten, bevor diese persistiert werden. Zusammen mit der bereits bei den Zeitreihen erwähnten generischen Schnittstelle stellt dies einen Unterstützungspunkt für einen Vermittler dar.

Bereitstellen von Metadaten

Auch für das Bereitstellen von Metadaten bedarf es einer entsprechenden Schnittstelle. Während diese bei Zeitreihen jedoch gänzlich generisch sein muss, kann die Schnittstelle für Metadaten durchaus das Vorhandensein gängiger Attribute ausnutzen und diese an der Schnittstelle für den einfacheren Zugriff direkt anbieten.

Ermittlung von Metadaten

Wie bereits in Abschnitt 4.4.6 erwähnt, gibt es im Wesentlichen zwei verschiedene Wege, wie Metadaten einem Vermittler bekannt gemacht werden: i) Die Klienten übermitteln von sich aus (push-basiert) die Daten an einen Vermittler oder ii) ein Vermittler sucht selbst nach Metadaten indem er periodisch entsprechende Anfragen im Netzwerk per Broadcast oder spezieller Discovery-Protokolle veröffentlicht.

Für die erste Variante bedarf es keines weiteren Unterstützungspunktes. Es müssen seitens eines Vermittlers lediglich entsprechende Schnittstellen, wie oben bereits beschrieben, bereitgestellt werden. Die zweite Variante muss durch eine entsprechende Komponente unterstützt werden, die in regelmäßigen Abständen Metadatenabfragen veröffentlicht. Je nach Einsatzzweck und Skopus eines Vermittlers kann dieses beispielsweise über *IP-Broadcasts* [Com06], das *Bluetooth SDP* [Blu10], ZigBee Discovery-Protokolle [Zig12] oder mittels spezieller Discovery-Verfahren für Geräte oder Dienste wie z.B. *UPnP* [UPn12] erfolgen. Nach Erhalt einer Metadatenabfrage antworten die Geräte bzw. Dienste mit ihren Metadaten, welche über oben erwähnte Schnittstellen einem Vermittler zugeführt werden.

5.4.6. Prozessverwaltung

Im Rahmen der Prozessverwaltung gilt es, den Vermittlungsprozess zu überwachen und administrative Aufgaben zu unterstützen. Hierzu gehören beispielsweise das Auffinden von Systemfehlern sowohl einzelner Komponenten als auch ganzer Verarbeitungsknoten, Gewährleistung des Schutzes von Daten und Diensten vor unberechtigten Zugriffen, die Verwaltung von Ressourcen sowie Funktionen zur Abrechnung von Dienstleistungen.

Logging & Accounting

Das Aufzeichnen von Dienstaufrufen, den Zugriffen auf Daten sowie ggf. detaillierte Meldungen von Ausführungszuständen bietet die Grundlage für eine Reihe höherwertiger Funktionen, wie z.B. das Testen und Debuggen sowie das Abrechnen von Dienstleistungen, aber es stellt auch eine wichtige Basis für die Erkennung unbefugter Zugriffe sowie Entscheidungen, z.B. zur optimalen Verwaltung von Ressourcen, dar [ED06, SHG+08].

Es existieren eine Reihe etablierter Rahmenwerke (z.B. *log4j* [Apa13c], *Java Logging API* [Ora13a], *slf4j* [Qua13b] oder *Logback* [Qua13a]) zur Integration von Logging-Funktionalitäten in eigene Anwendungen. Diese unterstützen unterschiedliche Granularitätslevel, Ausgabeformatierungen und Ausgabeströme. Eine wesentliche Herausforderung besteht auch hier in der Persistierung der unter Umständen sehr großen Datenmengen. Wie bereits erwähnt, bieten sich hier verteilte Datenhaltungslösungen an. Für manche Logging-Rahmenwerke existieren sogenannte *Appender*, die teils auch das direkte Festschreiben von Logeinträgen in nicht-relationale Datenbanken erlauben (z.B. für log4j und slf4j), wodurch große Mengen an Einträgen verteilt persistiert und abgefragt werden können.

Allerdings sind, je nach Anwendungszweck, gänzlich unterschiedliche Zugriffsmuster auf die Daten zu berücksichtigen, sodass die Art der Verteilung (Fragmentierung oder Replikation/Synchronisation der Daten) eine wichtige Rolle spielt. Einerseits bedarf es einer zeitnahen Analyse aktuell erhobener Logging-Daten für die Verwaltung von Ressourcen oder die Erkennung von

Angriffen, andererseits müssen z.B. für die Fehlersuche oder die Abrechnung große Mengen an Daten erst bei Bedarf bereitgestellt werden.

Damit höherwertige Dienstleistungen auf Basis der anfallenden Logging-Daten aufsetzen können, bedarf es daher als Unterstützungspunkt zunächst einer generischen Schnittstelle für den Abruf von Daten sowie eines geeigneten Persistenzmechanismus, welcher die unterschiedlichen Zugriffsanforderungen adäquat unterstützen kann.

Testing & Debugging

Die Praxis zeigt, dass Fehler in Softwaresystemen nahezu unumgänglich sind. Insbesondere interaktive Systeme, welche manuelle Eingaben akzeptieren sowie verteilte Systeme, in denen Aktivitäten parallel ausgeführt werden, bergen Risiken für Laufzeitfehler, welche durch gängige Methoden, z.B. der statischen Code-Analyse, schwer aufzudecken sind.

Ausgiebiges Testen ist daher eine wichtige Phase im Lebenszyklus jedes Systems [Som12]. Bereits erwähnt wurde die Forderung nach stark kohäsiven Komponenten, welche thematisch zusammenhängende Funktionen fokussieren. Derartige Komponenten lassen sich beispielsweise mittels sogenannter *Unit-Tests*, welche automatisierte Testdurchläufe erlauben, testen [GHKW08]. Da dies jedoch, wie auch die meisten anderen (nicht formalen) Tests, eine falsifizierende Methode ist, die nicht die Abwesenheit von Fehlern nachweisen kann, können zur Laufzeit dennoch immer wieder Fehler, z.B. aufgrund sogenannter *Race Conditions* [GHKW08], auftreten. Um derlei Fehler in einer nachträglichen Analyse des Laufzeitverhaltens aufzuspüren, bedarf es feingranularer Logging-Mechanismen (siehe oben) und geeigneter Test- und Debugging-Werkzeuge, mit Hilfe derer im Nachhinein die Gründe für Fehler gesucht werden können.

Hierfür bedarf es zunächst einer globalen Übereinkunft aller beteiligten Knoten hinsichtlich einer gemeinsamen Zeit, was entweder durch Integration entsprechender Protokolle zur Zeitsynchronisation oder der Einführung einer logischen Zeit erzielt werden kann. Während die globale Zeit eine Synchronisation aller Netzwerkknoten erfordert und nur eine Genauigkeit von wenigen Millisekunden bietet, kann eine logische Zeit, bei der lediglich eine *Happens-before*-Relation erforderlich ist, auf Nachrichtenebene umgesetzt werden. Hierbei inkrementieren Knoten, die an der Bearbeitung von Inhalten beteiligt sind, entsprechende Zeitstempel, sodass zumindest die Ablaufsequenz der Verarbeitung im Nachhinein ersichtlich ist. Ein solches Verfahren basierend auf *logischer Zeit* empfiehlt auch [Mat89].

Einen entsprechenden Unterstützungspunkt stellt die Integration von verteilten Logging-Mechanismen als Ergänzung zu den Logging-Rahmenwerken in wesentliche Komponenten des Vermittlungsprozesses dar. Diese könnten beispielsweise auf Basis von Vektoruhren [Lam78, Mat89] realisiert werden, welche als Metainformationen an auszutauschende Nachrichten angehängt und an geeigneten Punkten in einem entsprechenden Verzeichnis für die

spätere Analyse persistiert werden. Die Analyse der partiellen Ordnung von Nachrichten (ggf. inklusive der Inhalte) mittels entsprechender (graphischer) Werkzeuge stellt einen wichtigen Aspekt bei der Unterstützung des Testens und Debuggens dar [JLSU87].

Monitoring & Reporting

Das Monitoring & Reporting bezieht sich auf die Echtzeitüberwachung der Geschehnisse innerhalb des Systems sowie das Benachrichtigen administrativer Instanzen, sobald eine Abweichung vom normalen Systemzustand erkannt wird. Das Sammeln von Logging-Daten (siehe oben) ist hierfür eine fundamentale Voraussetzung. Durch das Formulieren von Regeln kann dann der normale Systemzustand beschrieben und die Regeln anhand der Logging-Daten zur Laufzeit ausgewertet werden.

Zur Formulierung der Regeln können unterschiedliche Herangehensweisen gewählt werden, beispielsweise *Composition Operators*, *Data Stream Query Language* oder *Production Rules* [FTR+10, Luc02]. Unabhängig von der Wahl der Sprache bedarf es einer Komponente, welche eine zeitnahe Überprüfung der Daten anhand der Regeln vornimmt und bei erkannten Abweichungen den Versand von Mitteilungen über eine Kommunikationskomponente initiiert. Diese stellen die beiden wesentlichen Unterstützungspunkte für das Monitoring & Reporting dar.

Zugriffskontrolle

Das System soll die Möglichkeit bieten, den Zugriff auf Daten und Dienste dahingehend einzuschränken, dass dieser nur autorisierten Instanzen gewährt wird. Da eine umfassende Zugriffskontrolle den Rahmen dieser Arbeit übersteigen würde, sollen lediglich grundlegende Unterstützungspunkte, die auch für andere Aufgaben des Vermittlungsprozesses benötigt werden, dargelegt werden.

Ein autorisierter Zugriff erfordert die Sicherstellung der Identität eines Klienten durch einen *Authentifikationsdienst* [ED06]. Wie bereits zuvor erläutert, bekommt jeder Klient eine logische Adresse zugewiesen, mittels derer er einerseits adressiert und andererseits eindeutig identifiziert werden kann. Da diese Identität öffentlich ist, bedarf es zusätzlich eines Geheimnisses, welches nur der Klient und der Vermittler kennen. Ein solches Geheimnis kann entweder durch etwas, das man weiß (z.B. ein Passwort), das man hat (z.B. eine Chipkarte), das man macht (z.B. Unterschrift) oder das man ist (z.B. biometrische Merkmale) repräsentiert werden [ED06]. Bei der Authentifikation muss der Klient seine Identität sowie das Geheimnis an den Vermittler übertragen und kann ggf. anschließend für den Zugriff auf bestimmte Daten und Dienste autorisiert werden.

Um Identitätsdiebstahl vorzubeugen und somit eine ggf. bereits erfolgte Autorisierung auszunutzen, müsste der Klient sich vor jedem Zugriff erneut authentifizieren. Um dieses Erfordernis zu umgehen werden in der Praxis

sogenannte *Single-Sing-On*-Lösungen verwendet [Wie07]. Hierbei bekommt der Klient bei erstmaliger Authentifizierung eine Zugriffsmarke (engl. *Token, Ticket*), übermittelt. Diese etabliert ein temporäres Geheimnis zwischen dem Klienten und dem Vermittler. Bei allen anschließenden Interaktionen entfällt eine erneute Authentifizierung zugunsten der Zugriffsmarke. Bei Bedarf kann auf diese Weise auch eine Anonymisierung des Zugriffs realisiert werden.

Die Autorisierung eines Zugriffs kann auf Ebene einzelner Nutzer erfolgen, wobei explizit vermerkt werden muss, für welche Daten und Dienste der Zugriff erlaubt ist. Da dies in mehrfacher Hinsicht einen beträchtlichen Aufwand darstellt, kann von einzelnen Nutzern abstrahiert und eine Zugriffserlaubnis auf Basis von Gruppen- oder Rollenzugehörigkeiten definiert werden (z.B. alle Administratoren einer Firma).

Vor einem Zugriff auf Daten oder Dienste muss schließlich ein *Autorisierungsdienst* anhand der Identität, bzw. der Zugriffsmarke, sowie ggf. der Gruppen- und Rollenzugehörigkeit des Klienten mittels sogenannter *Access Control Lists* über die Zugriffserlaubnis entscheiden [LSI10]. Auf diese Weise lassen sich einzelne Datensätze und Dienste vor unerlaubten Zugriff schützen. In Abschnitt 5.3.2 wurde zudem noch ein Schutz des Gesamtsystems vor unerlaubtem Zugriff (z.B. mittels direkter Adressierung vermittlungsinterner Dienste durch externe Klienten oder *Denial-of-Service*-Attacken) gefordert. Hierfür bedarf es einer Analyse der Dienstaufrufe bzw. der Aufrufreihenfolge für einzelne Klienten. Für das Testing & Debugging wurden die Verwendung logischer Zeitstempel für Nachrichten sowie für das Monitoring & Reporting eine Auswertung von benutzerdefinierten Regeln in nahezu Echtzeit als Unterstützungspunkte identifiziert. Beides zusammen bildet die Basis sogenannter *Trace Checker*, mit Hilfe derer die Aktionssequenz eines Klienten beobachtet und nachverfolgt werden kann. Durch Formulierung entsprechender Regeln (z.B. muss vor dem Aufruf des Autorisierungsdienstes der Authentifizierungsdienst aufgerufen worden sein) lassen sich einerseits unerlaubte Zugriffe aufdecken (siehe [Mat89]) und andererseits auch die Fehlersuche unterstützen.

Abrechnungsdienst

Für einen Einsatz in der Praxis kann es sinnvoll sein, die Kosten für den Betrieb der Infrastruktur eines Vermittlers auf die Klienten umzulegen. Hierfür ist es einerseits erforderlich, die Daten- und Dienstzugriffe zu protokollieren, und andererseits die Identität des Aufrufers nachzuweisen. Für beide Erfordernisse wurden oben bereits Unterstützungspunkte identifiziert. Zur Erstellung von Abrechnungen bedarf es lediglich eines weiteren Dienstes, bei dem Anbieter und Konsumenten ihre Abrechnungsinformationen hinterlegen können und welcher auf Basis der Zugriffsprotokolle entsprechende Abrechnungen erstellen kann.

Ressourcenverwaltung

Die Anforderungen einer flexiblen und gut skalierbaren Infrastruktur implizieren, dass die verwendeten und benötigten Ressourcen zur Laufzeit dem aktuellen Bedarf angepasst werden können. So können bei geringer Verarbeitungslast Ressourcen freigegeben werden, um z.B. Kosten zu sparen, und bei hoher Last neue Ressourcen akquiriert oder bestehende Ressourcen umgewidmet werden, um den Qualitätserfordernissen gerecht zu werden. Hierfür bedarf es einerseits der kontinuierlichen Beobachtung der aktuell verwendeten Ressourcen und deren Auslastung, sowie andererseits einer Möglichkeit, aktiv in die Infrastruktur einzugreifen, neue Ressourcen hinzuzufügen, nicht benötigte Ressourcen freizugeben und einzelne Dienste zwischen Ressourcen zu migrieren.

Für das (automatische) Akquirieren bzw. Freigeben von Ressourcen hat sich in der Praxis das *Cloud Computing*-Paradigma etabliert. Hierbei werden Seitens eines Anbieters Rechen-, Speicher- und Netzwerkkapazitäten angeboten, welche den Klienten in kürzester Zeit bereitgestellt und von diesen wieder freigegeben werden können. Eine Bezahlung, im Falle kommerzieller Anbieter, erfolgt ausschließlich für tatsächlich genutzte Ressourcen. Auf diese Weise müssen die Klienten keine eigenen Server oder Rechenzentren betreiben, sondern können den Betrieb der Infrastruktur sowie darin laufende Anwendungen in das Internet auslagern.

Man unterscheidet zwischen verschiedenen Cloud-Realisierungen [BBG10]:

Infrastruktur Bei *Infrastructure-as-a-Service* (IaaS) stellt der Anbieter lediglich die Ressourcen in Form (virtualisierter) Hardware sowie eine Schnittstelle zur Verwaltung der Ressourcen zur Verfügung.

Plattform Beim *Platform-as-a-Service*-Modell (PaaS) wird neben der technischen Infrastruktur noch zusätzlich eine spezielle Laufzeitumgebung zur Verfügung gestellt, welche die Instanziierung und Verteilung der Verarbeitungseinheiten einer Anwendung übernimmt. Hierfür muss die Anwendung jedoch speziell auf die Laufzeitumgebung zugeschnitten werden.

Anwendung *Software-as-a-Service* (SaaS) repräsentiert eine konkrete Anwendung (z.B. eine Tabellenkalkulation), welche in einer Cloud-Infrastruktur läuft und dem Klienten lediglich eine (graphische) Schnittstelle zur Interaktion bietet.

IaaS offeriert dem Klienten die größten Freiheitsgrade, indem es ihm die Verantwortung für die Ressourcenverwaltung sowie die Verteilung und Ausführung der Anwendung überlässt. Über entsprechende Schnittstellen kann dieser so Ressourcen (de-)allozieren und das Deployment von Anwendungen vornehmen. PaaS hingegen kann die Ressourcenverwaltung übernehmen, sofern der Klient seine Anwendung speziell auf die Plattform zugeschnitten hat. Da keine Standards für die Plattformen existieren, besteht jedoch die Gefahr des *Vendor Lock-In* [Sch10]. Das heißt, die Anwendung kann nicht ohne

größere Änderungen auf PaaS-Systemen anderer Anbieter installiert werden. SaaS ist im Kontext dieser Arbeit nicht weiter von Bedeutung, da es konkrete Anwendungen für den Endnutzer repräsentiert.

Damit auch der Vermittlungsprozess dynamisch Ressourcen (de-)allozieren kann, bietet es sich an, existierende IaaS- bzw. PaaS-Lösungen zu verwenden, da hier bereits das Monitoring sowie die Ressourcenverwaltung weitgehend realisiert sind. Neben sogenannten *Public Cloud*-Lösungen wie *Amazon Web Services* [Ama12], *Google Cloud Platform* [Goo12b], *Windows Azure* [Mic12] und *Salesforce* [Sal12], bei denen die Infrastruktur von kommerziellen Anbietern betrieben wird, existieren auch *Private Cloud*-Lösungen, welche den Betrieb in einer eigenen Infrastruktur erlauben. Hierzu gehören beispielsweise die *Eucalyptus Cloud* [Euc12], *OpenStack* [Ope12c] und *CloudStack* [Cit12].

Das Umwidmen von Ressourcen, also das Verschieben bzw. Migrieren einzelner Dienste wird von IaaS-Lösungen nicht geboten. Dies muss softwareseitig gelöst werden. Hierfür existieren z.B. mit mobilen Objekten und mobilen Agenten [Sat10, BR05, Whi97] Entwicklungsparadigmen, welche die Migration von zustandsbehafteten und in Ausführung befindlichen Programmteilen erlauben. Je nach verwendeter Programmiersprache zur Realisierung eines Vermittlers bzw. dessen Diensten ist dies jedoch nur eingeschränkt möglich. Möchte man zustandsbehaftete Entitäten verschieben, bedarf es als weiteren Unterstützungspunkt einer verteilten Datenbasis, in welcher der aktuelle Zustand persistiert und nach der Migration erneut geladen werden kann. Zustandslose Dienste können, sofern sie aktuell nicht ausgeführt werden, einfach beendet und an entfernter Stelle neu gestartet werden.

5.4.7. Datenabfrage

Wie in Abschnitt 4.4.7 beschrieben, lassen sich Abfragen bezüglich ihrer zeitlichen Dimension in historische Abfragen, ad-hoc Abfragen und ereignisbasierte Abfragen unterteilen. Zur Verarbeitung dieser Abfragetypen bedarf es vermittlerseitig individueller Unterstützungspunkte. Gleiches gilt für die Abfragestellung über unterschiedliche Medien wie Bild und Ton. Da die Abfragestellung für einen Konsumenten die wichtigste Interaktion mit einem Vermittler darstellt, das Formulieren einer syntaktisch und semantisch korrekten Abfrage aber durchaus anspruchsvoll sein kann, bedarf es hierfür noch einer Reihe weitergehender Unterstützungspunkte.

Historische Abfragen

Unter historischen Abfragen versteht man die klassischen Abfragen an ein Datenbankmanagementsystem (DBMS). In Abschnitt 5.3.1 wurde die Unterstützung beliebiger Datenmodelle von DBMS (relational, probabilistisch etc.) gefordert, da sich je nach Einsatzzweck unterschiedliche Modelle zum Persistieren und Abfragen der Daten eignen können. Allerdings ist die Formulierung einer Abfrage häufig vom jeweils adressierten DBMS abhängig und

aufgrund der besonderen Eigenschaften der Datenmodelle existiert keine einheitliche Abfragesprache für alle gebräuchlichen DBMS.

Es ist daher auch nicht einfach möglich, dem Konsumenten eine gänzlich generische Schnittstelle zur Abfrage beliebiger DBMS zu bieten, ohne die konkrete Formulierung der Abfrage in die Verantwortung des Konsumenten zu stellen (vgl. Abschnitt 5.4.4). Die Abfrage muss folglich vom Konsumenten in einer entsprechenden Abfragesprache verfasst und einem Vermittler übergeben werden. Hierfür sollte ein Vermittler benötigte Metainformationen über das zu adressierende DBMS zur Verfügung stellen (siehe Abschnitt 5.4.5).

Nach Übermittlung der Abfrage bedarf es ggf. einer Abfragevorverarbeitung auf Seite eines Vermittlers. Hier werden beispielsweise textuelle Informationen aus multimedialen Abfragen extrahiert, semantische Abbildungen und Ableitungen vorgenommen, aber auch eine Abfrageanalyse durchgeführt, um das zu adressierende DBMS zu ermitteln, an das die Abfrage zur Beantwortung weiterdelegiert werden kann.

Ad-hoc Abfragen

Ad-hoc Abfragen beziehen sich auf einen (möglichst) aktuellen Datenwert. Wie in Abschnitt 5.3.1.6 beschrieben, gibt es hierfür drei unterschiedliche Strategien:

1. Zurückliefern des letzten historischen Wertes
2. Direkte Abfrage beim Produzenten
3. Subskription einer ereignisbasierten Abfrage, um den nächsten zukünftigen Wert zu erhalten.

Die Auswahl der Strategie sollte dem Konsumenten überlassen werden und einem Vermittler zusammen mit der Abfrage mitgeteilt werden. Das Umwandeln einer ad-hoc Abfrage in eine historische bzw. ereignisbasierte Abfrage kann vom Vermittler im Rahmen der oben angesprochenen Abfragevorverarbeitung geschehen.

Das direkte Abfragen eines Produzenten muss von diesem speziell unterstützt werden, da nur bestimmte Produzenten das Tasking (siehe Abschnitt 4.4.8) bzw. eine Verwendung als sogenannte *Sensordatenbank* (siehe Abschnitt 4.4.4) unterstützen. Wie bei den historischen Abfragen auch, finden sich hierfür eine Reihe unterschiedlicher Abfragesprachen, sodass die Abfrageformulierung wiederum in die Verantwortung des Konsumenten fällt. Lediglich das Suchen, Adressieren und die letztendliche Kommunikation mit einem Produzenten oder der gewünschten Sensordatenbank kann durch einen Vermittler mittels der Metadatenverwaltung und Kommunikationsadapter unterstützt werden.

Ereignisbasierte Abfragen

Ereignisbasierte Abfragen sind in die Zukunft gerichtet. Das bedeutet, der Konsument wird jedes Mal benachrichtigt, sobald das von ihm spezifizierte Ereignis eingetreten ist. Im Gegensatz zu historischen Abfragen, die von einem DBMS beantwortet werden, werden ereignisbasierte Abfragen von sogenannten *Datenstrommanagementsystemen* (DSMS) (siehe Abschnitt 5.3.1.6) verarbeitet.

Die Abfragen werden hier subskribiert, bis sie vom Klienten explizit wieder de-subskribiert werden. Da sich jede subskribierte Abfrage auf die Leistung eines DSMS auswirkt, ist von Seiten eines Vermittlers dafür Sorge zu tragen, dass nicht weiter benötigte Abfragen von Zeit zu Zeit entfernt werden. Es gibt verschiedene Möglichkeiten dies zu realisieren. Beispielsweise können Abfragen, die sich auf Daten von Produzenten beziehen, die das System als inaktiv erachtet (z.B. weil diese sich explizit de-registriert haben), gefahrlos aus dem DSMS entfernt werden. Auch können Abfragen, deren Ergebnisse über einen längeren Zeitraum dem Konsumenten aufgrund Nicht-Erreichbarkeit nicht zugestellt werden konnten, de-subskribiert werden. Zusätzlich kann jede Abfrage mit einem Zeitstempel in der Zukunft versehen werden. Bei Erreichen dieses Zeitstempels wird die Abfrage entfernt, sofern der Konsument den Zeitstempel nicht rechtzeitig erneuern lässt (z.B. durch erneute Subskription der Abfrage). Das Entfernen einer Abfrage impliziert darüber hinaus die Benachrichtigung des Konsumenten.

Als Unterstützungspunkt ist somit eine Abfrageverwaltung erforderlich. Deren Aufgabe ist einerseits die Verwaltung des Lebenszyklus von Abfragen und andererseits die Zuordnung aller subskribierten Abfragen zu den entsprechenden Konsumenten. Insbesondere Letzteres ist wesentlich, da bei Eintreten eines Ereignisses, für welches Abfragen subskribiert sind, alle entsprechenden Konsumenten über das Ereignis informiert werden müssen.

Testunterstützung für ereignisbasierte Abfragen

Im Gegensatz zur syntaktischen Korrektheit lässt sich die semantische Korrektheit von Abfragen, die in die Zukunft gerichtet sind, nur schwerlich überprüfen. Während z.B. bei historischen Abfragen ein Ergebnis zeitnah zur Verfügung gestellt wird, können sich ereignisbasierte Abfragen über beliebige Zeiträume erstrecken. Möchte man beispielsweise informiert werden, sobald ein Temperatursensor im Wald eine ungewöhnlich hohe Temperatur misst, die auf einen Waldbrand hindeutet, wird man die semantische Korrektheit der Abfrage entweder durch einen Fehlalarm oder erst bei Eintreten des Ernstfalls verifizieren können.

Für den Test semantischer Korrektheit bedarf es daher auf Vermittlerseite einer Unterstützung für virtuelle Sensoren, welche physische Sensoren simulieren können. Mit Hilfe der Metadaten und ggf. historischer Zeitreihen eines Produzenten kann ein Vermittler eine Testunterstützung anbieten, die einem Konsumenten das Erstellen von Szenarien ermöglicht, in denen virtuelle Sen-

soren beispielsweise den Brand eines Waldes simulieren. Dabei ist zu beachten, dass simulierte Testdaten nicht mit den subskribierten Abfragen anderer Konsumenten interferieren.

Konsumentenseitige Abfrageverwaltung

Wie bereits erwähnt, muss eine ereignisbasierte Abfrage vom Konsumenten explizit de-subskribiert werden, sobald dieser an einem Ergebnis nicht mehr interessiert ist. Andernfalls kann sie von einem Vermittler nach einer gewissen Zeit automatisch, z.B. nach Ablauf eines Zeitstempels, invalidiert werden (siehe oben). Es bietet sich daher auch auf Konsumentenseite eine Verwaltungslösung für Abfragen an. Eine solche Abfrageverwaltung könnte Abfrageschablonen verwalten, vergangene Abfragen zur erneuten Abfragestellung vorhalten, rechtzeitig die Zeitstempel für bereits subskribierte Abfragen erneuern und den Konsumenten benachrichtigen, sobald eine Abfrage vom System automatisch entfernt wurde.

Ein solches Werkzeug für den Konsumenten ist nicht Teil des Vermittlungsprozesses, jedoch sollte ein Vermittler entsprechende Schnittstellen in seiner Abfrageverwaltung vorsehen, über die ein Konsument den aktuellen Status und ggf. auch eine Historie seiner Abfragen einsehen kann, und eine Komponente für die Benachrichtigung eines Konsumenten, sobald seine Abfragen automatisch de-subskribiert wurden.

Graphische Abfrageformulierung

Das Formulieren textueller Abfragen erfordert für gewöhnlich das Einhalten entsprechender Syntaxregeln, was für den durchschnittlichen Nutzer eine hohe Akzeptanzhürde darstellt. Insbesondere für weit verbreitete Abfragesprachen, z.B. die *Structured Query Language*, finden sich daher Werkzeuge zur Unterstützung einer graphischen Abfrageformulierung [FLZ11]. Diese machen sich die wohldefinierte Struktur einer Abfrage zu Nutze um dem Benutzer je nach Bedarf Vorschläge für jeweils relevante Teilaspekte einer Abfrage zu unterbreiten. Das Bereitstellen entsprechender Werkzeuge gehört nicht zum Vermittlungsprozess. Unterstützt werden können solche Werkzeuge aber durch das Bereitstellen von Metainformationen über verfügbare historische Datensätze, aktive Datenströme oder allgemein Produzenten in entsprechenden Formaten.

Multimediale Abfragen

Abfragen an Datenbank- oder Datenstrommanagementsysteme werden üblicherweise in textueller Form gestellt, wobei eine entsprechende Syntax vorgegeben ist. Bei einer Reihe von Anwendungsfällen kann es jedoch für den Konsumenten sinnvoll sein, einfache Abfragen mittels eines anderen Mediums, beispielsweise in akustischer oder visueller Form, zu stellen. Auf diese Weise kann ein Ort beispielsweise durch ein Bild, ein Lied durch einen akustischen

Fingerabdruck oder ein Text durch gesprochene Sprache repräsentiert werden. Eine Reihe frei verfügbarer Bibliotheken, insbesondere für den Umgang mit Audiodaten (z.B. *Java Speech API* [Ora13b], *musicg* [Mus12], *pHash* [KS12], *CMUSphinx*, [Car12]) widmen sich dieser Aufgabe.

Bevor eine Abfrage an das DBMS bzw. DSMS weitergeleitet werden kann, müssen die für die Abfrage relevanten Informationen aus dem Medium extrahiert werden. Gesprochene Sprache wird so beispielsweise in eine textuelle Form gebracht und anschließend weitergeleitet. Eine derartige Informationsextraktion bzw. Umkodierung ist auch Aufgabe der vormals erwähnten Abfragevorverarbeitung, die entsprechend der zu unterstützenden Medien geeignete Verarbeitungsfunktionen vorsehen muss.

5.4.8. Auftragsverwaltung

Produzenten können neben der Datenerhebung auch die Möglichkeit bieten Aufträge entgegenzunehmen und auszuführen (auch als *Tasking* bezeichnet). Mittels derartiger Aufträge können zum Beispiel der Vorgang der Datenerhebung konfiguriert, RFID Tags beschrieben oder im Fall von Smart Transducern eventuell vorhandene Aktuatoren gesteuert werden.

Delegation von Aufträgen

Ein Vermittler übernimmt in der Auftragsverarbeitung lediglich eine Delegationsfunktion und leitet an ihn übergebene Aufträge an die entsprechenden Ziele weiter. Dem Initiator eines Auftrags muss daher nicht nur das Ziel, sondern auch die Form des Auftrags bekannt sein. Durch die Verwendung logischer Adressen wird dem Auftraggeber jedoch die konkrete Adressierung durch einen Vermittler abgenommen. Beispielsweise muss ein Auftrag zum Beschreiben eines bestimmten RFID Tags nicht die konkrete Endpunktadresse des Lese/Schreib-Terminals enthalten, denn diese kann durch einen Vermittler zur Laufzeit ermittelt werden. Ist ein Ziel bei Auftragsstellung nicht erreichbar (zum Beispiel ein RFID Tag, was außerhalb des Lese/Schreib-Bereiches ist), kann ein Vermittler den Auftrag jedoch zwischenspeichern und zu einem späteren Zeitpunkt wiederholt versuchen zuzustellen. Auf diese Weise wird der Initiator davon entbunden bei nächster Verfügbarkeit des Ziels den Auftrag erneut abzuschicken.

Durchführbarkeitsanalyse

Der Sensor Web Enablement-Standard (siehe Abschnitt 5.4.9) berücksichtigt im Rahmen des Tasking die Autonomie der Produzenten dahingehend, dass es diesen freisteht einen Auftrag auszuführen. Entsprechend sieht der Standard eine Möglichkeit der Überprüfung vor, ob der Auftrag prinzipiell durchgeführt werden kann oder das Ziel den Auftrag überhaupt ausführen möchte. Eine solche Durchführbarkeitsanalyse wird vom Initiator des Auftrags angestoßen und ein Vermittler behält auch hier lediglich eine delegierende Rolle. Für den Fall,

dass ein Produzent keine derartige Analyse selbst durchführen kann, kann ein Vermittler jedoch versuchen, anhand der Metainformationen über den Produzenten die prinzipielle Durchführbarkeit zu bestätigen. Ein Auftrag, die Messperiode eines Sensors zu erhöhen, kann beispielsweise auch ohne Anfrage beim Produzenten falsifiziert werden, sofern ein Vermittler einen entsprechenden Eintrag über die minimale Auflösung des Messnehmers in den Metadaten findet.

Statusüberwachung

Wurde ein Auftrag von einem Produzenten angenommen, so muss dieser nicht sofort ausgeführt werden. Beispielsweise können mehrere länger andauernde Aufträge zeitnah bei einem Produzenten eintreffen und müssen von diesem in eine Warteschlange eingereiht werden. Entsprechend sieht der Sensor Web Enablement-Standard auch eine Möglichkeit der Statusabfrage vor. Sofern ein Vermittler einen Auftrag erfolgreich an einen Produzenten zustellen konnte, werden nachfolgende Statusabfragen lediglich weiter delegiert. Nur im Falle, dass ein Vermittler den Auftrag zwischenspeichern musste, weil der Produzent nicht erreichbar war, kann er eine Statusanfrage mit einer entsprechenden Meldung selbst beantworten.

5.4.9. Etablierte Standards

In Abschnitt 5.3.2 wurde gefordert, dass ein Vermittler die nicht-funktionale Anforderung der Offenheit realisieren soll. Das bedeutet, dass die Schnittstellen und Kommunikationsprotokolle der Vermittlungsdienste dokumentiert und veröffentlicht werden, damit einerseits Dritte das System erweitern oder einzelne Komponenten austauschen können und andererseits die Interoperabilität mit existierenden Systemen ermöglicht werden kann.

Insbesondere was die Schnittstellenbeschreibung, die Kommunikation und einzelne Vermittlungsdienste angeht, kann hier durch Verwendung etablierter Standards das Zusammenspiel mit anderen Systemen begünstigt werden. Bezüglich der Kommunikation wurde in Abschnitt 5.3.1.1 bereits festgestellt, dass es keine Übereinkunft hinsichtlich der verwendeten Technologien und Protokolle gibt. Hier bedarf es dem Bereitstellen von Kommunikationsadaptern, die eine entsprechende Übersetzung vornehmen können.

Für die Realisierung von Schnittstellen und für die Integration von Enterprise-Anwendungen haben sich im Internet zwei verschiedene Varianten sogenannter *Web Services* durchgesetzt [PZL08]. Man unterscheidet hier zwischen dem *WS*-Stack* und den *RESTful-Web Services*. Auf konzeptioneller Ebene sind die Unterschiede nur marginal, jedoch auf strategischer, architektonischer und technischer Ebene zeigen beide Varianten besondere Ausprägungen. So eignen sich RESTful-Web Services, wegen ihrer Einfachheit und Flexibilität insbesondere für die ad-hoc Integration von Anwendungen, während WS*-Web Services aufgrund ihrer Mächtigkeit vor allem im professionellen Unternehmenskontext eingesetzt werden [PZL08]. Idealerweise sollten

von einem Vermittler beide Varianten unterstützt werden, da, wie im Folgenden beschrieben, wichtige Standards des Sensor Web und des Internet der Dinge auf den WS*-Technologien aufbauen, REST-basierte Schnittstellen jedoch insbesondere mobilen und ressourcenbeschränkten Klienten zugutekommen.

Im Folgenden werden zwei Standardisierungsbestrebungen vorgestellt, deren Standards im Kontext dieser Arbeit von besonderer Bedeutung sind und daher gesondert berücksichtigt werden sollen, da sie für ein Zusammenspiel mit existierenden kontextverarbeitenden Systemen essentiell sind.

Integration der Standards des Sensor Webs

Das *Open Geospatial Consortium* (OGC) hat mit der Sensor Web Enablement-Initiative (SWE) eine Sammlung von Standards für das Sensor Web erarbeitet [BEJ+11, BPRD08, Ope08b]. Diese Standards umfassen einerseits Datenmodelle, welche zum Beispiel Sensoren, Aufträge und Nachrichteninhalte beschreiben, und andererseits Schnittstellenmodelle, welche einzelne Dienste und ihre Funktionen beschreiben (siehe auch Abschnitt 6.1.2).

Die Datenmodelle beruhen alle auf XML-Schemata, welche beispielsweise verpflichtende und optionale Attribute für Sensorbeschreibungen, Messwerte oder Aufträge festlegen. Die Schnittstellenmodelle beruhen auf oben erwähnten WS*-Technologien, wobei aufgrund der Komplexität des Standards und der damit einhergehenden zögerlichen Annahme in der Praxis zur Zeit auch REST-basierte Modelle entworfen werden.

Die SWE-Standards, insbesondere die Schnittstellenmodelle, beziehen sich auf eine Reihe von Unterstützungspunkten, die im Rahmen dieses Abschnittes bereits identifiziert wurden. So existieren Modelle für ein Metadatenverzeichnis (*Sensor Instance Registry*), für eine Verwaltung von Messungen (*Sensor Observation Registry*), einen allgemeinen Verzeichnisdienst (*OGC Catalogue Service*), eine Auftragsverwaltung (*Sensor Planning Service*) sowie einen Benachrichtigungsdienst (*Sensor Alert Service* bzw. *Web Notification Service*).

Aufgrund der Bedeutung dieser Standards für das Sensor Web ist es unabdingbar auch im Rahmen dieser Arbeit eine möglichst weitreichende Konformität zu diesen Standards sicherzustellen. Hierfür ist zunächst zu prüfen, welche Unterstützungspunkte von diesen Standards tangiert werden (siehe Abschnitt 6.1.2) und welche Unterstützungspunkte ggf. von existierenden Implementierungen dieser Standards profitieren können.

Integration der Standards des Internet der Dinge

Auch für das Internet der Dinge hat eine Organisation Standardisierungsbemühungen unternommen. *EPCglobal* bzw. *Global Standards One* ist ein Konsortium verschiedener Firmen und Organisationen, welches ebenfalls eine Reihe von Daten- und Schnittstellenmodellen standardisiert hat [EPC10, EPC12].

Auch hier beruhen die Datenmodelle auf XML-Schemata sowie die Schnittstellenmodelle auf den WS*-Web Services. Im Gegensatz zu SWE, welches ins-

besondere den Austausch von Sensorinformationen anstrebt, fokussieren diese Modelle allerdings die Verarbeitung und den Austausch von Objektdaten in globalen, organisationsübergreifenden Netzen (siehe Abschnitt 6.2.1).

Hierfür wurden unter anderem Datenmodelle für Objekte standardisiert, ebenso wie Kommunikationsadapter für RFID-Lesegeräte, Dienste zum Überführen von Objektkennungen in andere ID-Schemata, ein Repository für Objektdaten sowie ein Namensdienst, der Objektkennungen zu Repository-Adressen auflösen kann. Auch hier ist zu prüfen, welche Unterstützungspunkte seitens eines Vermittlers von den Standards tangiert werden (siehe Abschnitt 6.2.1), wie diese Standards mit den SWE-Standards zusammenspielen und ob ggf. existierende Implementierungen bei der Umsetzung der Vermittlungsdienste integriert werden können.

5.5. Zusammenfassung

Im Rahmen dieses Kapitels wurde aufbauend auf dem Prozess der Kontextdatenvermittlung (siehe Kapitel 4) der Begriff der *kontextbasierten Sicht* eingeführt. Eine solche Sicht präsentiert einem Konsumenten alle für ihn relevanten Kontextinformationen einer Entität oder einer Situation in einer für ihn geeigneten Repräsentation. Somit ist es diesem möglich, die Informationen ohne weiteren Integrationsaufwand zu verarbeiten, was insbesondere mobile und ressourcenbeschränkte Klienten entlastet, aber auch den Umgang mit der Heterogenität auf Seiten der Produzenten und Konsumenten erlaubt. Während also eine kontextbasierte Sicht das endgültige Resultat einer Vermittlung beschreibt, repräsentiert der Vermittlungsprozess den Weg zur Erzeugung dieses Resultats.

Das Erstellen einer solcher Sicht aus den Kontextdaten der Produzenten umfasst eine Reihe von Verarbeitungsschritten, welche durch einen Vermittler ausgeführt werden. Im Rahmen einer Anforderungsanalyse für den gesamten Vermittlungsprozess und das Erstellen kontextbasierter Sichten wurden in diesem Kapitel daher verpflichtende und optionale Funktionalitäten eines Vermittlers sowie nicht-funktionale Qualitätskriterien erarbeitet.

Anhand dieses Anforderungskatalogs wurden zu bewältigende Teilprobleme in allen Aufgabenbereichen identifiziert und existierende Realisierungsansätze diskutiert. Die sich hieraus ergebenden Unterstützungspunkte konkretisieren die Anforderungen und dienen zunächst der Bewertung verwandter Arbeiten aus den Bereichen des Sensor Webs und des Internet der Dinge im folgenden Kapitel und anschließend als Basis für den Entwurf und die prototypische Implementierung eines Vermittlers.

6. Bewertung und Vergleich verwandter Arbeiten

Nachdem im vorigen Kapitel das Konzept der Aufbereitung und Vermittlung kontextbasierter Sichten vorgestellt, eine Anforderungsanalyse durchgeführt und Probleme sowie Lösungsansätze identifiziert wurden, sollen in diesem Kapitel verwandte Arbeiten betrachtet und hinsichtlich der Anforderungen bewertet und miteinander verglichen werden.

Bezüglich der verschiedenen Aufgabenbereiche der Kontextdatenvermittlung (vgl. Kapitel 4) findet sich keine verwandte Arbeit, die einen ganzheitlichen Blick auf alle Aspekte der Aufbereitung und Vermittlung kontextbasierter Sichten wirft. Stattdessen existieren für jeden einzelnen Bereich verschiedene Arbeiten, die jeweils einen bestimmten Fokus haben und die bereits in den Abschnitten 4.4.1 bis 4.4.8 vorgestellt wurden.

In diesem Kapitel sollen daher nur Ansätze betrachtet werden, die mehrere Aufgabenbereiche der Vermittlung berücksichtigen. Hier finden sich einerseits eine Reihe von Arbeiten, die sich im Rahmen des Sensor Webs mit der Distribution und Verarbeitung von Kontextdaten beschäftigen, sowie andererseits Arbeiten, die insbesondere das Internet der Dinge fokussieren, und somit einen Schwerpunkt auf der Vermittlung von RFID-Leseereignissen und den speziellen Anforderungen dieser Kategorie von Kontextdaten setzen.

Die Unterscheidung zwischen Kontextdaten im Allgemeinen und RFID-bezogenen Daten im Speziellen ist aus Anwendungssicht zwar unerheblich, jedoch entstehen aus der unterschiedlichen Art der Erhebung dieser Daten besondere Anforderungen hinsichtlich der Vermittlung. Aus Sicht der vermittelnden Infrastruktur produzieren RFID-Lesegeräte nämlich aperiodische Ereignisse, wohingegen andere Sensoren kontinuierliche Ströme periodischer Ereignisse erzeugen und somit unterschiedliche Lastcharakteristika aufweisen.

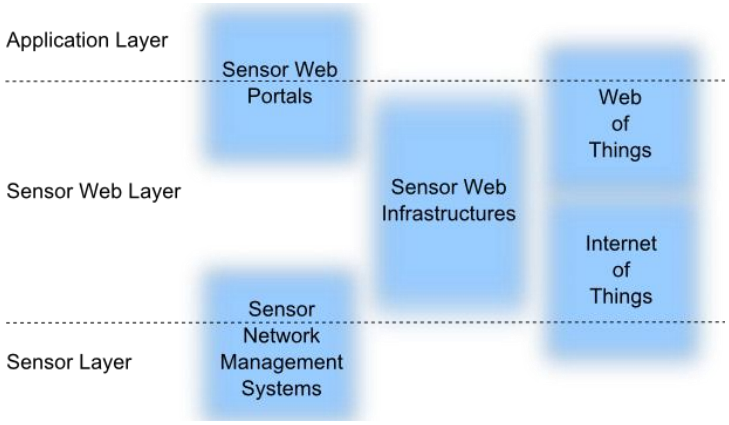


Abbildung 6.1.: Sensor Web Layer Stack (BEJ+11)

In [BEJ+11] findet sich eine ähnliche Unterscheidung: Abbildung 6.1 zeigt verschiedene Middleware-Klassen und ihre Einordnung in den sogenannten *Sensor Web Layer Stack*. Die diffusen Kanten der einzelnen Klassen deuten an, dass eine strikte Abgrenzung oft nicht möglich ist und Arbeiten in einer Klasse durchaus auch Aspekte der anderen Schichten betreffen.

So fokussiert die Klasse der *Sensor Web Portals* zwar Abfrage und Bereitstellung von Kontextdaten, greift hierfür aber auch auf Konzepte der Sensor Web-Schicht zurück. Ein Großteil der Arbeiten in diesem Bereich erlaubt es Benutzern, ihre eigenen Messreihen über ein Web Interface oder eine API einer zentralen Portalseite zur Verfügung zu stellen und Daten anderer Nutzer abzurufen. Das Hauptaugenmerk liegt hier auf benutzungsfreundlicher Bedienung und Datenpräsentation [BEJ+11].

Die *Sensor Network Management Systems* konzentrieren sich auf reine Verwaltungsangelegenheiten auf unterster Ebene des Sensor Web. *Wireless Sensor Networks* (WSN) sind auf dieser Ebene von besonderem Interesse und Arbeiten über Middleware-Systeme beschäftigen sich im Wesentlichen mit Aspekten wie Routing-Protokollen, Energiesparmaßnahmen, Lokalisierung von Sensoren innerhalb eines WSNs und dem Deployment von Anwendungen [BEJ+11].

Die *Sensor Web Infrastructures* sind in dieser Arbeit, wie bereits erwähnt, von besonderem Interesse, da sie eine wesentliche Rolle bei der Vermittlung von Kontextdaten zwischen der untersten Schicht und der Anwendungsschicht spielen. Häufig abstrahieren Arbeiten dieser Klasse von den darunter liegenden technischen Details der Sensornetzwerk-Verwaltung und bieten auch keine Konzepte für die Präsentation von Kontextdaten in Anwendungen oder für Benutzer [BEJ+11]. Verwandte Arbeiten dieser Klasse werden im folgenden Abschnitt 6.1 näher vorgestellt.

Die beiden Anwendungsklassen *Internet of Things* und *Web of Things* konzentrieren sich im Wesentlichen auf die virtuelle Repräsentation physischer Objekte durch universell eindeutige Bezeichner und Metadatenverzeichnisse. Während sich Arbeiten im Bereich des Internet der Dinge eher technischen Aspekten widmen, fokussieren Ansätze im Bereich des Web der Dinge eher Verarbeitungs- und Bereitstellungsaspekte [BEJ+11]. Der letztgenannte Bereich ist auch in dieser Arbeit von besonderer Bedeutung und eine Auswahl entsprechender verwandter Arbeiten findet sich in Abschnitt 6.2.

Den Abschluss dieses Kapitels bildet Abschnitt 6.3. Hier werden alle vorgestellten Arbeiten hinsichtlich geeigneter Kriterien des Anforderungskatalogs aus Abschnitt 5.3 bewertet und die jeweiligen Ansätze und Ziele dem Konzept der Aufbereitung und Vermittlung kontextbasierter Sichten gegenübergestellt.

6.1. Middleware-Plattformen zur Kontextdatenvermittlung

Im Folgenden werden eine Reihe von Sensor Web-Infrastrukturen vorgestellt, wobei insbesondere ein Fokus auf deren Architektur sowie auf die durch diese Arbeiten eingeführten Abstraktionen gelegt wird. Eine qualitative Bewertung der Arbeiten erfolgt in Abschnitt 6.3.

6.1.1. Global Sensor Network

Das *Global Sensor Network* (GSN) ist eine Plattform zur weltweiten Vernetzung von Sensoren und Sensornetzwerken [AHS06b, AHS06c, AHS06a, AHS07]. Ziel dieses Vorhabens ist die Entwicklung einer skalierbaren Infrastruktur, um heterogene Technologien aus dem Bereich der Sensornetzwerke zu integrieren. In Anlehnung an das Erfolgsrezept des Internets, wurden hierfür eine Menge einfacher logischer Abstraktionen und grundlegende Kommunikationsprotokolle entwickelt, um Interessenten den weltweiten Zugriff auf Sensoren und Sensordaten zu ermöglichen.

Ein wichtiger Aspekt stellt bei GSN beispielsweise das aus dem Bereich der Datenbanken bekannte Prinzip der *Datenunabhängigkeit* dar [KE11]. Dies bedeutet, dass der Zugriff auf Sensordaten unabhängig von der physischen Infrastruktur geschehen soll. Man spricht in diesem Fall auch von *Zugriffstransparenz* [TS07]. Auf diese Weise soll die Heterogenität im Bereich der Sensoren und Sensornetze vor dem Benutzer verborgen werden. Hierfür führt GSN die Abstraktion eines *virtuellen Sensors* ein, welcher eine logische Sicht auf Sensornetze erlaubt und maßgeblich zur Datenunabhängigkeit beiträgt. Ein virtueller Sensor agiert als Stellvertreter oder Repräsentant einer oder mehrerer Datenquellen (Sensornetze oder andere virtuelle Sensoren) und versteckt die Details des physischen Zugriffs hinter einer generischen Schnittstelle. Zudem kann ein virtueller Sensor eintreffende Daten filtern oder aggregieren, um so eine bestimmte Sicht auf die Datenquelle abzuleiten.

Virtuelle Sensoren wiederum sind in sogenannten *Containern* enthalten, welche letztlich auf den physischen Knoten im globalen Sensornetzwerk ausgeführt werden und sich im Wesentlichen für die Ressourcenverwaltung, die Abfrageverarbeitung und die Kommunikation verantwortlich zeichnen. Im globalen Sensornetz sind diese Container untereinander in einer Peer-To-Peer-Topologie organisiert und können über dieses Overlay-Netzwerk Abfragen und Daten weltweit verteilen.

Abbildung 6.2 stellt die Architektur eines Containers und die von diesem gebotenen Funktionen dar. Auf unterster Ebene finden sich die virtuellen Sensoren, die eine Reihe von Metadaten (z.B. Ein-/Ausgabeformate und Endpunktadressen) sowie eventuelle Filter- oder Aggregationsfunktionen spezifizieren. Darüberliegend sorgt ein *Virtual Sensor Manager* (VSM) für die Verwaltung des Lebenszyklus virtueller Sensoren (Instanziierung, Terminierung und Ressourcenallokationen) sowie die Einhaltung qualitativer Verbindungskriterien (Verbindungsabbrüche, Verzögerungen, fehlende Daten etc.). Daten, die einen virtuellen Sensor passieren, werden schließlich vom VSM über eine Persistenzschicht an den *Query Manager* weitergeleitet. Dieser ist für die Verwaltung und Verarbeitung von Nutzerabfragen sowie die Auslieferung des Abfrageergebnisses an den Nutzer zuständig. Die Kommunikation erfolgt letztlich über die eigentliche Dienstschnittstelle (*Interface Layer)*, welche Zugriffsfunktionen für Nutzer oder andere GSN Container bietet. Diese Schnittstelle wird durch die Zugriffskontrolle (*Access Control*) vor nicht-autorisierten Zugriffen geschützt.

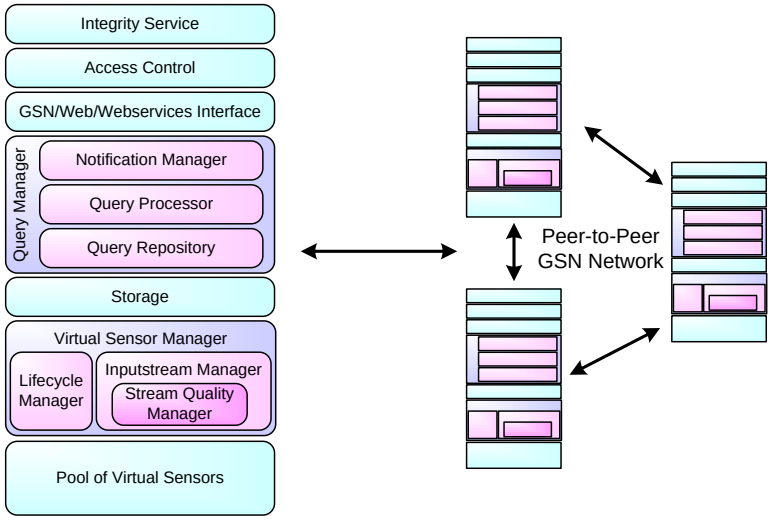


Abbildung 6.2.: Architektur eines GSN-Containers (nach (AHS06b))

Auf oberster Ebene sorgt zudem ein Integritätsdienst (*Integrity Service*) für die Vertraulichkeit und Integrität der versendeten Daten durch Anwendung elektronischer Signaturen.

Damit ein Sensor oder Sensornetz an dem globalen Netzwerk teilnehmen kann, muss hierfür eine Beschreibung in Form einer deklarativen Spezifikation der Ein- und Ausgabedaten sowie einer Reihe von Metadaten erfolgen. Mittels einer derartigen Beschreibung wird ein Sensor dann bei einem verteilten Verzeichnisdienst indiziert und steht fortan zur Abfrage von Daten bereit. Ein Interessent wendet sich auf der Suche nach Daten an diesen verteilten Verzeichnisdienst, erhält eine Beschreibung des virtuellen Sensors sowie dessen Containers und kann anschließend über die vom Container angebotenen Dienste mit dem Sensor oder Sensornetz kommunizieren. Darüber hinaus ist es möglich, virtuelle Sensoren zu kaskadieren und in jeder Kaskade eine Filterung oder Aggregation von ein oder mehreren Datenströmen anderer virtueller Sensoren vorzunehmen. Ähnlich wie in einer service-orientierten Architektur können somit die Sensoren und Sensornetze zu Diensten abstrahiert und beliebig orchestriert werden, um aus einer Reihe primitiver Messwerte höherwertige Informationen abzuleiten.

6.1.2. Sensor Web Enablement

Die *Sensor Web Enablement*-Initiative (SWE) des *Open Geospatial Consortiums* (OGC) erarbeitet ein Rahmenwerk offener Standards, um mittelfristig die vorherrschenden proprietären Lösungen existierender Sensor Web-Projekte abzulösen [BEJ+11, BPRD08, Ope08b]. Die hierfür vorgeschlagenen Modelle, Enkodierungen und Dienstschnittstellen der SWE-Architektur sollen die Umsetzung von interoperablen und skalierbaren dienstorientierten Netzwerken aus heterogenen Sensorsystemen und Anwendungen ermöglichen. Dabei

adressiert die Initiative eine Reihe typischer Funktionen eines Sensor Webs. Hierzu gehören insbesondere [Ope08b]:

- Auffinden von Sensorsystemen, Messdaten und sogenannten Beobachtungsprozessen entsprechend den Bedürfnissen von Anwendungen und Nutzern.
- Ermitteln der Eigenschaften von Sensoren und der Zuverlässigkeit von Messdaten.
- Zugriff auf Sensorparameter und -prozesse, die es einer Anwendung erlauben, Messdaten ohne weitere Kenntnis des Sensorsystems zu verarbeiten.
- Abfragen von Messdaten oder Zeitreihen in einem standardisierten Format.
- Beauftragen (engl. *Tasking*) von Sensoren, um Messungen bestimmter Aspekte zu erlauben oder den Fokus der Messungen zu ändern.
- Abonnieren und Veröffentlichen von Warnungen bzw. Ereignissen, sobald Messdaten bestimmten Kriterien entsprechen.

Um diese Ziele zu erreichen, hat SWE drei verschiedene Informationsmodelle zur Beschreibung von Sensoren und Sensormessungen sowie vier verschiedene Schnittstellenmodelle entworfen [Ope08b].

Sensor Model Language (SensorML) Standardmodelle und XML-Schema zur Beschreibung von Sensorsystemen und Beobachtungsprozessen.

Observations and Measurements Schema (O&M) Standardmodelle und XML-Schema zur Enkodierung von Beobachtungen und Messungen.

Transducer Model Language (TransducerML) Konzeptueller Ansatz und XML-Schema zur Unterstützung von Echtzeit-Datenströmen.

Sensor Observation Service Web Service-Schnittstellendefinition für die Abfrage und Filterung von Messdaten.

Sensor Planning Service (SPS) Web Service-Schnittstellendefinition für die Anforderung benutzerdefinierter Messreihen.

Sensor Alert Service (SAS) Web Service-Schnittstellendefinition zum Abonnieren und Veröffentlichen von Warnungen bzw. Ereignissen.

Web Notification Service (WNS) Web Service-Schnittstellendefinition für die asynchrone Auslieferung von Messdaten und Warnungen eines der zuvor genannten Web Services.

Mit dem *New Generation Sensor Web* [BEJ+11] hat die OGC das Rahmenwerk in Teilen überarbeitet. Das O&M- sowie das SensorML-Modell in der Version 2.0 basieren nun auf einem gemeinsamen *SWE Common*-Standard.

In diesem Zuge wurden auch die konzeptuellen von den Implementationsmodellen separiert, sodass neben XML- auch weitere Kodierungen (z.B. die leichtgewichtige *JavaScript Object Notation* [Cro13]) verwendet werden können. Die TransducerML wurde aus dem Rahmenwerk entfernt, da sich kein praktischer Nutzen abzeichnete. Stattdessen wurde mit der *Event Pattern Markup Language* (EPML) ein neues Informationsmodell für den ebenfalls neuen *Sensor Event Service* (SES) eingeführt. Der SES erweitert und ersetzt den SAS, damit zukünftig auch komplexe Bedingungen in Form von EPML-Ausdrücken für ein Abonnement von Benachrichtigungen verwendet werden können. Außerdem wurden mit der *Sensor Instance Registry* (SIR) sowie der *Sensor Observation Registry* (SOR) zwei neue Schnittstellenmodelle standardisiert, die das Auffinden von Sensormetadaten und geeigneten Messdatensätzen erlauben. Da der Standardisierungsprozess jedoch noch nicht vollständig abgeschlossen ist und wenig Referenzimplementierungen vorhanden sind, wird im Folgenden beispielhaft das Zusammenspiel der älteren Modelle beschrieben.

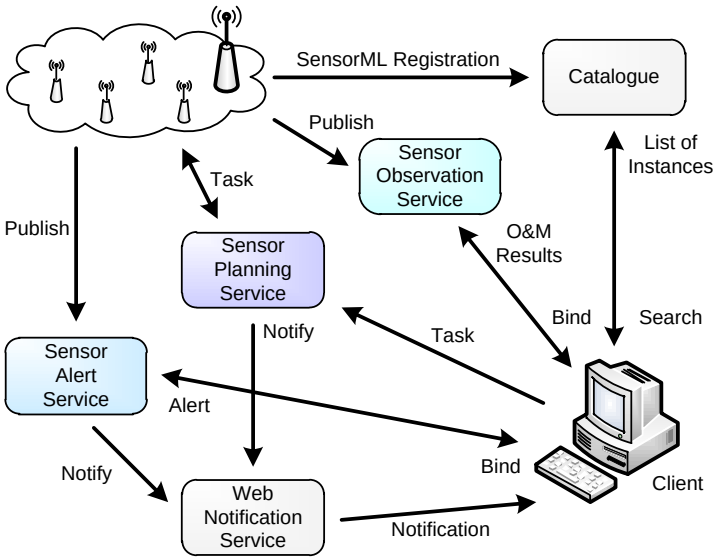


Abbildung 6.3.: Interaktion der SWE-Standards (nach (Ope08b))

Abbildung 6.3 zeigt ein vereinfachtes Zusammenspiel der einzelnen Dienste und Enkodierungen. Initial registrieren sich Sensoren mittels einer SensorML-kodierten Nachricht sowohl bei einem SOS sowie zusätzlich bei einem Verzeichnisdienst (z.B. dem *OpenGIS Catalogue Service* [Ope07a, JNSH10]), bei dem auch der SOS registriert ist. Fortan können Sensoren ihre Messdaten an den SOS und/oder eine dem SOS angeschlossene Datenbank versenden. Ein Benutzer, der diese Messdaten abrufen möchte, sendet zunächst eine entsprechende Suchabfrage an den Verzeichnisdienst und erhält eine Liste mit SOS-Instanzen. Schließlich bindet sich der Benutzer an einen SOS und bekommt die angefragten Daten in Form von O&M-kodierten Nachrichten zugesandt.

An diesem Beispiel wird bereits deutlich, dass SWE keine konkrete Architektur beschreibt und auch keine funktionalen Abstraktionen einführt. Stattdes-

sen werden lediglich Enkodierungen und Schnittstellen standardisiert, ohne dass Aussagen über mögliche Formen der Implementierung getroffen werden. Typische Entwurfsentscheidungen für verteilte Systeme, die z.B. die Sicherheit, Nebenläufigkeit und Skalierbarkeit betreffen, bleiben unberücksichtigt [CKDB06]. Dies führt dazu, dass die Problemstellung, wie eine solche Architektur global verteilt realisiert werden könnte, von SWE nicht direkt adressiert wird. [Ope08b] abstrahiert zudem von diesem Problem, indem es als *Best Practice* die Dienste in einem Rechenzentrum ansiedelt, ohne die globale Skala explizit zu konkretisieren. Es gibt jedoch beispielsweise mit *52° North* [52N12a] und *Constellation SDI* [Con12] zwei Projekte, die auf Basis der SWE-Standards entsprechende Implementierungen entwickelt haben. Allerdings wird auch von diesen das Problem der globalen Verteilung nicht weiter aufgegriffen. Zudem existiert eine konzeptuelle Lücke zwischen den heterogenen Sensoren auf der untersten Ebene und den Diensten (z.B. SOS, SAS, SPS), die Zugriff auf die Sensoren erlauben, da für jeden der Dienste Adapter für das Auffinden und die Kommunikation mit nicht SWE-konformen Sensoren bereitgestellt werden müssen.

Sensor Bus Das *Sensor Bus*-Projekt [BFJP10] von 52° North befasst sich mit oben erwähntem Problem des Auffindens und Zugriffs auf heterogene Sensoren. Allerdings hat dieses Projekt im Wesentlichen das Ziel, Sensoren zum Beispiel einem Verzeichnisdienst wie der SIR bekannt zu machen. Die SIR, als Teil des Next Generation SWE, ist vergleichbar mit dem OpenGIS Catalogue Service, entsprechend kann diese neue Entwicklung Basis zukünftiger, global verteilter Sensor Web-Projekte werden.

Der Sensor Bus repräsentiert eine Schicht, die zwischen den eigentlichen Sensoren (als Datenproduzenten) und Diensten (als Datenkonsumenten) vermittelt. In [BFJ10] werden einfache Interaktionsmuster zwischen diesen beiden Rollen identifiziert, die über den Bus mittels beliebiger Nachrichtenaustauschsysteme (z.B. *Java Message Service* [Sun02], *Twitter* [Twi13]) bzw. Kommunikationsprotokolle (z.B. *Extensible Messaging and Presence Protocol* [XMP12]) interagieren können. Auf diese Weise ist es einheitlich möglich, zwischen den heterogenen Sensoren auf der einen Seite und Diensten auf der anderen Seite zu vermitteln. Im Kontext der SWE-Initiative wird der Sensor Bus verwendet, um Sensoren der SIR und Messdaten dem SAR und SOS direkt zugänglich zu machen sowie eine einheitliche Schnittstelle für das Tasking von Sensoren zu bieten.

Auf SWE aufbauende Projekte In [CKDB06, CB07] wird mit *NICTA Open Sensor Web Architecture* (NOSA) eine Architektur vorgestellt, welche, basierend auf den SWE-Standards, eine Reihe weiterer Dienste zur Verteilung und Verarbeitung der Sensordaten in Grids einführt. Auf diesem Wege soll eine global skalierbare Infrastruktur geschaffen werden, welche die Verarbeitung großer Mengen an Sensordaten von heterogenen Sensornetzen innerhalb ei-

nes Grids erlaubt. Das Projekt wurde im März 2008 abgeschlossen und nicht weitergeführt [Nat12a].

Auch PULSENet [FMF09], ein Framework der Northrop Grumman Corporation, sowie das *Global Earth Observation System of Systems* der NASA [MCF+08] und andere Projekte (eine größere Auswahl an Arbeiten findet sich beispielsweise in [BPRD08]) bauen auf den SWE-Standards auf, um ein globales Sensor Web zu realisieren. Allerdings bieten sie kein umfassendes Konzept zur Verarbeitung und Bereitstellung von Kontextinformationen, sondern verstehen einzelne SWE-Instanzen (z.B. SOS, SAS oder SPS) als autonome Dienste, welche über einen speziellen Verzeichnisdienst, z.B. den OpenGIS Catalogue Service, global auffindbar werden. Dem Paradigma einer dienstorientierten Architektur folgend ist es somit möglich, global verteilte Sensordaten z.B. in Geschäftsprozesse zu integrieren, einfach indem entsprechende Dienstinstanzen über den Verzeichnisdienst zur Laufzeit gefunden und gebunden werden können.

6.1.3. Sensor Web Agent Platform

Ausgehend von der Feststellung, dass existierende Sensordaten-Middleware-Systeme entweder speziell für eine Domäne entwickelt oder für die reine Sammlung und Verbreitung von Sensordaten entworfen wurden, sieht sich die *Sensor Web Agent Platform* (SWAP) als eine Infrastruktur, die insbesondere Endbenutzern den einfachen Zugriff auf gewünschte Sensorinformationen erlaubt [MS06, MTS+06]. Die grundlegende Idee ist, Benutzern auf einfache Art Sensordienste und -anwendungen zur Verfügung zu stellen. Hierfür sieht SWAP einen sogenannten *User Agent* vor, mit Hilfe dessen ein Benutzer nach Anwendungen, Diensten oder Informationen suchen kann und der anschließend eine individuelle Sicht auf die Ressourcen präsentiert.

Damit dieser User Agent der Intention des Nutzers folgend auf die entsprechenden Daten und Verarbeitungsdienste zugreifen kann, müssen alle Daten und Dienste innerhalb der Infrastruktur eine spezifizierte Semantik besitzen. Entsprechend ist in der SWAP-Architektur eine eigene Ontologie-Infrastruktur vorgesehen, welche entsprechende Konzeptualisierungen zur Verfügung stellt. Beispielsweise werden physische Sensoren auf der untersten Ebene (vgl. *Sensor Layer*, Abbildung 6.4) durch Agenten repräsentiert, deren Nachrichteninhalte bei der Kommunikation eine festgelegte Bedeutung erhalten. Auch bietet SWAP Zugriff auf Dienste des Sensor Web Enablement (z.B. Sensor Planning Service, Sensor Alert Service etc.), wobei eigene Ontologien für die SWE-Standards entworfen wurden, um die Kodierungen und Dienste semantisch anzureichern. Auf der nächsthöheren Ebene (*Knowledge Layer*) sorgen sogenannte *Workflow Agents* für die Verarbeitung der Sensordaten entsprechend der Bedürfnisse des Endbenutzers. Basierend auf *OWL-S*, einer semantischen Auszeichnungssprache für Dienste [W3C04a], kann ein Workflow-Agent andere Werkzeug- und Modellierungsagenten zur Lösung eines spezifischen Problems orchestrieren. Nach Ausführung des Workflows wird das Er-

gebnis an einen Anwendungsagenten in der obersten Schicht übermittelt und schließlich dem Benutzer präsentiert.

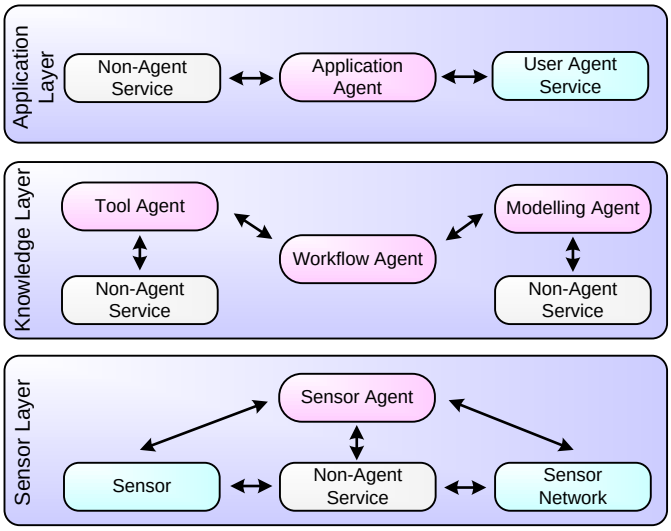


Abbildung 6.4.: SWAP-Architektur (nach (MS06))

Mittels dieser Architektur soll einerseits insbesondere eine Infrastruktur bereitgestellt werden, die es ermöglicht, Informationen heterogener Sensorquellen für komplexe Endbenutzeranwendungen zur Verfügung zu stellen, welche automatisch aufgefunden und verwendet werden können. Andererseits soll die Fusion von Daten verschiedenster Sensor- und Simulationsquellen mit unterschiedlicher zeitlicher und räumlicher Auflösung durch Verwendung von Ontologien ermöglicht werden. Dabei wurden etablierte Standards wie OWL-S oder SWE durch eigene proprietäre Infrastrukturkomponenten auf Basis des *MASII*-Agentensystems ergänzt [MS06].

6.1.4. GeoSwift

Das *Geospatial Sensor Web Information Fusion Testbed* (GeoSWIFT) der Universität York hat sich zum Ziel gesetzt, eine *Geospatial Information Infrastructure* zu etablieren, welche den Skopus existierender Sensornetzwerk-Infrastrukturen erweitert [LCT05]. Ausgehend von der Behauptung, dass andere Sensor Web-Projekte entweder die Beobachtung der Umwelt fokussieren oder militärischen Zwecken dienen, soll die GeoSWIFT-Infrastruktur der Erde eine Art „elektronische Haut“ verleihen, welche mittels großer Mengen an Sensoren die Ereignisse der physischen Welt wahrnehmen und weiterleiten kann. Zur Realisierung dieser Metapher wurde eine Infrastruktur entwickelt, welche eine Reihe funktionaler und nicht-funktionaler Anforderungen erfüllen soll: So muss die Infrastruktur Sensordaten, verknüpft mit räumlichen Informationen, speichern, verbreiten, austauschen, verwalten, veranschaulichen und analysieren können und dabei interoperabel, offen, erweiterbar und skalierbar sein [LCT05]. Um dies zu erreichen, werden in [LTC04, LCT05] die unterschiedlichen Schichten der Infrastruktur beschrieben:

Sensorschicht In dieser untersten Schicht sind alle Arten von Sensoren angesiedelt. Hierzu gehören sowohl in-situ als auch entfernte Sensoren und Sensornetzwerke und gleichfalls Sensoren, welche diskrete Ereignisse veröffentlichen, sowie strombasierte Quellen.

Kommunikationsschicht Diese Schicht übernimmt eine vermittelnde Rolle und kontrolliert die Daten- bzw. Befehlsübertragung zwischen den anderen Schichten. Sie soll von Übertragungsmedien, Protokollen und Netzwerktopologien abstrahieren und erlaubt klientenseitig einen transparenten Zugriff auf die heterogenen Datenquellen der Sensorschicht.

Informationsschicht Innerhalb dieser Schicht werden oben erwähnte funktionale Anforderungen erfüllt. Zu den wesentlichen Informationen, die hier gespeichert und verarbeitet werden, gehören Metadaten der Sensoren sowie aktuelle Messwerte und Zeitreihen.

Das Hauptaugenmerk des GeoSWIFT-Projekts fällt auf die Informationsschicht. Hier finden sich im Wesentlichen drei Arten von Komponenten (vgl. Abbildung 6.5): eine *Sensing Registry*, beliebig viele *Sensing Server* sowie *Viewer*. Die Interaktion dieser Komponenten beruht auf Web Service-Aufrufen und folgt dem allgemeinen Aufbau einer service-orientierten Architektur, in der sich Dienstanbieter (Sensing Service) bei einem globalen Verzeichnis (Sensing Registry) registrieren und somit von Klienten (Viewer) aufgefunden und gebunden werden können. Um Interoperabilität zu gewährleisten, orientieren sich die Schnittstellen der einzelnen Komponenten an den Standards der Sensor Web Enablement-Initiative (vgl. Abschnitt 6.1.2).

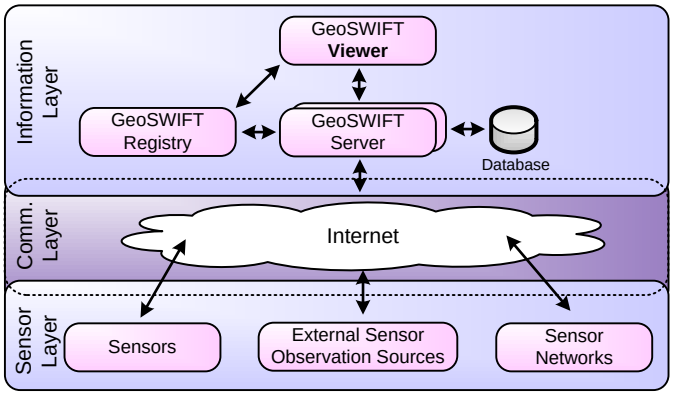


Abbildung 6.5.: GeoSWIFT-Architektur (nach (LTC04, LCT05))

Dieselben Kritikpunkte, die bereits für die SWE Best Practices angeführt wurden, gelten auch für GeoSWIFT. In [LTC04, LCT05] wurde keine Evaluation des Systems vorgestellt und es bleibt fraglich, insbesondere wegen der zentralen Registrierung, ob die nicht-funktionale Anforderung der Skalierbarkeit durch die Architektur erfüllt werden kann. Die Initiatoren des GeoSWIFT-Projekts argumentieren ähnlich und setzen daher bei dem Nachfolger *GeoSWIFT 2.0* auf eine Peer-to-Peer-Infrastruktur, welche räumliche Abfragen

verarbeiten kann [Lia08]. Zu diesem Zweck führen sie neue Architekturkomponenten für das *räumliche Sensor Web* (engl. *Spatial Sensor Web*, SSW) ein:

SSW-Layer Grundlegend wird das SSW in mehrere SSW-Schichten unterteilt, von denen jede Schicht spezifische Dienste zur Wahrnehmung und Verarbeitung genau eines Phänomens (z.B. Temperatur) bietet. Jede Schicht kann aus einer Vielzahl sog. SSW-Schichtknoten desselben Phänomentyps bestehen.

SSW Layer Nodes Innerhalb einer Schicht bieten SSW-Schichtknoten ihre Dienste an. Sie sind untereinander in einer Peer-to-Peer-Topologie verbunden und können mittels der *Peer-to-Peer Spatial Access Method* (P2PSAM) räumliche Abfragen über eine SWE-konforme Schnittstelle bearbeiten.

Spatial Sensor Nodes Die räumlichen Sensorknoten fungieren als eine Art Gateway zwischen den eigentlichen Sensoren und den SSW-Schichtknoten. Auf der einen Seite werden Sensordaten entgegengenommen und schließlich an ein oder mehrere Knoten der SSW-Schicht weitergeleitet.

GeoSWIFT 2.0 Client Nutzer können mit dieser Software auf die GeoSWIFT-Daten zugreifen die Funktionen des SSWs in Anspruch nehmen.

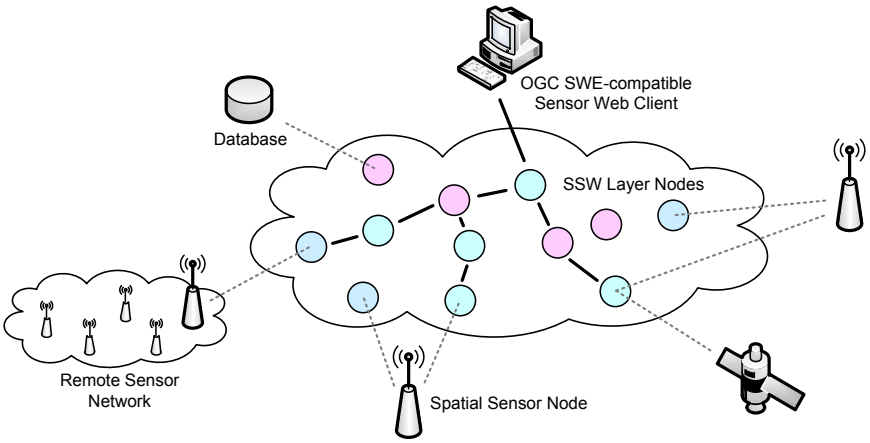


Abbildung 6.6.: GeoSWIFT 2.0-Architektur (nach (Lia08))

Das Zusammenspiel dieser Komponenten bei der Bearbeitung einer Abfrage wird in Abbildung 6.6 veranschaulicht. Zur Visualisierung eines Phänomens im GeoSWIFT Client übermittelt ein Nutzer seine Abfrage an einen beliebigen SSW-Schichtknoten. Dieser leitet die Abfrage innerhalb des Peer-to-Peer-Netzes an sogenannte *Candidate Nodes* weiter, die potentiell Daten über das jeweilige Phänomen innerhalb der angefragten räumlichen Fläche vorliegen haben. Um Kandidaten zur Beantwortung der Abfrage ausfindig zu machen, greift ein SSW-Schichtknoten unter Verwendung von P2PSAM auf eine verteilte Hash-Tabelle zu, in der zu jedem SSW-Schichtknoten dessen räumliche

Abdeckung vermerkt ist. Mittels dieser Methode kann eine Abfrage parallel von mehreren Knoten verarbeitet werden, die als Ergebnis ihre Informationen über das Phänomen an den ursprünglichen SSW-Schichtknoten zur Aggregation zurücksenden.

6.1.5. Hourglass

Hourglass bezeichnet eine Infrastruktur, um Anwendungen einen möglichst einfachen Zugriff auf Daten geographisch verteilter Sensornetze zu ermöglichen. Hierbei werden insbesondere die Aspekte des Auffindens geeigneter Sensoren und Dienste, der Verarbeitung von Sensordaten innerhalb der Infrastruktur sowie der Aufbau einer verlässlichen Kommunikation zwischen Datenproduzenten und -konsumenten adressiert [SPL+04].

Die wesentliche Abstraktion, die durch Hourglass eingeführt wird, ist die eines *Circuits* (dt. Schaltkreis, Kanal). Ein Circuit verbindet einen Konsumenten von Sensordaten über mehrere Zwischenstellen mit einem oder mehreren Produzenten (vgl. Abbildung 6.7). Somit wird eine sichere, verlässliche, verbindungsorientierte Nachrichtenübertragung realisiert, welche zudem die Überwachung von Quality-of-Service (QoS)-Parametern erlaubt. Die Zwischenstellen, auch als *Service Provider* (SP) bezeichnet, übernehmen dabei gleich mehrere Aufgaben: Als Teil eines SPs zeichnet sich der *Circuit Manager* für den Aufbau und die Verwaltung der Circuits verantwortlich. Eine *Registry* erlaubt die Publikation und Suche nach angebotenen Diensten sowie die Abfrage aktueller QoS- und Management-Parameter. Und ein *Service* erlaubt schließlich die Verarbeitung (z.B. Filterung, Aggregation etc.) der durchgeleiteten Sensordaten, wobei Hourglass kein Datenmodell definiert, sondern die Services das untereinander verwendete Format selbst aushandeln müssen.

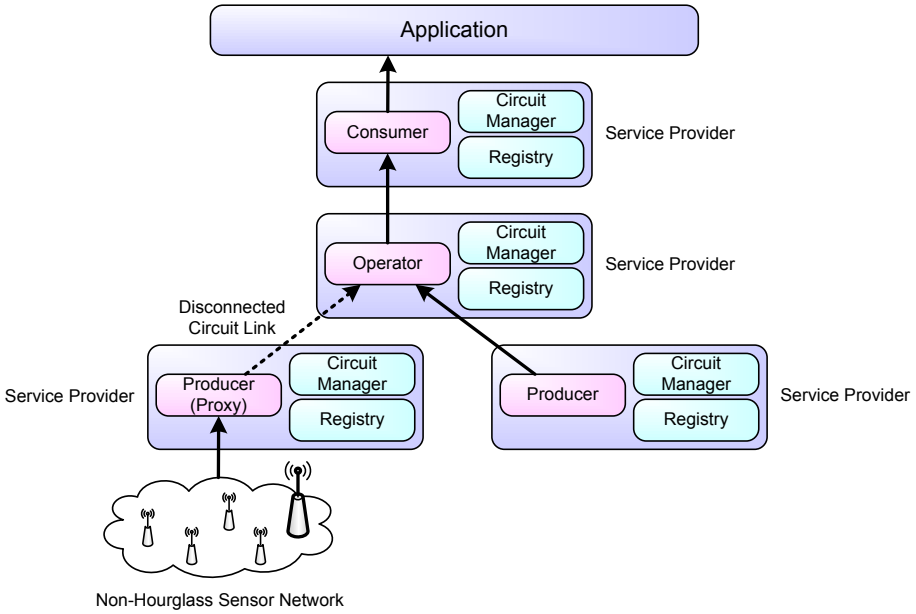


Abbildung 6.7.: Hourglass-Architektur (nach (SPL+04))

Damit eine Anwendung bestimmte Sensordaten übermittelt bekommen kann, muss diese zunächst eine Circuit-Anfrage an einen SP stellen. Eine solche Anfrage enthält einerseits Informationen über den Produzenten bzw. die gewünschten Daten, andererseits eine Art Orchestrierungsbeschreibung der aufzurufenden Dienste sowie ggf. QoS-Anforderungen. Dabei können entweder konkrete Endpunktadressen der Dienstanbieter angegeben werden oder eine abstrakte Beschreibung des Dienstes in Form von Themen (engl. *Topics*), Prädikaten und QoS-Parametern, zu der die Circuit Manager eigenständig versuchen, passende Dienste in der Registry zu finden.

Der Sensordatenstrom beginnt schließlich, sobald der komplette Circuit aufgebaut wurde. Hierfür kommuniziert der SP mit allen anderen SPs entlang des Circuits sowie dem Produzenten und lässt sich deren Bereitschaft zur Teilnahme bestätigen. Erst wenn alle Teilnehmer gebunden wurden, veranlasst der Circuit Manager den Versand der Sensordaten vom Produzenten über die Service Provider hin zum Konsumenten. Fällt ein Teilnehmer des Circuits temporär aus, können sogenannte *Buffer Services* den Datenstrom zwischenspeichern. Auf diese Weise ist es insbesondere auch mobilen Teilnehmern möglich, sowohl als Produzenten, Konsumenten und Service Provider zu agieren, da zeitweilige Verbindungsabbrüche durch die Hourglass-Infrastruktur kompensiert werden können. Durch die Möglichkeit eigene Dienste in Form von Service Providern in die Hourglass-Infrastruktur einzubringen und durch die Registry auffindbar zu machen, können zudem insbesondere auch ressourcenintensive Verarbeitungsaufgaben von (mobilen) Konsumenten in die Infrastruktur verlagert werden.

6.1.6. HiFi

Der *HiFi*-Ansatz beschreibt eine sogenannte *High Fan-in*-Architektur, also einen verteilten Systemaufbau, an dessen Rand sich eine Vielzahl Sensoren befinden und dessen innere Knoten aus herkömmlichen Computern bzw. Servern bestehen, die nach den Prinzipien der kaskadierenden Ströme und sukzessiver Aggregation angeordnet sind [FJK+05]. Die der Architektur zugrunde liegende Erwartung ist, dass die Granularität der Abfragen gröber und der Skopus größer wird, je mehr man sich vom Rand des Systems hin zur Anwendung bewegt. Aus diesem Grund wird in dem hierarchischen Systemaufbau frühestmöglich begonnen, Sensordaten zu aggregieren und weitergehend zu verarbeiten, bevor sie in der Hierarchie weiter nach oben gereicht werden. So sollen Knoten, die in der Hierarchie höher angeordnet sind und somit Sensordaten von mehreren Strömen empfangen, entlastet werden.

Jeder Knoten in dem HiFi-Netzwerk besteht im Wesentlichen aus einem Abfrageplaner und einem Datenstromprozessor. Sendet eine Anwendung eine Abfrage an einen HiFi Server, unterteilt dieser die Abfrage in eine Reihe von Subabfragen und delegiert diese an untergeordnete Knoten, welche zur Beantwortung der Abfragen beitragen können. Die untergeordneten Knoten verfahren genauso, bis die Abfrage am Rand des HiFi-Systems angekommen ist. Der Da-

tenstromprozessor auf jedem Knoten, den eine Abfrage passiert hat, versucht diese fortan mit den kontinuierlich eintreffenden Daten zu beantworten. Erst wenn eine Abfrage vollständig beantwortet werden konnte, wird die Antwort in der Hierarchie weiter nach oben gereicht, wobei der nächsthöhere Knoten seinerseits versucht, die bei ihm registrierte Abfrage in Übereinstimmung zu bringen. Auf diese Weise reduziert sich die Datenmenge auf dem Weg durch das HiFi-System, da lediglich Sensordaten, die zum Beantworten von Abfragen beitragen, berücksichtigt werden.

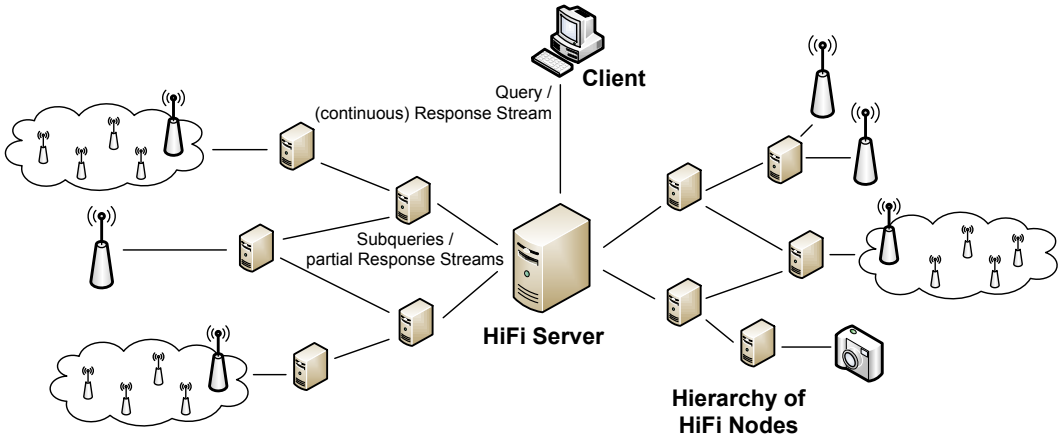


Abbildung 6.8.: HiFi-Architektur (nach (FJK+05))

Abbildung 6.8 zeigt den Aufbau eines HiFi-Netzwerkes (in vereinfachter Topologie). In der Mitte befindet sich ein zentraler HiFi-Server an den Abfragen gestellt werden können. Nach Unterteilung in Sub-Abfragen werden diese an entsprechende Unterknoten delegiert. Dort wiederholt sich der Vorgang. Sind die jeweiligen (Sub-)Abfragen bei allen Knoten registriert, werden sie fortan gegen alle eintreffenden Daten ausgeführt. Wird eine Abfrage erfolgreich ausgewertet, schickt der entsprechende Knoten das Ergebnis eine Ebene höher zur dortigen Auswertung etc., solange, bis die Abfragen wieder explizit deregistriert werden.

6.1.7. IrisNet

IrisNet (*Internet-scale Resource-Intensive Sensor Network Services*) ist eine Softwarearchitektur zur Abfrage global verteilter Sensordaten, welche die spezifischen Anforderungen von datenintensiven Anwendungen adressiert [GKK+03]. Das besondere Augenmerk liegt hierbei weniger auf der Erhebung der Daten aus heterogenen Quellen, sondern vielmehr auf der verteilten Speicherung und Abfrage größerer Mengen an Daten unter Berücksichtigung der Zugriffslokalität.

Zwei wesentliche Abstraktionen werden hierfür eingeführt: *Sensing Agents* (SA) und *Organizing Agents* (OA). Ausgehend von der Annahme, dass Sensoren (z.B. eine Kamera) direkt an einen Computer angeschlossen sind, haben SAs die Aufgabe, ein oder mehrere eingehende Sensordatenströme zu verarbei-

ten und schließlich an einen nahe gelegenen OA, welcher sich für die Speicherung und Abfrageverarbeitung der Daten verantwortlich zeichnet, weiterzuleiten. Die Verarbeitung innerhalb eines SAs geschieht einerseits durch *Privacy Filter*, welche eintreffende Sensordaten nach Maßgabe des Sensorbetreibers vorverarbeiten, um beispielsweise Sensordaten zu anonymisieren (z.B. Gesichter oder Autokennzeichen unkenntlich machen), und andererseits durch sogenannte *Senselets*. Solche Senselets repräsentieren kleine Programme, die auf den eintreffenden Sensordatenströmen ausgeführt werden. Jeder Dienst, der sich für die Sensordaten interessiert, darf zur Laufzeit seine eigenen Senselets in einem SA installieren, sodass anwendungsabhängige Verarbeitungslogik bereits am Ort der Datenerhebung ausgeführt werden kann.

Die aus der Verarbeitung resultierenden höherwertigen Informationen werden schließlich vom SA an einen oder mehrere OAs weitergeleitet. OAs organisieren sich in Gruppen, wobei jede Gruppe eine hierarchisch organisierte, verteilte XML-Datenbank für Sensordaten einer bestimmten Anwendungsdomäne verwaltet. Diese Gruppenstruktur erlaubt die einfache Verteilung der Daten innerhalb eines (globalen) Netzwerkes und sorgt für Robustheit und gute Skalierbarkeit. Die Mechanismen zur (bedarfsabhängigen) Verteilung und Replikation der Daten innerhalb einer OA-Gruppe übernimmt IrisNet. Gleiches gilt für das Routing und die Verarbeitung der *XPath*-basierten [Wor10] Abfragen, da auch hier ein transparenter Zugriff auf die Daten gewährleistet werden soll. Beispielsweise ist es denkbar, dass sich eine Abfrage über diverse OAs einer Gruppe erstreckt. Ein OA, der eine Abfrage entgegennimmt, durchsucht seinen lokalen Datenbestand nach möglichen Ergebnissen und delegiert die Abfrage oder Teile der Abfrage gegebenenfalls an weitere OAs seiner Gruppe. Der Robustheit und losen Kopplung wegen, greift ein OA zum Auffinden geeigneter anderer OAs auf das *Domain Name System* (DNS) [PM87] zurück, in welchem OAs neben ihrer Adresse auch ihren Zuständigkeitsbereich für Sensordaten innerhalb der hierarchischen Datenbank hinterlegen.

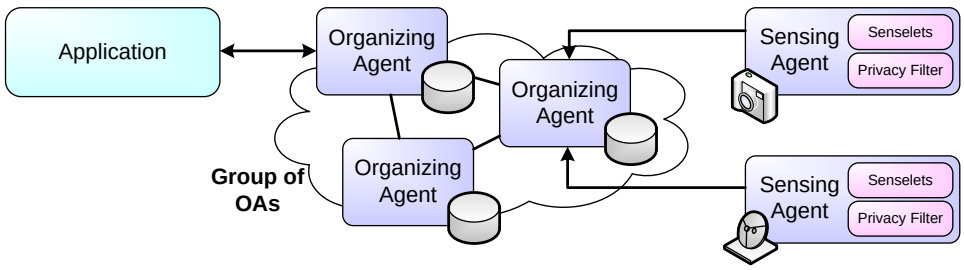


Abbildung 6.9.: IrisNet-Architektur (nach (GKK+03))

In Abbildung 6.9 ist die 2-Tier-Architektur von IrisNet, bestehend aus SAs und OAs, dargestellt. Eine bestimmte Anwendung, im Beispiel eine Anwendung zum Suchen freier Parkplätze, stellt eine Abfrage an einen Parkplatz-Suchdienst. Dieser wird durch eine dedizierte Gruppe von OAs repräsentiert, welche jeweils einen Teil des Gesamtdatenbestandes des Dienstes vorhalten. Die Abfrage wird dann kooperativ von den OAs beantwortet, wobei lediglich

Daten aus der Vergangenheit berücksichtigt werden. Eine direkte Interaktion zwischen Anwendung und SAs sowie eine in die Zukunft gerichtete Abfrage zur proaktiven Benachrichtigung, sobald ein Parkplatz frei wird, sind in IrisNet nicht vorgesehen.

6.1.8. SStreaMWare

Das Ziel von *SStreaMWare* ist die Verwaltung von Sensordaten heterogener Sensoren in verteilen Umgebungen [GRL+08]. Der besondere Fokus liegt hierbei auf der Abfragestellung und der verteilten Abfragebearbeitung. SStreaMWare bezeichnet eine Infrastruktur, welche sich aus hierarchisch angeordneten Software-Diensten zusammensetzt, die zur Laufzeit gesucht und dynamisch gebunden werden können. Auf diese Weise wird eine klassische serviceorientierte Architektur mit lose gekoppelten Diensten realisiert. Die Hierarchie lehnt sich dabei an der geographischen Ausdehnung des Einsatzgebietes an. Auf der untersten Ebene befinden sich die physischen Sensoren, welche von *Proxies* und *Adaptern* der nächsthöheren Ebene angesprochen werden können (siehe Abbildung 6.10). Diese Ebene bildet somit eine Hardware-Abstraktionsschicht für die nächsthöhere Ebene der *Gateways*, welche für die Datenverwaltung eines geographisch relativ eng umrissenen Areals, beispielsweise einzelne Container, Räume oder Gebäude, verantwortlich sind. Eine Menge von Gateways wiederum werden durch eine *Control Site* administriert, entsprechend größer ist auf dieser Hierarchieebene der Bereich des Zugriffs auf Sensoren und Sensordaten. Eine Control Site bildet den Ansatzpunkt für Anwendungsprogramme, d.h. im Grunde genommen repräsentiert sie die Wurzel der SStreaMWare-Hierarchie. Allerdings können Control Sites auch beliebig hierarchisch angeordnet werden, sodass sich mit jeder zusätzlichen Ebene der geographische Einflussbereich vergrößert, d.h. grundsätzlich lässt sich SStreaMWare auch als eine globale Infrastruktur nutzen.

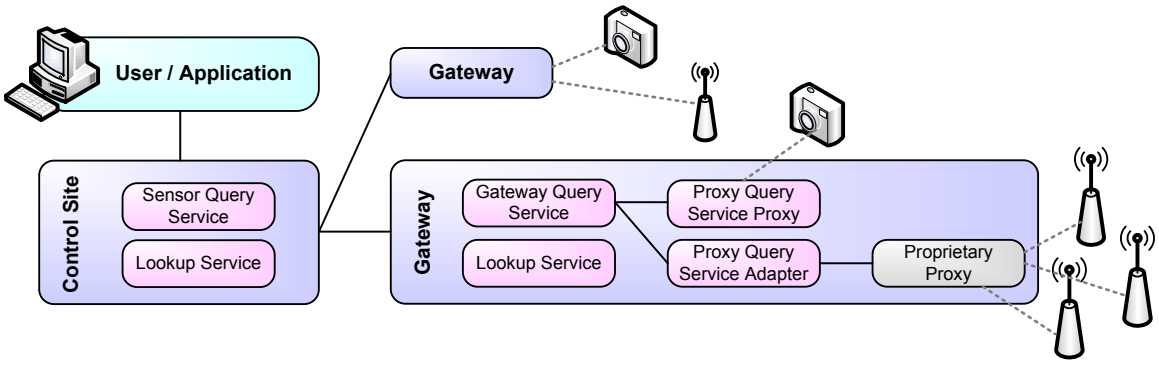


Abbildung 6.10.: SStreaMWare-Architektur (nach (GRL+08))

Die Besonderheit von SStreaMWare liegt in der verteilten Abfragebearbeitung für heterogene Sensordatenströme. Um Daten heterogener Quellen abfragen zu können, basiert die Abfrageverarbeitung auf einem generischen Schema zur Repräsentation von Messdaten und Sensormetadaten. Die Abfragen

selbst beruhen auf einem relationalen Modell und werden deklarativ formuliert. Dieses ermöglicht eine hierarchische Dekomposition einer Abfrage, sodass diese entsprechend in unterschiedlichen Ebenen der Gesamtarchitektur von SStreaMWare verarbeitet wird. Dies führt dazu, dass SStreaMWare gut skalieren kann, da einerseits die Abfrageverarbeitung auf unterschiedlichen Netzwerkknoten ausgeführt wird und andererseits die Ergebnisdaten bereits in den unteren Ebenen der Architektur aggregiert werden können und somit die Auslastung höherer Ebenen reduzieren.

6.2. Integrationsplattformen für RFID-Ereignisse

Die im Folgenden vorgestellten Infrastrukturen für das Web der Dinge nehmen eine Sonderrolle im Kontext des Sensor Webs ein, da sie insbesondere für die Verarbeitung von RFID-Leseereignissen entworfen wurden. Sie spezialisieren sich also auf eine bestimmte Art von Kontextdaten, wobei derartige Leseereignisse neben elektronischen Produktcodes zusätzlich auch weitergehende (Sensor-)Informationen tragen können. Keine der vorgestellten Arbeiten detailliert jedoch die Behandlung dieser zusätzlichen Daten. Die Auswahl der Arbeiten erfolgte bezüglich ihrer Relevanz, ist jedoch nicht vollständig. Eine Übersicht ähnlicher Arbeiten findet sich beispielsweise in [SLZ08, SGCB05]. Eine qualitative Bewertung der Arbeiten erfolgt in Abschnitt 6.3.

6.2.1. EPCglobal / Fosstrak

EPCglobal ist ein Konsortium, welches den Umgang mit sogenannten *Electronic Product Codes* (EPC) standardisiert [EPC10, EPC12, SGCB05]. Ähnlich wie die Sensor Web Enablement-Initiative versucht auch EPCglobal der Heterogenität durch Standardisierung von Informations- und Schnittstellenmodellen gerecht zu werden. So gibt es eine Reihe von Informationsmodellen zur Beschreibung von Produktcodes, zum Nachrichtenaustausch zwischen verschiedenen Institutionen bzw. Rollen sowie zur persistenten Speicherung von Informationen in Datenbanken und Verzeichnissen.

Neben den Informationsmodellen spezifizieren die EPCglobal-Standards eine abstrakte Architektur, die das Zusammenspiel einzelner Rollen und Dienste sowie die Protokolle zum Austausch von Nachrichten unter diesen beschreibt. Abbildung 6.11 zeigt die Interaktionen der wesentlichen Rollen untereinander. Auf unterster Ebene befinden sich die *RFID Reader* (RR), welche die Präsenz von *RFID Tags* wahrnehmen und entsprechende Informationen über das standardisierte *Reader Interface* der *Filtering & Collection*-Rolle (FC) übergeben. Da die RRs inhärent unzuverlässige Rohdaten liefern, obliegt es FC die Daten ein oder mehrerer angeschlossener RRs zu filtern, zu aggregieren und zu transformieren und ein sogenanntes *Application Level Event* (ALE) zu erzeugen. Dieses wird über eine entsprechende Schnittstelle der *EPCIS Capturing Application* schließlich im *EPC Information Services Repository* (EPCIS) persistiert. Klienten, die an Informationen über ein bestimmtes Objekt interessiert

sind, stellen mit dessen EPC eine Anfrage an den *Object Name Service* (ONS), welcher eine Liste mit Adressen von EPCIS Repositories zurückliefert, die potentiell Informationen über das Objekt verwalten könnten. Der Klient kann nun sukzessive die einzelnen EPCIS Repositories über die *EPCIS Accessing Application* ansprechen.

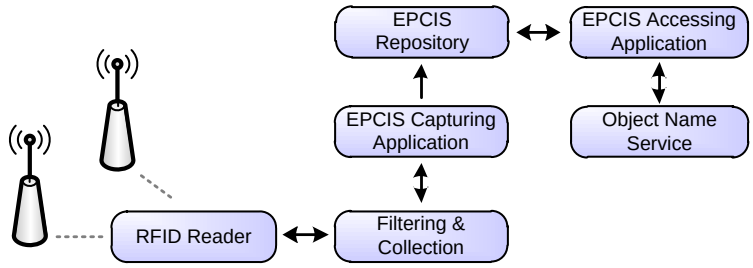


Abbildung 6.11.: Abstrakte Architektur eines EPCglobal-Netzwerkes (nach (EPC10))

Da EPCglobal lediglich die Informations- und Schnittstellenmodelle sowie die funktionalen Rollen beschreibt, obliegt es anderen Projekten konkrete Systemarchitekturen zu entwerfen. Neben kommerziellen Produkten von SAP (siehe Abschnitt 6.2.3), Oracle, Microsoft und anderen[1] existieren auch freie Open Source-Implementierungen wie *Fosstrak*[2] [Fos12, FRL07], das *LLRP Toolkit Project* [LLR12] und *Rifidi* [Rif12].

Konkrete Ideen für verteilte Architekturen geben die Open Source-Implementierungen nicht vor, sodass die Handhabung größerer Datenmengen durch die jeweils gebotenen Komponenten fraglich erscheint. Jedoch existieren einige Ansatzpunkte, die aus der Entwicklergemeinde in die Projekte eingebracht wurden, um stellenweise die Leistung zu verbessern. So finden sich alternative Datenbanken für die EPCIS-Implementierung von Fosstrak [3] [4] oder optimierte Realisierungen für z.B. EPCIS und das ONS [KFZB11] um nur wenige Beispiele zu nennen.

6.2.2. Savant

Die *Savant* Middleware, entworfen vom Auto-ID Center, dient der Vermittlung von RFID-Leseereignissen zwischen Lesegeräten und Enterprise-Anwendungen [CTAO03]. Im Rahmen der Vermittlung werden Ereignisse nicht nur einfach weitergeleitet, sondern können beliebig vorverarbeitet (z.B. gefiltert, aggregiert etc.) werden. Das Ziel ist insbesondere eine Reduktion der Datenmenge, die von den Lesegeräten erzeugt wird, sowie die Aufbereitung der Rohdaten, um den Anwendungen höherwertige Ereignisse zur Verfügung zu stellen.

[1] Weitergehende Informationen finden sich z.B. in [SGCB05]
[2] Vormals unter dem Namen *Accada* entwickelt.
[3] Fosstrak with Oracle and PostgreSQL:
`http://sourceforge.net/mailarchive/message.php?msg_id=23157984`
[4] Using MS SQL Server for EPCIS Repository:
`http://sourceforge.net/mailarchive/message.php?msg_id=21116989`

Die Infrastruktur sieht eine Vielzahl hierarchisch angeordneter Savant-Knoten vor [Aut02]. Auf unterster Ebene sind die *Edge Savants* direkt mit den RFID-Lesegeräten verbunden und empfangen die Leseereignisse. Diese werden verarbeitet und zu einem *Internal Savant* der nächsthöheren Ebene weitergeleitet, bis sie schließlich an die Anwendungen übergeben werden.

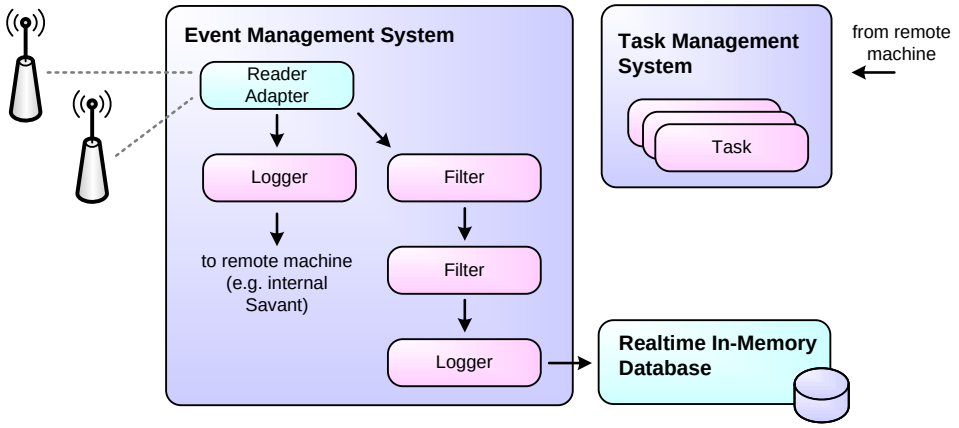


Abbildung 6.12.: Architektur eines Savant-Knoten, Beispielkonfiguration (nach (CTAO03, Aut02))

Die Architektur eines Savants umfasst ein *Event Management System* (EMS), eine *Realtime In-Memory Event Database* (RIED) und ein *Task Management System* (TMS) (vgl. Abbildung 6.12). Das EMS empfängt die Leseereignisse und führt Filter-, Aggregations- und Logging-Funktionen aus, die in beliebiger Weise sequenziert werden können. Um Daten zur Laufzeit vorübergehend zu speichern, kommt mit der RIED eine leichtgewichtige In-Memory-Datenbank zum Einsatz. Das TMS schließlich dient der Ausführung beliebiger Tasks, die das System zur Laufzeit empfängt. Tasks können beliebige Operationen sein, z.B. das Sammeln bzw. Abfragen von Daten oder das Versenden von Nachrichten, sobald bestimmte Ereignisse eintreten.

6.2.3. SAP Auto-ID Infrastructure

Die kommerzielle *Auto-ID Infrastructure* (AII) der Firma SAP AG gehört zu den Marktführern im Bereich der RFID-Systeme [BLHS04, FSB06]. Hauptaufgabe dieser Infrastruktur ist die Integration von RFID- oder sonstigen Datenquellen in Enterprise-Applikationen. Hierfür werden die Daten aus den unteren Sensorschichten durch ein Zusammenspiel einzelner Infrastrukturkomponenten in höherwertige Informationen konvertiert, die schließlich bei der Ausführung von Geschäftsprozessen Verwendung finden können [BLHS04]. Da über das Produkt nur wenig [SLZ08] und darüber hinaus widersprüchliche [FSB06] Dokumentation frei verfügbar ist, beschränkt sich die folgende Betrachtung auf die wesentlichen Komponenten des Systems.

Das System ist als eine 4-Tier-Architektur entworfen (siehe Abbildung 6.13), wobei genau genommen lediglich die beiden mittleren Schichten der Auto-ID

Infrastruktur zuzurechnen sind. Auf der untersten Ebene befinden sich die RFID-Lesegeräte, wobei auch andere Geräte über entsprechende Adapter angesprochen werden können. Die auf dieser Ebene generierten Daten werden an die *Device Operation Layer* (DOL) übertragen. *Device Controller* übernehmen auf dieser Schicht nicht nur die Koordination mehrerer Datenquellen, sondern können die empfangenen Daten bereits vorverarbeiten. Diese Vorverarbeitung geschieht mittels sogenannter *Data Processor*, die jeweils bestimmte Verarbeitungsfunktionen (Filtern, Anreichern, Aggregieren, Zwischenspeichern etc.) übernehmen und beliebig miteinander kombiniert werden können. Die so verarbeiteten Informationen werden schließlich an die *Business Process Bridging Layer* (BPBL) hochgereicht, wo die eintreffenden Nachrichten mit existierenden Geschäftsprozessen assoziiert werden. Diese Schicht übernimmt eine vermittelnde Rolle zwischen den beiden anliegenden Schichten dahingehend, dass Ereignisse der DOL in Bezug auf Geschäftsaspekte interpretiert werden, um dann in geeigneter Form den Anwendungen (z.B. Asset Management- oder Supply Chain Management-Applikationen) der *Enterprise Application Layer* (EAL) zur Verfügung gestellt zu werden. Um dies zu bewerkstelligen, finden sich ein oder mehrere *Auto-ID Nodes* in der BPBL. Eine Rule Engine innerhalb der Knoten versucht eintreffende Nachrichten auf vordefinierte Regelsätze abzubilden, die bei erfolgreicher Übereinstimmung spezifizierte Aktionen ausführen können.

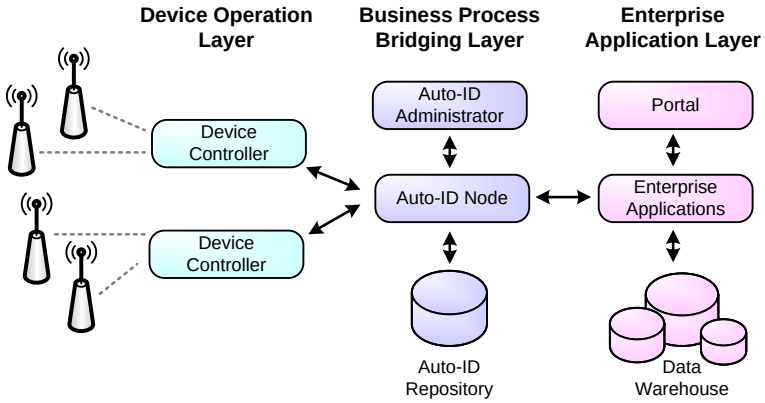


Abbildung 6.13.: Architektur der Auto-ID Infrastructure (nach (BLHS04))

In der Praxis wurde diese Architektur jedoch anders umgesetzt [FSB06]. Die DOL wird durch Drittanbieter realisiert und die *SAP Exchange Infrastructure* entkoppelt nunmehr die BPBL und die EAL. Auch die Auto-ID Nodes finden sich in der Implementierung nicht wieder, sondern wurden durch *Core Services* und *Integration Services* ersetzt. Innerhalb der BPBL bleibt die grundlegende Funktionsweise jedoch bestehen und lediglich die Benennung der einzelnen Verantwortlichkeiten hat sich geändert.

6.3. Bewertungsergebnisse

Nachdem nun die wesentlichen verwandten Arbeiten vorgestellt wurden, sollen diese nachfolgend anhand eines eigenen Kriterienkatalogs bewertet und miteinander verglichen werden. Die Bewertungskriterien hierfür wurden aus der Analyse funktionaler und nicht-funktionaler Anforderungen (siehe Abschnitt 5.3.1 bzw. 5.3.2) abgeleitet.

Die Bewertung erfolgt weitestgehend auf Basis vorhandener Primärliteratur. In Fällen, in denen keine Aussagen zu einzelnen Kriterien in der Primärliteratur zu finden sind, erfolgt die Bewertung entweder durch Aussagen in Sekundärquellen, durch Interpretation oder es wird keine Aussage getroffen, da ein Kriterium keine Erwähnung fand. Von einer qualitativen Bewertung der Kriterien in Form abgestufter Noten, z.B. ob ein Kriterium „gut“ oder „weniger gut“ erfüllt ist, wird abgesehen, da dies in vielen Fällen eine rein subjektive Einschätzung erfordern würde und somit das Gesamtbild verzerren könnte. Im Rahmen dieser Arbeit ist es ausreichend, eine Bewertung hinsichtlich dessen vorzunehmen, ob ein bestimmtes Kriterium erfüllt, teils erfüllt oder gar nicht adressiert wurde. Um die Bewertungen der Kriterien aller Arbeiten nachvollziehen zu können, findet sich in Anhang A eine kommentierte Übersicht aller Bewertungen.

Tabelle 6.14 zeigt eine Zusammenfassung der Bewertungsergebnisse. Fast alle Kriterien wurden von mehr als einer der Arbeiten adressiert. Es existiert jedoch keine Arbeit, die auch nur annähernd alle Kriterien erfüllt. Deutlich zu sehen ist, dass alle Arbeiten auf die Heterogenität zumindest der Produzenten eingehen, wobei eine explizite Unterstützung mobiler Endpunkte nur in den wenigsten Fällen geboten wird. Konsumenten wird von allen Plattformen ein transparenter Zugriff auf die Sensordaten ermöglicht. Der Zugriff erfolgt in den meisten Fällen sowohl mittels historischer als auch ereignisbasierter Abfragen. Ad-hoc Abfragen unterstützen lediglich die Standards des Sensor Web Enablement, wobei auch hier keine Unterstützung unterschiedlicher Strategien gegeben ist. Multimediale Abfragen werden bei keiner der Arbeiten unterstützt.

Um das große Volumen an Sensordaten handhaben zu können, lassen sich alle Plattformen in einem (globalen) Netzwerk verteilen, hierbei finden in den meisten Fällen Peer-to-Peer- und hierarchische Topologien Einsatz. Damit einher geht auch das verteilte Persistieren von Daten und Metadaten, wobei nur ein Teil der Arbeiten auch den netzwerktransparenten Zugriff auf verteilte Daten erlaubt. Der Einsatz unterschiedlicher Datenhaltungssysteme, zum Beispiel relationaler und räumlich-zeitlicher Systeme, wird nur von wenigen Plattformen implizit unterstützt. Gleiches gilt auch für den Umgang mit semantischen Annotationen und Konzeptualisierungen.

Der im Rahmen dieser Arbeit ganz wichtige Aspekt der weitergehenden Verarbeitung von Daten auf dem Weg vom Produzenten hin zum Konsumenten adressieren weniger als die Hälfte der Plattformen, wobei keine Plattform sowohl den Produzenten als auch den Konsumenten die Möglichkeit zur Defini-

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	Global Sensor Network	Sensor Web Enablement	Sensor Web Agent Platform	GeoSwift	Hourglass	HiFi	IrisNet	SStreaMWare	EPCglobal / Fosstrak	Savant	SAP Auto-ID Infrastructure
Funktionale Anforderungen											
1. Datenübermittlung											
1.1 Kommunikationsadapter	++	++	++	++	++	+	++	++	++	++	++
1.2 Logische Adressierung	++	-	-	-	-	-	-	-	-	-	-
1.3 Push- und Pull-basierte Kommunikation	+	++	++	-	-	-	/	-	+	+	++
1.4 Mobilitätsverwaltung	+	-	-	-	++	++	-	-	-	-	-
2. Datenverarbeitung											
2.1 Orchestrierung von Verarbeitungsfunktionen	+	-	++	-	+	-	-	-	-	++	++
2.2 Angebot typischer Verarbeitungsfunktionen	+	-	-	-	+	+	-	-	+	++	++
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	-	++	-	+	-	++	-	-	++	-
2.4 Auffinden von Verarbeitungsdiensten	-	-	++	-	++	-	-	-	-	+	/
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	-	++	-	++	-	++	-	-	++	/
3 Datenverwaltung											
3.1 Unterschiedliche Datenbanksysteme	+	+	+	-	+	+	-	-	-	++	-
3.2 Verteiles Persistieren von Zeitreihen	++	+	+	++	+	++	++	-	-	++	++
3.3 Semantische Abbildungen und Ableitungen	-	++	++	-	-	-	-	-	-	-	-
4 Prozessverwaltung											
4.1 Logging & Accounting	/	/	/	/	/	/	/	/	/	-	/
4.2 Testing & Debugging	/	/	/	/	/	/	/	/	/	-	+
4.3 Monitoring & Reporting	+	-	+	-	+	++	++	-	-	+	/
4.4 Ressourcenverwaltung	+	-	-	-	+	++	-	-	-	-	/
4.5 Zugriffskontrolle	++	-	-	-	-	++	+	-	-	-	/
5 Metadatenverwaltung											
5.1 Unterschiedliche Datenbanksysteme	-	+	+	-	-	-	-	-	-	-	-
5.2 Verteiles Persistieren von Metadaten	++	+	+	++	++	++	-	++	-	-	++
5.3 Semantische Abbildungen und Ableitungen	-	++	++	-	+	-	-	-	-	-	-
5.4 Metadatenermittlung	++	++	-	-	-	-	-	-	-	-	-
6 Datenabfrage											
6.1 Historische Abfragen	++	++	++	++	-	++	++	-	++	++	++
6.2 Ereignisbasierte Abfragen	++	++	++	-	+	++	+	++	++	+	++
6.3 Ad-hoc Abfragen	-	++	-	-	-	-	-	-	-	-	-
6.4 Verschiedene ad-hoc Abfragestrategien	-	-	-	-	-	-	-	-	-	-	-
6.5 Multimediale Abfragen	-	-	-	-	-	-	-	-	-	-	-
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	-	-	-	-	-	-	-	++	-	++
7 Auftragsverwaltung											
7.1 Delegation von Aufträgen	-	++	+	-	-	-	-	-	++	++	++
7.2 Durchführbarkeitsanalyse von Aufträgen	-	++	-	-	-	-	-	-	-	-	-
7.3 Statusüberwachung von Aufträgen	-	++	-	-	-	-	-	-	-	-	-
Nicht-funktionale Anforderungen											
Offenheit	++	++	++	+	-	-	+	+	++	++	-
Heterogenität	++	+	++	+	++	++	+	+	+	++	+
Flexibilität	++	++	++	-	++	++	-	-	+	++	-
Erweiterbarkeit	++	++	++	+	+	+	+	++	++	++	-
Verteilung	++	+	++	++	++	++	+	++	+	++	++
Skalierbarkeit	++	+	++	++	++	++	++	+	+	+	+
Robustheit	++	+	+	+	++	+	++	-	+	+	++
Transparenz	++	+	++	++	++	++	++	++	++	++	++
Sicherheit	++	-	-	-	-	+	++	-	/	-	/
Lose Kopplung	/	++	++	++	+	++	+	++	/	+	/
Starke Kohäsion	++	++	+	/	++	++	/	++	+	++	/
Standards	+	++	++	++	-	-	+	+	++	++	+

Abbildung 6.14.: Übersicht der Bewertungsergebnisse verwandter Arbeiten

tion individueller Verarbeitungsketten gibt. Dafür ist es einem Konsumenten immerhin bei vier Arbeiten erlaubt, eigene Verarbeitungsschritte bereitzustellen, wobei diese in Form eigener Programme bereitgestellt werden und somit auch zustandsbehaftet sein können.

Zu den meisten Plattformen gehören Werkzeuge zur Verwaltung des Systems bzw. Prozesses. Dabei fanden Werkzeuge zum Logging & Accounting sowie zum Testing & Debugging in der Literatur kaum Erwähnung. Ein Teil der Arbeiten bietet jedoch Werkzeuge zum Monitoring & Reporting, meist um den Ausfall einzelner Komponenten zu überwachen, teils aber auch um eine Ressourcenverwaltung, oft in Form einfacher Lastverteiler, zu realisieren. Auch eine Zugriffskontrolle für Daten bieten nur wenige der Arbeiten, wobei diese Kontrolle sich nur in einem Fall auch auf die Dienste erstreckt.

Hinsichtlich der nicht-funktionalen Anforderungen gilt festzustellen, dass die meisten mehr oder weniger von allen Arbeiten erfüllt werden. So setzen alle Arbeiten auf etablierte Kommunikations- und meist auch auf gängige Datenformatstandards. Nur ein Teil der Arbeiten berücksichtigt jedoch die speziellen Standards des Sensor Webs bzw. des Internet der Dinge. Auch hinsichtlich Flexibilität und Erweiterbarkeit der Architektur haben nahezu alle Arbeiten entsprechende Vorstellungen, wobei insbesondere der Begriff der Flexibilität durchaus dehnbar ist. Da alle Arbeiten auf eine verteilte Ausführung ausgelegt sind, behaupten auch alle, gut skalieren zu können. Außerdem verfügen sie über mehr oder minder ausgefeilte Mechanismen, um das Gesamtsystem auch im Fehlerfall noch betriebsbereit zu halten. Eng einher mit der Verteilung geht auch die Forderung nach loser Kopplung, die in den meisten Fällen durch Verzeichnisse zum Auffinden von Diensten bzw. wohldefinierte Schnittstellen zum lokalen Austausch von Daten erreicht wird. Die starke Kohäsion ist in den meisten Fällen durch Modularisierung und damit einhergehend durch das Zuweisen eng umrissener Aufgaben zu einzelnen Modulen bzw. Komponenten gegeben.

Wie bereits erwähnt, spielt die Verarbeitung von Kontextdaten auf dem Weg vom Produzenten hin zum Konsumenten eine zentrale Rolle im Rahmen dieser Arbeit. Zusammenfassend soll daher insbesondere dieser Aspekt noch einmal aufgegriffen werden. Nachfolgend wird daher jede der Arbeiten hinsichtlich dieses Aspektes dem in dieser Arbeit vorgestellten Ansatz der Aufbereitung und Vermittlung kontextbasierter Sichten gegenüber gestellt:

Global Sensor Network Das Global Sensor Network (GSN) fokussiert insbesondere den einheitlichen Zugriff auf Sensoren bzw. Sensornetze, um die Heterogenität auf diesen Ebenen vor dem Nutzer zu verbergen. Unter anderem mit einer flexiblen und erweiterbaren Architektur, logischer Adressierung, verteilte (einfacher) Datenverarbeitung und -speicherung sowie der Unterstützung historischer und ereignisbasierter Abfragen deckt das GSN eine Reihe von Anforderungen ab, die auch an einen Vermittler im Kontext dieser Arbeit gestellt werden. Allerdings geht der Vermittlungsprozess über den Fokus von GSN hinaus und schenkt insbesondere der Verarbeitung von Daten zur Erstel-

lung kontextbasierter Sichten auf dem Weg vom Produzenten hin zum Konsumenten Beachtung. Jedoch ergänzen sich die beiden Ansätze dahingehend, dass GSN vor allem die Aufgabenbereiche Datenerhebung und -übermittlung adressiert, um einen globalen Zugriff auf Daten zu ermöglichen und somit dem Verständnis des Vermittlungsprozesses nach selbst als Produzent agieren und Daten zur weiteren Verarbeitung an den Vermittler übergeben kann.

Hourglass Hourglass ist eine Infrastruktur zur Durchleitung von Sensordaten von Produzenten hin zu Konsumenten, in Form eines robusten Datenstroms (*Circuits*). Dabei unterstützt es Konsumenten bei der Suche nach geeigneten Datenquellen und erlaubt die Ausführung sequentieller Verarbeitungsschritte auf durchgeleiteten Daten. Insofern finden sich auch Ähnlichkeiten zum Konzept der Vermittlung und Aufbereitung kontextbasierter Sichten. Jedoch erlaubt die Verarbeitung lediglich das sukzessive Ausführen bestimmter Funktionen ohne die Möglichkeit, Einfluss auf den Kontrollfluss zu nehmen (z.B. durch konditionale Ausführung oder Parallelisierung). Außerdem gibt es keine explizite Unterstützung für historische und ad-hoc Abfragen. Ereignisbasierte Abfragen müssen in Form proprietärer Filter programmiert werden.

IrisNet IrisNet ist eine globale Infrastruktur zum Speichern und Abfragen großer Mengen an Sensordaten. Das Hauptaugenmerk von IrisNet liegt dabei auf einer möglichst frühen Verarbeitung von Daten nahe dem Produzenten sowie einer der Lokalität der Daten entsprechenden Persistierung, was durch einen hierarchisch organisierten Verbund leistungsstarker Computer realisiert wird. Durch diesen Fokus unterscheidet es sich wesentlich von der Vermittlung kontextbasierter Sichten, da die Präsentation der Daten für den Benutzer nicht adressiert wird.

Sensor Web Enablement Das Ziel der Sensor Web Enablement-Initiative ist es, den Zugriff für Konsumenten auf Meta- und Sensordaten durch Standards zu vereinheitlichen. Diese Standards regeln die Beschreibung und das Auffinden von Sensoren und Daten, deren Zugriff sowie die Kontrolle von Sensoren[5]. Die Vermittlung kontextbasierter Sichten sollte auf diesen Standards aufbauen, hat jedoch insbesondere das Ziel den Zugriff auf Kontextinformationen sowie deren Aufbereitung in Form kontextbasierter Sichten für den Klienten so einfach und aufwandsarm wie möglich zu machen.

GeoSWIFT GeoSWIFT (2.0) ist eine auf Teilen des OGC-Standards basierende Peer-to-Peer-Infrastruktur, welche mittels eines speziellen räumlichen Abfragemechanismus das effiziente Abfragen und Fusionieren von Sensordaten

[5]Eine Bewertung im Rahmen dieses Vergleichs ist schwierig, da die Standards selbst kein konkret realisiertes System darstellen. Daher fand die Bewertung unter anderem auf Basis einer offenen Implementierung der Standards von 52° North sowie unter Berücksichtigung weiterer Standards des Open Geospatial Consortiums, welche die SWE-Standards ergänzen können, statt.

in einem globalen Netz mit vielen Teilnehmern ermöglicht. Es bietet jedoch keinerlei Unterstützung zur Verarbeitung der rohen Sensordaten, sondern präsentiert dem Nutzer, genau wie beim Sensor Web Enablement, ein strukturiertes Ergebnisdokument zur weiteren Verarbeitung.

Sensor Web Agent Platform SWAP beschreibt eine Infrastruktur, die den Nutzern einen einfachen Zugriff auf Sensoren und Sensordaten erlauben soll. Das Ziel hierbei ist, die rohen Sensordaten nach Maßgabe der Benutzer so weitgehend zu verarbeiten, dass diese die Daten einfach weiterverwenden können. Dies deckt sich weitgehend mit der Vermittlung kontextbasierter Sichten. Wesentliche konzeptionelle Unterschiede finden sich bei den Einschränkungen auf bestimmte Technologien sowie in der Art der Erstellung kontextbasierter Sichten.

HiFi HiFi beschreibt ein System kaskadierender Datenströme, welche Produzenten mit Konsumenten verbinden. Das Besondere hierbei ist die hierarchische Topologie sowie damit einhergehend die Abfrageverarbeitung, welche die Abfragelast effektiv im Gesamtsystem verteilen kann. HiFi ist im Wesentlichen ein System, um effizienten und ortstransparenten Zugriff auf Sensordaten zu ermöglichen und unterscheidet sich dahingehend von der Vermittlung kontextbasierter Sichten, dass Daten auf ihrem Weg durch die Hierarchie zwar verarbeitet werden, die Konsumenten jedoch außer auf die Inhalte keinen weiteren Einfluss, z.B. auf die Repräsentation, nehmen können.

SStreaMWare Der Fokus von SStreaMWare liegt auf der Abfrage von Daten verteilter, heterogener Sensoren. Genau wie HiFi folgt es einem hierarchischen Ansatz mit verteilter Abfrageverarbeitung. Auf dem Weg durch die Hierarchie werden die Daten weder verarbeitet noch persistiert, sodass Klienten lediglich Rohdaten als Antwort auf ereignisbasierte Abfragen erhalten. Somit unterscheidet sich das Ziel von SStreaMWare deutlich von einer Vermittlung und Aufbereitung kontextbasierter Sichten.

EPCglobal / Fosstrak Das Standardisierungskonsortium EPCglobal hat Informations- und Schnittstellenmodelle für den Umgang mit elektronischen Produktcodes (EPC) entwickelt, welche u.a. durch die Open Source-Software Fosstrak als Referenzimplementierung umgesetzt wurden. EPCs stellen eine spezielle Form von Kontextdaten dar, welche Dingen eine digitale Identität verleihen und somit eine wesentliche Rolle im Internet der Dinge spielen. Mit Hilfe der Fosstrak Middleware lassen sich diese Codes verarbeiten und aufbereiten, sodass Konsumenten sogenannte Geschäftsereignisse, also abstrakte und auf ihre Bedürfnisse zugeschnittene „Sichten“ erhalten. Durch die Einschränkung speziell auf die Verarbeitung von EPCs und damit einhergehenden spezifischen Vermittlungsfunktionen unterscheidet sich dieser Ansatz jedoch von der allgemeinen Aufbereitung und Vermittlung kontextbasierter Sichten.

Savant Auch die Savant Middleware ist auf die Verarbeitung von RFID-Leseereignissen spezialisiert. In der abstrakten Architektur der EPCglobal-Standards nimmt es die Rolle der Filtering & Collection Middleware ein. Auch Savant ist darauf spezialisiert rohe RFID-Daten mittels Filter, Logger und Tasks zu höherwertigen Geschäftsereignissen aufzubereiten, bietet jedoch, genau wie Fosstrak, nur eingeschränkte Möglichkeiten allgemeine Kontextdaten zu vermitteln.

SAP Auto-ID Infrastructure Haupttaufgabe der kommerziellen SAP Auto-ID Infrastructure (SAP AII) ist ebenfalls die Verarbeitung von RFID-Leseereignissen, wobei zwar allgemeine Sensordaten explizit auch Berücksichtigung, jedoch lediglich ein geringer Teil der EPCglobal-Standards Anwendung finden. Auch hier steht das Aufbereiten der Rohdaten zu geschäftsrelevanten Ereignissen im Vordergrund. Allerdings sind die Verarbeitungsfunktionen sehr eingeschränkt und die Dokumentation dieses kommerziellen Produktes lässt Fragen hinsichtlich der Erweiterbarkeit offen.

6.4. Zusammenfassung

Nachdem in Kapitel 5 ein Konzept der Aufbereitung und Vermittlung kontextbasierter Sichten präsentiert und ein entsprechender Anforderungskatalog aufgestellt wurden, ist dieses Kapitel den konzeptionell verwandten Arbeiten gewidmet. Ausgehend von einer Einordnung verschiedener Anwendungsklassen in den *Sensor Web Layer Stack* wurden insgesamt acht prominente Arbeiten aus der Klasse der *Sensor Web Infrastructures* ausgewählt und vorgestellt. Hinzu kamen drei Arbeiten aus dem Bereich des *Internet / Web of Things*, die sich insbesondere auf die Vermittlung und Verarbeitung von RFID-Leseereignissen als eine spezielle Variante von Kontextdaten konzentrieren.

Im Rahmen eines Vergleichs basierend auf dem Anforderungskatalog aus Abschnitt 5.3 wurden alle elf Arbeiten hinsichtlich ausgewählter funktionaler und nicht-funktionaler Kriterien bewertet. Das Ergebnis dieser Untersuchung war, dass keine der Arbeiten alle Anforderungen adressiert, sondern jeweils nur Teilbereiche des Vermittlungsprozesses aufgegriffen werden. Dabei liegt das Hauptaugenmerk auf der Erfassung, Verteilung und Abfrage von Sensordaten. Insbesondere der Aspekt der Verarbeitung und Aufbereitung von Daten nach Maßgabe der Produzenten sowie das Konzept konsumentenspezifischer Kontextrepräsentationen in Form kontextbasierter Sichten fand kaum bzw. keine Berücksichtigung. Dabei ist festzustellen, dass existierende Arbeiten durchaus auch in den hier vorgestellten Vermittlungsprozess einbezogen werden können bzw. der Vermittlungsprozess dahingehend auf diesen Arbeiten aufsetzen kann, dass ein Teil der existierenden Arbeiten die Aufgabenbereiche der Datenerhebung und -übermittlung übernehmen und diese somit als Produzenten an den Vermittlungsprozess angebunden werden können. Auf diese Weise wäre es beispielsweise möglich, die sichere und robuste Übertragung von Sensordaten über die *Hourglass-Circuits* oder die Fähigkeit zur Abstrak-

tion realer mittels virtueller Sensoren des *Global Sensor Networks* zu nutzen, um darauf aufbauend im Rahmen des Vermittlungsprozesses kontextbasierte Sichten aus den bereitgestellten Daten zu erzeugen.

Für die Realisierung des Konzeptes unter Berücksichtigung der aufgestellten Anforderungen sollen im folgenden Kapitel Architektur und Implementierung einer vermittelnden Infrastruktur präsentiert werden, wobei auch auf die Integration der in diesem Kapitel vorgestellten Standards des Sensor Webs und des Internet der Dinge eingegangen wird.

7. Architektur und Implementierung eines Kontextdatenvermittlers

Bevor Architektur und die wichtigsten Komponenten einer exemplarischen Implementierung einer vermittelnden Infrastruktur für kontextbasierte Sichten aufgezeigt werden, sollen zunächst die Interaktionen und Zusammenhänge zwischen einzelnen Rollen, wie sie in Abschnitt 4.2 definiert wurden, veranschaulicht und hinsichtlich ihres Standes im Gesamtsystem verortet werden. Abbildung 7.1 zeigt eine Draufsicht auf die wichtigsten Teile eines solchen Gesamtsystems. Im Zentrum befindet sich ein Vermittler, welcher hier zunächst als abstrakte Instanz verstanden werden soll (erst im späteren Verlauf wird ein Vermittler dann als ein verteiltes System eingeführt werden). Wesentliche Aufgabe eines Vermittlers ist die Erstellung kontextbasierter Sichten nach Maßgabe der Konsumenten (siehe Kapitel 5). Die Sichten basieren auf Kontextdaten, welche in erster Instanz durch Produzenten (z.B. Sensoren) erhoben werden. Produzenten und Konsumenten bilden somit als Klienten den äußeren Ring, auch als Rand oder Peripherie des Netzes oder Systems (engl. *Network Edge*) bezeichnet.

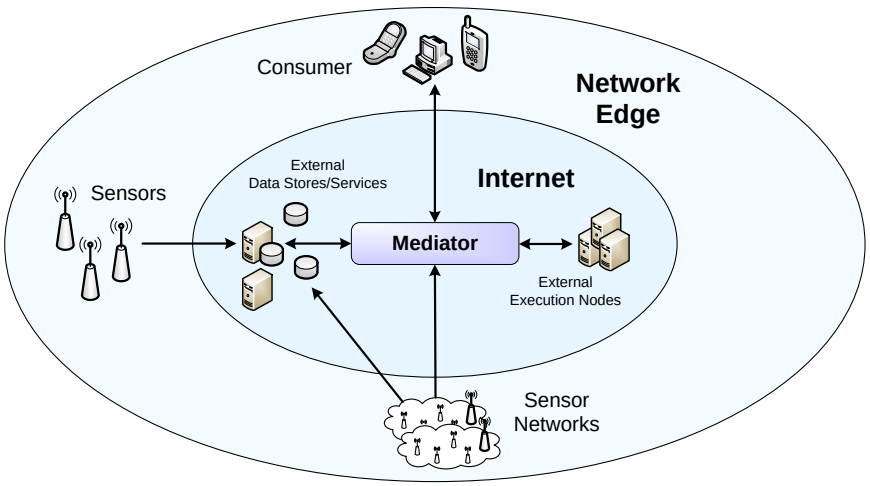


Abbildung 7.1.: Beteiligte im Rahmen des Vermittlungsprozesses (Draufsicht)

Die Produzenten können die Daten entweder direkt an einen Vermittler leiten oder in einer beliebigen Art von Datenhaltungssystem im Internet, auf welches dieser Vermittler zugreifen kann, hinterlegen. Solch ein Datenhaltungssystem kann beispielsweise eine eigene Datenbank, eine Storage Cloud oder auch z.B. ein Dienst des Sensor Webs (z.B. ein SWE-Sensor Observation Service) oder ein Dienst des Internet der Dinge (z.B. ein EPC Information Service) sein.

Der mittlere Ring repräsentiert allgemein das Internet. Er umfasst einerseits das Sensor Web bzw. das Internet der Dinge, steht aber auch für alle anderen externen Ressourcen, die im Rahmen des Vermittlungsprozesses durch einen Vermittler im Internet zugegriffen werden. Hierzu gehören beispielsweise externe Ressourcen, auf die ein Vermittler im Rahmen der Verarbeitung zugreifen muss, z.B. externe Kontextdatenbanken oder Verarbeitungsdienste, die von Dritten angeboten werden.

Abbildung 7.2 nimmt eine andere Perspektive ein und zeigt auf der linken Seite einen Querschnitt der Zusammenhänge (in Anlehnung an [ZSM09]). Auf unterster Ebene finden sich die Sensoren. Diese werden auf der nächsthöheren Ebene zu Sensornetzen zusammengeschlossen. Sowohl einzelne Sensoren als auch Sensornetze können Teil des Sensor Webs bzw. des Internet der Dinge sein. Über dieser Schicht liegt dann ein Vermittler, welcher mit allen anderen Schichten wie oben beschrieben interagieren kann und dahingehend einen Mehrwert bringt, als dass er von der darunterliegenden Heterogenität abstrahiert und höherwertige Verarbeitungsfunktionen hinzufügt.

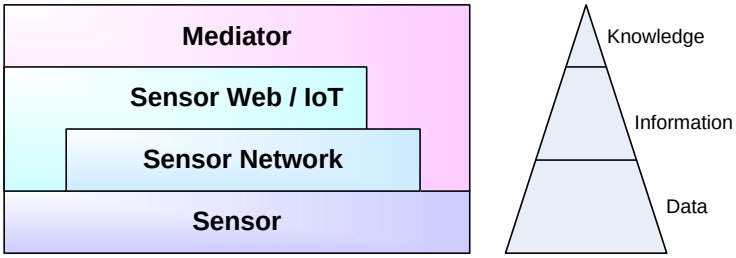


Abbildung 7.2.: Beteiligte im Rahmen des Vermittlungsprozesses (Querschnitt)

Diese höherwertigen Verarbeitungsfunktionen erlauben das Verknüpfen von Informationen als Grundlage für das Ableiten neuen Wissens. Die Abbildung 7.2 kann daher (mit Vorsicht) auch in Bezug auf die Wissenspyramide [RH08] dahingehend interpretiert werden, als dass auf den unteren Schichten Daten erhoben werden. Durch Aufbereitung, Strukturierung und Interpretation repräsentieren diese auf Ebene des Sensor Webs/Internet der Dinge Informationen. Die weitere Aufbereitung, die Aggregation und das Verknüpfen mit zusätzlichen Informationen (d.h. eine Kontextualisierung), die Analyse und das Lernen erfolgt dann auf der obersten Schicht des Vermittlers, wo Informationen zu Wissen abgeleitet werden können [RH08]. Auch wenn der Begriff des Internet des Wissens (engl. *Internet of Knowledge*) in der Literatur gelegentlich verwendet, jedoch nirgends definiert oder erklärt wird, bietet diese Interpretation zumindest eine Basis für weitere Konkretisierungen dieses Begriffs.

Bei den Phrasen „Vermitteln von Wissen“ und „Vermitteln von Eindrücken“ geht es darum, einer Person einen Sachverhalt oder eine Situation verständlich zu beschreiben. „Verständlich“ bedeutet hierbei, dass adäquate Mittel, d.h. eine geeignete Repräsentation der Informationen, welche die Person versteht, gewählt werden müssen (vgl. Abschnitt 4.1). Hier lässt sich eine Analogie zur

„Vermittlung kontextbasierter Sichten“, wie sie in Abschnitt 5.1 definiert ist, ziehen. Im Rahmen des Vermittlungsprozesses sollen einem Konsumenten alle relevanten Elemente eines Kontextes in einer geeigneten Repräsentation zur Verfügung gestellt werden. Da dies je nach Maßgabe des Konsumenten auch mit einer Kontextualisierung, also der Verknüpfung von Informationen, einhergehen kann, lässt sich ein Vermittler auch als eine Middleware-Infrastruktur für das Internet des Wissens auffassen.

Im folgenden Abschnitt 7.1 sollen zunächst wesentliche Entwurfsentscheidungen für die Realisierung einer Middleware dargestellt und konkrete Kernkomponenten mit ihren Schnittstellen, Abhängigkeiten und Interaktionen beschrieben werden. Hier werden auch verschiedene Integrationslösungen für existierende Standards diskutiert und eine Reihe optionaler Erweiterungen vorgestellt. Auf dem Entwurf aufbauend werden schließlich in Abschnitt 7.2 eine exemplarische, prototypische Realisierung der Middleware, die hierbei verwendeten Technologien sowie die im Rahmen der Arbeit entwickelten Werkzeuge präsentiert.

7.1. Entwurf einer Middleware-Architektur

Da nun die Rolle eines Vermittlers im Gesamtsystem nochmals mittels zwei verschiedener Perspektiven eingeordnet wurde, soll im Folgenden unter Berücksichtigung der Anforderungen aus Abschnitt 5.3 ein Grobentwurf der Architektur eines Vermittlers erfolgen. Hierbei ist es zunächst erforderlich, Paradigmen der Softwareentwicklung hinsichtlich ihrer Eignung zu untersuchen (Abschnitt 7.1.1). Im Anschluss daran sollen in Abschnitt 7.1.2 zunächst die Kernkomponenten eines Vermittlers identifiziert und im Hinblick auf das zuvor gewählte Paradigma entworfen werden. Die Beschreibung der Interaktionen dieser Kernkomponenten im Rahmen des Vermittlungsprozesses erfolgt dann in Abschnitt 7.1.3. Verschiedene Ansätze zur Integration der Standards des Sensor Webs und des Internet der Dinge werden in Abschnitt 7.1.4 diskutiert, bevor in Abschnitt 7.1.5 die optionalen Erweiterungen der Middleware vorgestellt werden. Den Abschluss des Entwurfs bildet Abschnitt 7.1.6 mit einer Diskussion um geeignete Datenmodelle zur Verarbeitung und Repräsentation von Sensordaten.

7.1.1. Softwareentwicklungsparadigmen

Im Laufe der letzten Jahrzehnte haben sich eine Reihe von Paradigmen für die Entwicklung komplexer, verteilter Softwaresysteme etabliert. Durch die steigenden Ansprüche an Softwaresysteme einerseits sowie den technologischen Fortschritt andererseits, stoßen aktuelle Paradigmen jedoch an die Grenzen der Beherrschbarkeit. Die zunehmende Komplexität der Software, ein steigender Grad an Verteilung, einhergehend mit einer hohen Dynamik der logischen Umwelt, die Notwendigkeit nebenläufiger oder paralleler Verarbeitung sowie höhere Ansprüche hinsichtlich nicht-funktionaler Aspekte wie Skalierbarkeit,

Robustheit und Sicherheit bedürfen der sorgfältigen Auswahl eines geeigneten Paradigmas zur Entwicklung von Softwaresystemen [BP11]. Eine Auswahl prominenter Paradigmen soll im Folgenden kurz vorgestellt und die einzelnen Paradigmen anschließend hinsichtlich ihrer Eignung zur Entwicklung eines Vermittlers bewertet werden.

Objektorientierte Softwareentwicklung Die objektorientierte Softwareentwicklung ist das älteste und gleichzeitig populärste der hier vorgestellten Paradigmen. In diesem Paradigma stellen Objekte die Entitäten erster Ordnung dar. Sie strukturieren und repräsentieren den Gegenstandsbereich eines Softwaresystems. Der Zustand eines Objektes wird mittels Attributen beschrieben, welche entweder primitive Datentypen (z.B. Wahrheitswerte, Zahlen oder Zeichenketten) oder Referenzen auf andere Objekte darstellen. Auf diese Weise kann ein Objekt durch Aggregation anderer Objekte auch komplexe Entitäten des Gegenstandsbereichs beschreiben [HV07].

Über eine Schnittstelle bietet ein Objekt Methoden sowohl für den lesenden und schreibenden Zugriff auf dessen Zustand als auch für die Ausführung etwaiger Anwendungslogik an. Auf diese Weise kapselt ein Objekt sowohl seinen Zustand als auch sein internes Verhalten und erlaubt externe Zugriffe lediglich über die Methoden der Schnittstelle. Außerdem basiert die Objektorientierung auf dem Prinzip der Vererbung, d.h. Objekte können Generalisierungen oder Spezialisierungen anderer Objekte darstellen und zur Laufzeit polymorph verwendet werden. Das bedeutet, dass ein Objekt, welches eine Methode eines anderen Objektes aufruft, keine Kenntnis von dessen konkreter Ausprägung (z.B. innerhalb des Vererbungsbaumes) haben muss.

Nebenläufige Verarbeitung wird durch Aufspaltung des Kontrollflusses mittels sogenannter *Threads* realisiert, welche jeder für sich einen eigenen, neuen Kontrollfluss erzeugen. Der Zugriff auf gemeinsam genutzte Ressourcen muss manuell synchronisiert werden. Eine verteilte Ausführung kann entweder durch den Austausch proprietärer Nachrichten über Netzwerk-Sockets erfolgen oder verteilungstransparent mittels entfernter Methodenaufrufe (engl. *Remote Procedure Call* bzw. *Remote Method Invocation*).

Komponentenorientierte Softwareentwicklung Komponenten abstrahieren im Vergleich zu Objekten stärker vom Gegenstandsbereich [SGM02]. Eine Komponente stellt eine Sammlung fachlich zusammenhängender Funktionen dar, d.h. sie sind in der Regel stark kohäsiv. Auch sie kapseln ihre Interna über wohldefinierte Schnittstellen, haben aber im Gegensatz zu Objekten keinen von außen observierbaren Zustand, d.h. die Schnittstellen dienen rein fachlichen Zwecken. Komponenten gelten als eine Einheit beim Aufsetzen und in der Verwendung (engl. *Deployment*), sie sind also in sich abgeschlossen. Dennoch können auch sie andere Komponenten zur Erfüllung ihrer Aufgabe voraussetzen, wozu neben der Spezifikation der eigenen angebotenen Schnittstelle auch noch Angaben zu benötigten Schnittstellen anderer Komponenten erforderlich sind.

Im Gegensatz zu Objekten verfügen Komponenten über keine eigene Identität. Jede Kopie einer Komponente ist gleichwertig. Daher ist es in den meisten Fällen sinnlos, zweimal dieselbe Komponente in einem Betriebssystemprozess laufen zu lassen. Jedoch ist es problemlos möglich und im Hinblick auf Leistungsaspekte ggf. auch performanter, dieselbe Komponente mehrfach in verschiedenen Prozessen (ggf. auf unterschiedlichen Geräten) anzubieten, um so eine Verteilung und größere Robustheit zu erreichen. Da Komponenten in sich abgeschlossen sein sollten, bedarf es bei einer Verteilung und dem Zugriff auf Ressourcen auch keiner expliziten Synchronisation einzelner Komponenten untereinander. Das Ausführungsmodell basiert auf dem Prinzip der Steuerungsumkehr, d.h. die Infrastruktur ist für die Verwaltung des Kontrollflusses sowie etwaiger nicht-funktionaler Aspekte verantwortlich [PB13].

Dienstorientierte Architekturen Bei dienstorientierten Architekturen (engl. *Service-oriented Architectures)* stellen Dienste die Entitäten erster Ordnung dar [SH05]. Sie abstrahieren (ähnlich wie Komponenten) von Objekten dahingehend, dass sie an ihrer Schnittstelle meist Methoden einer höheren fachlichen Ebene anbieten. Typischerweise verbirgt die Schnittstelle auch die konkrete Implementierung des Dienstes (welcher z.B. mittels Objekten realisiert sein kann). In Verbindung mit der Tatsache, dass Dienste meist über ein Netzwerk angeboten werden und die Interaktion über den Austausch von Nachrichten (bzw. Dokumenten) verläuft, gelten sie daher als plattformunabhängig.

Idealerweise besitzen sie eine starke Kohäsion, d.h. sie kapseln fachlich zusammenhängende Funktionalitäten, können jedoch zur Erbringung einer Funktionalität wiederum auf andere Dienste zugreifen. Dies kann auch organisationsübergreifend, z.B. im Rahmen von Workflows (Geschäftsprozesse auf technischer Ebene), geschehen. Workflows repräsentieren eine logische Abfolge von Aktivitäten, wobei in der Praxis eine Aktivität häufig durch einen Dienst realisiert wird. Bei Ausführung eines Workflows müssen die Dienstinstanzen nicht im Vorwege feststehen, sondern können dynamisch zur Laufzeit über ein Verzeichnis aufgefunden und gebunden werden. Offene Standards sorgen nicht nur für die Interoperabilität unter Diensten, sondern regeln auch eine Reihe weiterer Aspekte, zum Beispiel das Aushandeln nicht-funktionaler Ausführungskriterien [Mel10].

Agentenorientierte Softwareentwicklung Das Paradigma der agentenorientierten Softwareentwicklung basiert auf Softwareagenten und einer abstrakten Umwelt als wesentlichen Anwendungsbausteinen. Agenten werden eine Reihe von Eigenschaften zugeschrieben, die sie deutlich von den anderen Paradigmen abgrenzen [WJ95]: i) Sie sind autonom, d.h. sie haben selbst die Kontrolle über ihren internen Zustand und ihr Verhalten. ii) Sie können ihre Umwelt wahrnehmen und auf Ereignisse in dieser Umwelt kontextabhängig reagieren. iii) Sie können jedoch auch unabhängig von äußeren Einflüssen selbständig die Initiative ergreifen, d.h. sie sind proaktiv. iv) Außerdem verfügen sie über soziale Fähigkeiten, das bedeutet sie können mit anderen

Agenten kommunizieren, verhandeln und kooperieren, um gemeinschaftlich Ziele zu verfolgen. Und schließlich v) sind sie auch zu komplexeren Verhaltensweisen durch Anwendung mentalistischer Konzepte (z.B. dem *Belief-Desire-Intention*-Modell [RG95]) befähigt, sodass sie beispielsweise selbst geeignete Pläne zur Erfüllung eines Ziels auswählen oder den Nutzen von Alternativen eigens abwägen können.

Die Kommunikation zwischen Agenten erfolgt mittels des Austauschs von Nachrichten. Dabei sind Intention einer Nachricht sowie ihre Rolle in übergeordneten Interaktionsprotokollen (z.B. Auktionen oder Verhandlungen) standardisiert. Ähnlich wie bei Komponenten und Diensten ist damit grundsätzlich eine Plattformunabhängigkeit gewährleistet. Auch stellen Agenten eine in sich geschlossene Einheit dar, welche Zustand und Anwendungslogik nach außen hin durch Schnittstellen kapselt, jedoch besitzt diese im Gegensatz zu den Komponenten eine eigene Identität. Auch können Agenten Dienste anbieten, wodurch wiederum eine lose Kopplung und das dynamische Binden von Funktionen zur Laufzeit ermöglicht wird.

Aktive Komponenten Das Paradigma aktiver Komponenten basiert auf den Annahmen, dass sich die Realwelt mit aktiven und passiven Entitäten nicht adäquat mittels rein objekt- oder komponentenbasierten Ansätze modellieren lässt und andere Ansätze, z.B. Workflows oder Agenten, welche fachliche Dienste auswählen und verwenden können, für die Realisierung komplexer, verteilter Softwaresysteme vorteilhaft sind [PB11]. Es vereinigt entsprechend die wesentlichen Konzepte von Komponenten, Diensten und Agenten [BP11].

Ähnlich der komponentenbasierten Sichtweise deklarieren auch aktive Komponenten angebotene und benötigte Dienste explizit (siehe Abbildung 7.3). Das Verhalten der aktiven Komponenten lehnt sich jedoch an die agentenorientierte Softwareentwicklung an. Die Anwendungslogik hängt von der internen Architektur einer Komponente ab, welche je nach Einsatzkontext unterschiedliche Ausprägungen annehmen kann. So ist es z.B. möglich, die Anwendungslogik mittels verschiedener agentenspezifischer Architekturen, z.B. *BDI*- (Belief, Desire, Intention) oder *Micro*-Agenten zu realisieren, was um die Möglichkeit der Angabe z.B. BPMN-basierter Workflows erweitert wird [PB11].

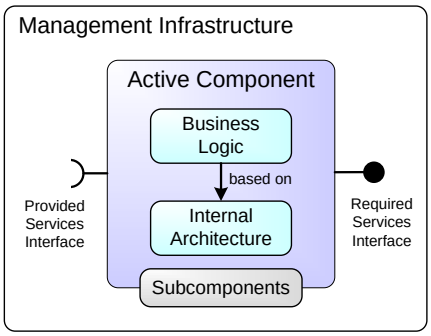


Abbildung 7.3.: Konzeptionelle Darstellung einer aktiven Komponente (nach (BP11))

Ein Interpreter sorgt schließlich für die schrittweise Ausführung der Anwendungslogik, wobei Schritte auch Aktionen eng gekoppelter externer Subkomponenten (z.B. eine grafische Benutzungsoberfläche) oder externe Dienstaufrufe darstellen können. Im Rahmen der Ausführung können auch Dienste anderer aktiver Komponenten aufgerufen werden, welche unter Angabe der Schnittstellenbeschreibung zur Laufzeit vom System automatisch aufgefunden und gebunden werden können. Dienstaufruf und die Rückgabe etwaiger Ergebnisse (ebenso wie interne Methodenaufrufe) verlaufen hierbei asynchron über sogenannte *Futures*, wodurch unter Anderem Deadlocks prinzipiell vermieden werden können, da aktive Komponenten nicht während des Wartens blockiert werden, sondern weitere Aktionen ausführen können [PB11].

Auswahl eines geeigneten Paradigmas In [BP11] wurde eine grobe Klassifizierung verteilter Systeme vorgeschlagen. Hierbei wurden einzelne Klassen hinsichtlich ihrer Ansprüche an die wesentlichen Herausforderungen verteilter Anwendungen wie Nebenläufigkeit, Verteilung und das Erfüllen nichtfunktionaler Kriterien eingeordnet. Auch wurden oben aufgeführte Paradigmen dahingehend bewertet, wie gut sie den jeweiligen Herausforderungen begegnen können (vgl. Abbildung 7.4).

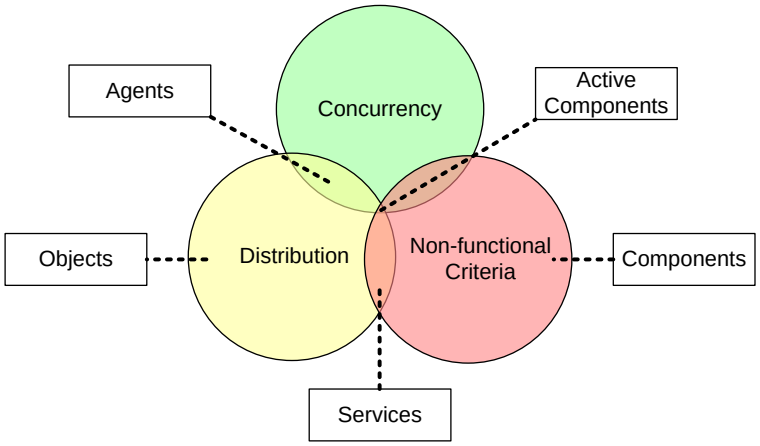


Abbildung 7.4.: Einordnung von Softwareentwicklungsparadigmen gemäß der Herausforderungen verteilter Systeme (BP11)

Das Paradigma der Objektorientierung (mit Ergänzung des entfernten Methodenaufrufs) eignet sich beispielsweise für die Entwicklung verteilter Systeme, bringt jedoch keine adäquaten Konzepte für den Umgang mit Nebenläufigkeit und nicht-funktionalen Aspekten mit sich. Komponenten hingegen bieten durch die Steuerungsumkehr häufig die Möglichkeit, nichtfunktionale Aspekte von außen zu beeinflussen, verfügen jedoch über keine hinreichenden Konzepte zur nebenläufigen Ausführung. Dienstorientierte Architekturen werden für die Entwicklung verteilter Anwendungen verwendet und über mögliche Dienstgütevereinbarungen decken sie ebenfalls nichtfunktionale Aspekte ab. Allerdings adressiert auch dieses Paradigma nicht

die Nebenläufigkeit. Die agentenorientierte Softwareentwicklung bietet hinreichende Konzepte für verteilte, nebenläufige Anwendungen, lässt jedoch nicht-funktionale Aspekte außen vor. Das Paradigma „Aktiver Komponenten“, als einer konzeptionellen Vereinigung all dieser Denkweisen, bietet geeignete Konzepte sowohl für die Verteilung und Nebenläufigkeit als auch den Umgang mit nicht-funktionalen Anforderungen. Genau in diese Schnittmenge fällt die Klasse der Ubiquitous Computing-Anwendungen [BP11], zu denen auch die Vermittlung kontextbasierter Sichten gehört.

Die Aufgabe eines Vermittlers ist es, die unter Umständen sehr großen Datenmengen zwischen einer Vielzahl an Klienten in mobilen, ubiquitären Systemen zu vermitteln. Dies ist nicht durch einen zentralisierten Ansatz ohne die nebenläufige Verarbeitung der Kontextdaten möglich. Zudem kommen eine Reihe nicht-funktionaler Kriterien, z.B. Skalierbarkeit und Robustheit, zum Tragen. Aus diesem Grund bietet sich für den Entwurf und die Realisierung eines Vermittlers das Paradigma aktiver Komponenten an. Im Folgenden werden daher zunächst die wesentlichen Kernkomponenten eines Vermittlers identifiziert und als aktive Komponenten modelliert. Im Anschluss daran wird die Interaktion der Komponenten untereinander vorgestellt und schließlich die Integration der Standards des Internet der Dinge sowie des Sensor Webs präsentiert.

7.1.2. Kernkomponenten der Middleware

Im Folgenden sollen die Kernkomponenten der Middleware vorgestellt werden. Mit Hilfe dieser Komponenten ist ein Vermittler in der Lage, Sensordaten von Produzenten entgegenzunehmen, nach deren Maßgabe zu verarbeiten und ggf. zu persistieren. Konsumenten können Abfragen an diesen Vermittler stellen und die Ergebnisse in Form einer kontextbasierten Sicht aufbereiten und zustellen lassen. Damit sind alle nicht-optionalen funktionalen Anforderungen aus Abschnitt 5.3.1 erfüllt. Optionale Anforderungen, z.B. eine Zugriffskontrolle, semantische Abbildungen und Ableitungen, multimediale Abfragen oder die Delegation von Aufträgen, werden in Abschnitt 7.1.5 als Erweiterungen eines Vermittlers präsentiert.

Dies demonstriert auch die Erfüllung der nicht-funktionalen Anforderungen Flexibilität und Erweiterbarkeit. Die Kernkomponenten sind verpflichtende Bestandteile der Architektur. Sie sind jedoch so gewählt und ausgestaltet, dass sie nur geringe Ansprüche an die Ausführungsumgebung stellen, sodass ein Vermittler auch auf Geräten mit geringer Ressourcenausstattung, z.B. Mobiltelefonen, lauffähig ist. Auch wenn im Folgenden davon ausgegangen wird, dass ein Vermittler sowie die Produzenten und Konsumenten auf mehreren Knoten in einem Netzwerk verteilt ausgeführt werden, so ist es durchaus möglich und je nach Anwendungsfall auch sinnvoll, alle Komponenten auf einem einzigen Gerät laufen zu lassen.

7.1.2.1. Proxies und Gateways

Um von der Heterogenität der Kommunikationskanäle auf Seiten der Produzenten und Konsumenten zu abstrahieren, bedarf es eines Netzübergangs, welcher die wesentlichen Kommunikationstechnologien der Klienten beherrscht und vermittlerseitig in ein geeignetes Protokoll zur weiteren Verarbeitung übersetzt. Diese Rolle übernehmen Proxies und Gateways. Ein Proxy dient hierbei einerseits der Übersetzung zwischen verschiedenen Kommunikationstechnologien (z.B. Bluetooth nach IP) und andererseits zum Zwischenspeichern von Nachrichten, falls ein Adressat temporär nicht erreichbar ist (siehe auch Abschnitt 4.2.4).

Betrachtet man einen Vermittler als verteiltes System, so befinden sich Gateways am Rande des Netzes. Die gesamte Kommunikation zwischen Klient (bzw. Proxy) und Vermittler läuft ausschließlich über die Gateways. Sie nehmen Nachrichten eines Klienten an und leiten sie an die entsprechenden Adressaten weiter (siehe Abbildung 7.5).

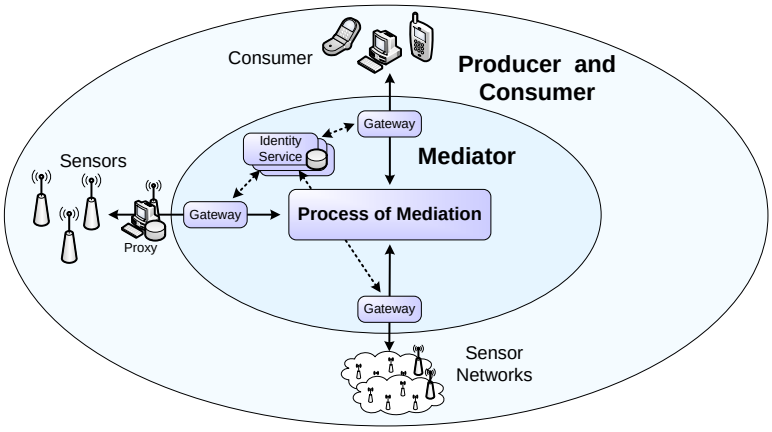


Abbildung 7.5.: Proxy und Gateway in der Gesamtarchitektur

Wie in Abschnitt 5.3.1.1 erläutert, soll eine logische Adressierung unterstützt werden. Das bedeutet, Endpunkte werden mittels eines technologieneutralen Schemas adressiert. Gateways extrahieren die logische Adresse aus einer eintreffenden Nachricht, lassen diese von einem entsprechenden Dienst in ein oder mehrere physische Endpunktadressen übersetzen und leiten die Nachricht entsprechend weiter. Sendet ein Produzent beispielsweise einen neuen Messwert, so muss die Nachricht an die erste Verarbeitungsstufe im Vermittlungsprozess adressiert werden. Die konkrete Endpunktadresse eines entsprechenden Dienstes wird vom Gateway ggf. über einen Identitätsdienst (siehe Abschnitt 7.1.2.2) ermittelt und schließlich weitergeleitet. Auch der umgekehrte Weg des Nachrichtenversands von einem Vermittler hin zu einem Klienten wird über ein Gateway abgewickelt. Das Ergebnis einer ereignisbasierten Abfrage beispielsweise, wird mit der logischen Adresse des Empfängers versehen und an ein Gateway gesendet. Dieses erfragt zunächst beim Iden-

titätsdienst die aktuelle physische Endpunktadresse des (mobilen) Klienten und versucht dann die Nachricht schließlich zuzustellen.

Gateways bieten zudem einen ersten Ansatzpunkt zur Verteilung der Last auf die Knoten der nachgelagerten Verarbeitung. Im Rahmen der Adressauflösung können dem Gateway zu einer logischen Adresse (z.B. eines Diensttyps) mehrere physische Endpunktadressen (sowie ggf. weitere Metadaten) zurückgeliefert werden. Das Gateway kann anhand bestimmter Kriterien, z.B. aktuelle Last oder geographische Nähe, selbst eine geeignete Instanz auswählen und somit bereits am Rand des Netzwerkes den Datenverkehr kanalisieren.

Komposition Aktive Komponenten können Dienste für die Verwendung durch Dritte anbieten. Diese Dienste müssen in Form einer Schnittstellenbeschreibung explizit deklariert werden (siehe Abbildung 7.3). Eine Komponente kann einen Dienst entweder komplett selbständig erbringen oder Teilaufgaben durch die Dienste anderer Komponenten realisieren lassen. Auf diese Weise lassen sich Komposite zusammenstellen, die höherwertige Dienste durch Komposition mehrerer anderer Dienste erbringen. Neben den von einer aktiven Komponente angebotenen Diensten müssen daher auch alle dafür benötigten Dienste explizit deklariert werden.

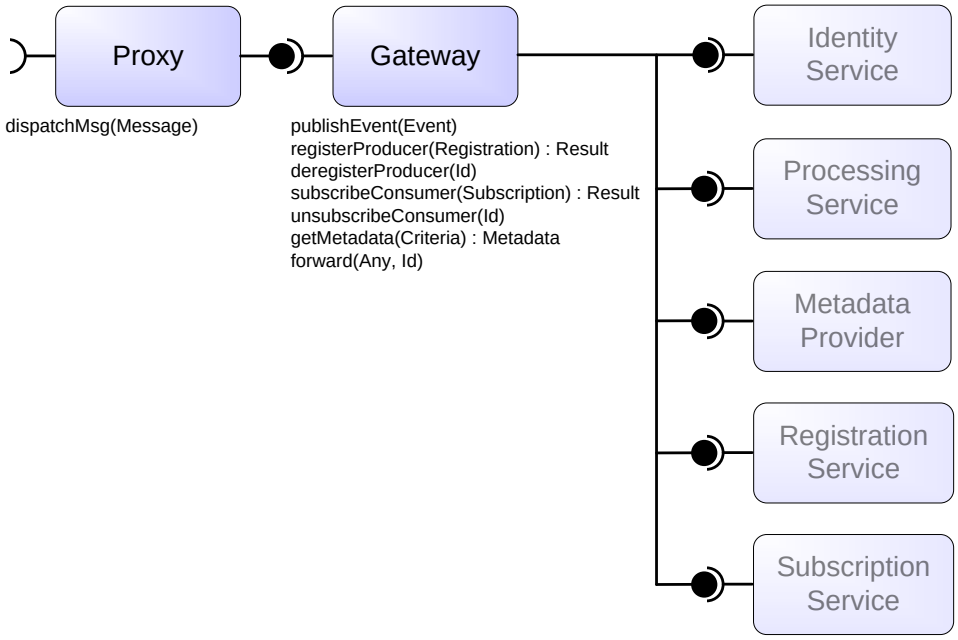


Abbildung 7.6.: Proxy und Gateway: Angebotene und benötigte Dienste

Abbildung 7.6 zeigt unterhalb des Gateways die Methoden der Schnittstelle, die Klienten aufrufen können, um sich bei einem Vermittler zu registrieren, Abfragen zu stellen, Sensordaten zu veröffentlichen oder Metadaten abzufragen. Auf der rechten Seite finden sich die Dienste, die ein Gateway benötigt. Hierzu gehören beispielsweise i) der Identitätsdienst zur Publikation des eige-

nen Endpunkts und zum Ermitteln der Endpunktadressen von Klienten und ggf. anderen Diensten, ii) der Verarbeitungsdienst, an den eintreffende Sensordaten weitergeleitet werden, iii) der Registrierungs- sowie iv) der Subskriptionsdienst, mittels derer sich Produzenten und Konsumenten registrieren bzw. Abfragen subskribieren können, sowie v) die Metadatenbereitstellung, um eigene Metadaten zu veröffentlichen, die Dienstauswahl zu unterstützen und entsprechende Anfragen der Klienten zu beantworten.

7.1.2.2. Identitätsdienst

Wie bereits erwähnt, spielt der Identitätsdienst (engl. *Identity Service*) eine wesentliche Rolle im Rahmen des Vermittlungsprozesses. Dieser Dienst verwaltet die Identitäten aller Beteiligten. Hierzu gehören einerseits die Produzenten und Konsumenten sowie andererseits alle adressierbaren Vermittlungsdienste bzw. Komponenten[1]. Eine Identität besteht aus einem eindeutigen Bezeichner, logischen und physischen Endpunktadressen sowie ggf. weiteren Informationen, z.B. dem Zeitpunkt der letzten Adressaktualisierung.

Die Schnittstelle des Identitätsdienstes, dargestellt in Abbildung 7.7, sieht Methoden zum Hinzufügen, Aktualisieren und Löschen sowie zum Abfragen von Identitäten vor. Beim Abfragen und Löschen ist die eindeutige Kennung einer Identität (kurz *Id*) stets anzugeben. Alternativ kann beim Abfragen auch eine logische Adresse übergeben werden, da diese sich, z.B. im Fall von Diensttypen, auch auf eine Menge von Identitäten beziehen kann. Der Identitätsdienst benötigt die Metadatenbereitstellung, um eigene Metainformationen, z.B. Leistungskennzahlen, zu veröffentlichen und weist daher eine entsprechende Abhängigkeit auf.

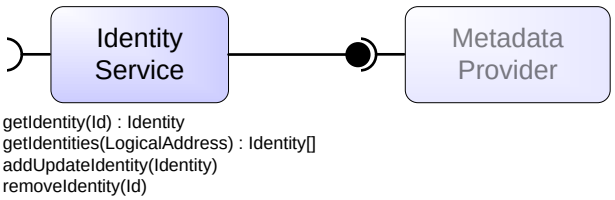


Abbildung 7.7.: Identitätsdienst: Angebotene und benötigte Dienste

Aufgrund der zentralen Bedeutung des Identitätsdienstes muss dieser gut skalieren können. Aus diesem Grund ist der Dienst zunächst zustandslos und kann daher beliebig oft instanziiert werden. Die Identitätsdaten werden in einem Datenhaltungssystem hinterlegt, welches performant und grundlegend verfügbar sein soll. Falls mehrere Instanzen von Identitätsdiensten ausgeführt werden, so müssen diese sich untereinander synchronisieren. Dies kann entweder manuell über den Austausch von Nachrichten geschehen oder durch Verwendung eines verteilten Datenhaltungssystems. In beiden Fällen ist die Toleranz gegenüber dem Ausfall einzelner Knoten gefordert. Aufgrund des *CAP-*

[1]Einige Komponenten werden erst auf Anfrage erzeugt und existieren nur für den Zeitraum der Bearbeitung einer einzelnen Anfrage. Diese werden nicht durch den Identitätsdienst verwaltet.

Theorems [EBF+10], welches besagt, dass in Bezug auf Verfügbarkeit, Konsistenz und Ausfalltoleranz immer nur zwei Eigenschaften gleichzeitig erfüllt sein können, wird die Konsistenzbedingung aufgeweicht. So sollen kurzzeitige Inkonsistenzen der Identitätsdaten der Dienste gestattet sein.

7.1.2.3. Metadatenbereitstellung

Metadaten repräsentieren Informationen über z.B. Sensormessungen, angebotene Verarbeitungsdienste, Produzenten und Konsumenten sowie allgemein Informationen über die Prozessausführung. Gateways beispielsweise können Metadaten nutzen, um aus einer Menge an Diensten gleichen Typs eine geeignete Instanz (z.B. eine wenig ausgelastete Instanz) auszuwählen. Produzenten und Konsumenten benötigen Metadaten, um geeignete Verarbeitungsfunktionen für Sensordaten sowie Abfrageparameter auszuwählen. Auch können sich Konsumenten mittels Metadaten einen Überblick über für sie relevante Sensorinformationen (z.B. in einem bestimmten Areal) verschaffen.

Die Schnittstelle der Metadatenbereitstellung (engl. *Metadata Provider*) sieht grundlegende Methoden zum Abfragen, Speichern, Ändern und Löschen von Metadaten vor (vgl. Abbildung 7.8). Das Abfragen geschieht entweder durch Angabe der Kennung eines Produzenten, Konsumenten oder Verarbeitungsdienstes oder durch Angabe bestimmter Kriterien, z.B. einem Ort oder einem Sensortyp. Es existiert eine Abhängigkeit zum Identitätsdienst um die eigene Identität zu veröffentlichen, damit andere Komponenten den Dienst ggf. auffinden können.

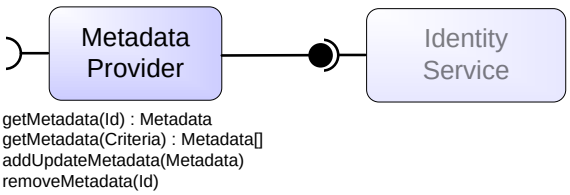


Abbildung 7.8.: Metadatenbereitstellung: Angebotene und benötigte Dienste

Genau wie der Identitätsdienst ist auch die Metadatenbereitstellung zustandslos und somit mehrfach instanziierbar. Ebenso werden alle Metadaten in einer (verteilten) Datenhaltung persistiert, wobei dieselben Anforderungen wie bei dem Identitätsdienst gelten. Der wesentliche Unterschied liegt in der Art der Daten, welche hier semi-strukturiert sind und somit kein einheitliches Datenmodell verwenden.

7.1.2.4. Registrierung

Es ist zwar grundsätzlich nicht notwendig, dass Produzenten sich bei einem Vermittler registrieren, jedoch können sie bei einer Registrierung Informationen hinterlegen, welche beispielsweise für die Verarbeitung der von ihnen veröffentlichten Kontextdaten nützlich sind oder etwaigen Konsumenten das

Auffinden benötigter Informationen erleichtern. Dies können zum Beispiel Verarbeitungsanweisungen an den Vermittler sein, etwaige Metadaten zu den von ihnen veröffentlichten Sensordaten (z.B. Datentypbeschreibungen) oder spezielle Informationen, die während der Verarbeitung benötigt werden, z.B. private Schlüssel zur späteren Ver- oder Entschlüsselung der von ihnen publizierten Daten.

Die Schnittstelle ist, genau wie die Schnittstellen aller Komponenten, die primär der Datenhaltung dienen, einfach gehalten und bietet lediglich die typischen Methoden zum Eintragen, Aktualisieren, Löschen und Abfragen von Registrierungsinformationen (siehe Abbildung 7.9). Die Informationen selbst werden weitestgehend nicht lokal vorgehalten, sondern an spezialisierte Dienste, z.B. die Metadatenbereitstellung, die Workflow-Verwaltung oder den Filterdienst (siehe Abschnitt 7.1.2.7) weitergeleitet. Um diese aufzufinden, kann der Registrierungsdienst Abfragen an den Identitätsdienst stellen. Außerdem werden eigene Identitäts- und Metainformationen über die entsprechenden Dienste veröffentlicht.

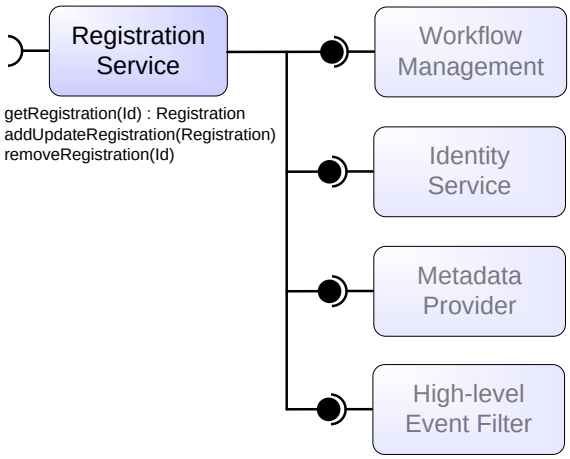


Abbildung 7.9.: Registrierung: Angebotene und benötigte Dienste

Auch die Registrierung ist zustandslos und beliebig häufig instanziierbar. Da auch hier Daten, z.B. die Referenzen auf Workflows und Metadaten, persistent abgelegt werden sollen, gelten die gleichen Anforderungen hinsichtlich einer (verteilten) Datenhaltung wie bei der Metadatenbereitstellung.

7.1.2.5. Low-level Verarbeitungsdienst

Bevor Produzenten einen Datensatz veröffentlichen, d.h. einem Vermittler zur weiteren Verarbeitung übergeben, können sie von diesem auszuführende Verarbeitungsschritte auf Datensätzen bestimmen. Hierzu können sie im Rahmen der Registrierung entsprechende Verarbeitungsanweisungen übermitteln. Diese Anweisungen können prinzipiell beliebig sein, müssen jedoch in Form eines Dienstes seitens des Vermittlers oder eines externen Endpunktes angeboten werden. Typische Verarbeitungsschritte wurden in Abschnitt 2.3 vorgestellt.

Zu den Standarddiensten, die ein Vermittler anbietet, gehören beispielsweise das Filtern, Aggregieren und Komprimieren von Kontextdaten, ebenso wie das Anreichern mit Daten aus externen Quellen (z.B. eines *EPC Information Services* oder eines *Sensor Observation Services*, vgl. Abschnitt 7.1.4.1 bzw. 7.1.4.2).

Die auszuführenden Verarbeitungsschritte werden vom Produzenten (oder seinem Stellvertreter) in Form eines Workflows vorgegeben. Der Workflow deklariert alle auszuführenden Aktivitäten sowie den Kontrollfluss in Form einer abstrakten *Orchestrierungsbeschreibung*. In der Praxis haben sich hierfür die *Business Process Model and Notation* (BPMN) [Obj11a] sowie die *Business Process Execution Language* (BPEL) [Org07] neben anderen Beschreibungssprachen durchgesetzt (siehe Abschnitt 5.4.3). Während BPMN eine Beschreibung auf fachlicher Ebene darstellt, beschreibt BPEL die Ausführung auf einer technischen Ebene [Obj11a, Org07]. Im Rahmen des Vermittlungsprozesses ist eine Beschreibung der Verarbeitungs-Workflows von Produzenten und Konsumenten auf fachlicher Ebene ausreichend, da diese zum einen abstrakter und einfacher durch den Klienten erfolgen kann, und zum anderen die Ausführungsumgebung für die automatische Ergänzung technischer Details, z.B. die Benennung konkreter Dienstendpunkte, verantwortlich ist.

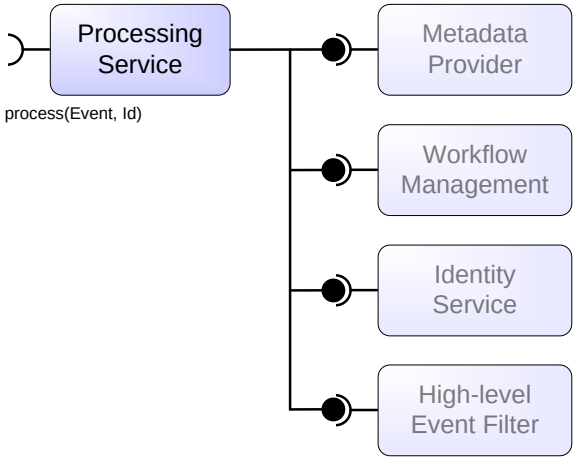


Abbildung 7.10.: Verarbeitungsdienst: Angebotene und benötigte Dienste

Sobald ein Produzent einen neuen Datensatz über das Gateway veröffentlicht, erzeugt das Gateway eine neue aktive Komponente, die mit der Verarbeitung des Datensatzes beauftragt wird. Dieser neue Verarbeitungsdienst (engl. *Processing Service*) fragt bei der Workflow-Verwaltung (siehe Abschnitt 7.1.2.6) die Orchestrierungsbeschreibung ab und ergänzt diese ggf. durch Metadaten, welche über die Metadatenbereitstellung abgerufen werden. Sofern eine Orchestrierungsbeschreibung vorliegt, wird diese Schritt für Schritt ausgeführt, wobei dem Kontrollfluss folgend die einzelnen Dienste aufgerufen werden. Auf diese Weise kann der Datensatz des Produzenten nach dessen Maßgabe verarbeitet werden. Alle zuvor erwähnten Dienste gehören somit zur Schnittstelle des Verarbeitungsdienstes, welche in Abbildung 7.10

dargestellt ist. Wurden die Daten schließlich an den Filterdienst (engl. *High-level Event Filter*) übergeben, endet der Lebenszyklus des Verarbeitungsdienstes und die Komponente wird finalisiert, um von ihr gebundene Ressourcen wieder freizugeben.

7.1.2.6. Workflow-Verwaltung

Das Persistieren und Bereitstellen der Workflows wird von der Workflow-Verwaltung übernommen. Dieser Dienst bietet nach außen Schnittstellen zum Speichern, Ändern, Löschen und Abrufen von Workflows. Außerdem kann er eine Validierung von Workflows beim Speichern vornehmen. Zudem soll es Externen möglich sein, registrierte Workflows (eine entsprechende Erlaubnis des Urhebers bzw. Lizenz vorausgesetzt) anhand bestimmter Kriterien (z.B. genutzte Aktivitäten, Schlüsselwörter etc.) abzurufen, einerseits als Beispiel für die Verwendung bestimmter Verarbeitungsdienste, andererseits aber auch zur eigenen Weiterverwendung, da Workflows oftmals etablierte Verarbeitungsketten repräsentieren und daher nicht von jedem Klienten neu entworfen werden müssen.

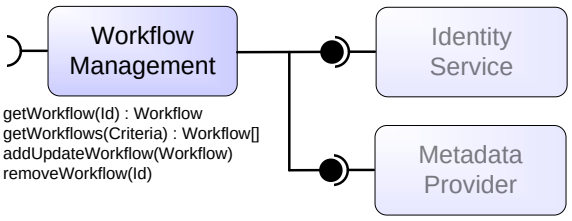


Abbildung 7.11.: Workflow-Verwaltung: Angebotene und benötigte Dienste

Die Schnittstelle mit angebotenen und benötigten Diensten ist in Abbildung 7.11 dargestellt. Die Metadatenbereitstellung sowie der Identitätsdienst werden für die Publikation eigener Meta- bzw. Identitätsinformationen benötigt. Für die Datenhaltung der Workflow-Verwaltung gelten dieselben Anforderungen wie für den Identitätsdienst.

7.1.2.7. Filterdienst

Das Veröffentlichen von Sensordaten bedeutet nicht, dass diese Daten zwangsläufig in einer Datenbank gespeichert und dauerhaft zugänglich sind. Dies ist zwar eine Möglichkeit, jedoch gehört das Persistieren von Sensordaten nicht zu den Kernfunktionen eines Vermittlers, da die Sensordaten in vielen Fällen keine bedeutende Information tragen und ihr Volumen schnell die Kapazität eines Datenhaltungssystems übersteigen kann. Aus diesem Grund ist das Persistieren nur eine Erweiterung eines Vermittlers (siehe Abschnitt 7.1.5.1). Das Veröffentlichen zielt daher primär auf sogenannte *ereignisbasierte Abfragen* (siehe Abschnitt 5.3.1.6). Konsumenten können mittels solcher Abfragen ein Interesse an zukünftigen Ereignissen (Auftreten bestimmter Sens-

ordaten sowie kausaler und temporaler Beziehungen zwischen diesen) anmelden.

Derlei Abfragen werden beim Filterdienst (engl. *High-level Event Filter*) registriert und fortlaufend mit eintreffenden Sensordaten verglichen. Sobald eine Abfrage erfolgreich abgeglichen wurde, wird das Ergebnis zur weiteren Behandlung an einen Verarbeitungsdienst weitergereicht, welcher nach Maßgabe der Konsumenten entsprechende Verarbeitungsvorschriften zur Erstellung kontextbasierter Sichten ausführt und den Konsumenten über das Ergebnis informiert (siehe Abschnitt 7.1.2.9). Die Schnittstelle des Filterdienstes ist in Abbildung 7.12 dargestellt. Meta- und Identitätsinformationen des Dienstes werden über die entsprechend benötigten Dienstschnittstellen veröffentlicht.

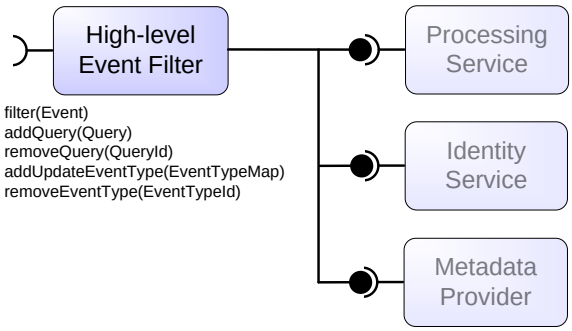


Abbildung 7.12.: Filterdienst: Angebotene und benötigte Dienste

Aufgrund der potentiell großen Menge an Sensordaten, die kontinuierlich als Datenstrom beim Filterdienst eintreffen, sowie der ebenfalls großen Anzahl subskribierter Abfragen muss der Filterdienst besonders performant sein. Herkömmliche Datenbanken können aufgrund ihrer Funktionsweise derlei Anforderungen prinzipiell nicht oder nur ungenügend erfüllen. Stattdessen kommen hier sogenannte *Datenstrommanagementsysteme* (DSMS) zum Einsatz (vgl. Abschnitt 5.3.1.6).

In derlei Systemen werden die subskribierten Abfragen häufig nicht persistiert, sondern ausschließlich im Speicher gehalten. Gleiches gilt für eintreffende Daten, welche, nachdem sie mit allen subskribierten Abfragen abgeglichen wurden, verworfen werden. Das bedeutet, der Filterdienst selbst ist zustandsbehaftet. Aus diesen Gründen ist es jedoch schwierig und nur unter Performanzeinbußen möglich, einen solchen Filterdienst der besseren Skalierbarkeit halber über mehrere Knoten zu verteilen. Der Zustand, bestehend aus allen subskribierten Abfragen sowie dem Zustand der Abfragen[2], könnte zwar zwischen einzelnen Instanzen des Filterdienstes synchronisiert werden, verursacht jedoch einen sehr großen Verwaltungsaufwand, zumal auch temporale und kausale Beziehungen, die in Abfragen formuliert werden können, Berücksichtigung finden müssen.

[2]Abfragen besitzen auch einen Zustand, der sich u.a. aus der Bindung von Variablen mit konkreten, bisher erfüllten Ereignisbedingungen sowie dem Zustand von Zeitfenstern der Abfrage ergibt.

Es existieren jedoch auch verteilte Lösungen für DSMS (z.B. *Integrasoft CEP Cloud Services* [Int10a], *Next CEP System* [SMMP09], *DiCEPE* [PHR+12], *DistCED* [PSB03]), die aber teils nicht frei verfügbar sind und deren Evaluation den Rahmen dieser Arbeit übersteigen würde. Daher kommt in dieser Arbeit ein zentralisierter Filterdienst zum Einsatz, dessen Schnittstelle generisch ist, der jedoch keine weiteren Anforderungen bzgl. der Datenhaltung stellt.

7.1.2.8. Subskriptionsdienst

Um eine Abfrage beim Filterdienst zu hinterlegen, muss sich ein Konsument bei einem Vermittler anmelden (engl. *subscribe*). Hierfür sendet er über ein Gateway eine entsprechende Subskriptionsnachricht, welche im Wesentlichen aus zwei Teilen besteht: einer Abfrage und einem (optionalen) Workflow. Die Abfrage spezifiziert den Kontext, an dem der Konsument interessiert ist. Eine solche Abfrage kann beispielsweise ein Ereignis beschreiben, auf welches anschließend im Filterdienst gewartet wird, oder Filterbedingungen für persistierte Kontextdaten, die in einem Datenhaltungssystem hinterlegt sind (nicht Bestandteil der Kernkomponenten). Der Workflow dient, genau wie bei der Registrierung von Produzenten, der Verarbeitung von Daten. In diesem Fall handelt es sich jedoch um das Ergebnis oben erwähnter Abfrage, dessen weitere Verarbeitung durch eine Orchestrierungsbeschreibung spezifiziert wird. Das Ziel hierbei ist die Erstellung einer kontextbasierten Sicht, welche alle relevanten Informationen in einer geeigneten Repräsentation für den Konsumenten darstellt.

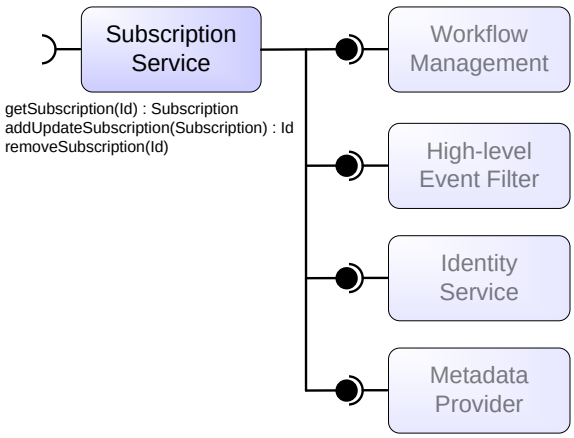


Abbildung 7.13.: Subskriptionsdienst: Angebotene und benötigte Dienste

Die entsprechende Schnittstelle des Subskriptionsdienstes ist in Abbildung 7.13 dargestellt. Der Dienst der Workflow-Verwaltung wird für das Hinzufügen, Aktualisieren und Löschen von Workflows benötigt, der Filterdienst ist notwendig, um die Abfragen subskribieren zu können, und mit Hilfe des Identitätsdienstes werden die benötigten Dienstinstanzen ggf. aufgefunden und die eigene Dienstidentität veröffentlicht. Die Metadatenbereitstellung

kann ggf. die Auswahl von Dienstinstanzen durch Metadaten unterstützen und empfängt ebenso die Leistungskennzahlen vom Subskriptionsdienst.

Ähnlich wie der Registrierungsdienst leitet auch der Subskriptionsdienst einen Großteil der Subskriptionsdaten an andere, spezialisierte Dienste weiter. In einem eigenen (verteilten) Datenhaltungssystem werden lediglich die Referenzen auf die ausgelagerten Daten persistiert. Der Subskriptionsdienst ist zustandslos und kann daher beliebig oft (verteilt) instanziiert werden. Die Synchronisation erfolgt, ebenso wie bei den anderen Diensten, entweder manuell über den Austausch von Nachrichten unter allen Instanzen oder automatisch mittels der verteilten Datenhaltung.

7.1.2.9. High-level Verarbeitungsdienst

Der Verarbeitungsdienst auf Konsumentenseite (engl. *High-level Processing Service*) unterscheidet sich nicht wesentlich von dem Verarbeitungsdienst auf Produzentenseite. Beiden Diensten ist gemein, dass sie Kontextdaten entgegennehmen, einen Workflow zur klientenbestimmten Verarbeitung der Daten anfordern und ausführen und das Ergebnis weiterleiten. Entsprechend ähneln sich auch die Schnittstellen der beiden Dienste.

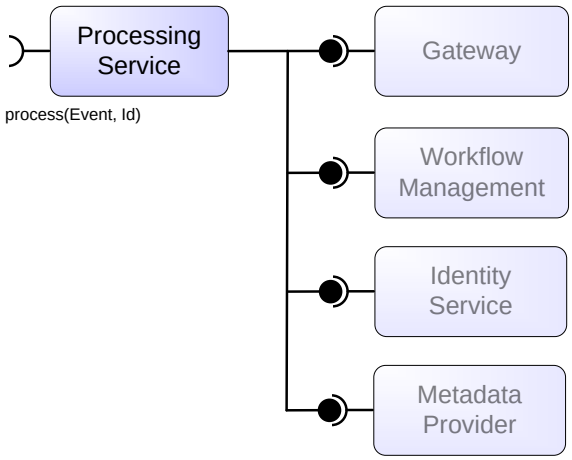


Abbildung 7.14.: Verarbeitungsdienst: Angebotene und benötigte Dienste

Abbildung 7.14 stellt die Schnittstelle mit benötigten Diensten für den konsumentenseitigen Verarbeitungsdienst dar. Der wesentliche Unterschied zur Schnittstelle des produzentenseitigen Verarbeitungsdienstes (vgl. Abbildung 7.10) liegt darin, dass der konsumentenseitige Dienst statt des Filterdienstes einen Gateway-Dienst benötigt, da das Ergebnis der Workflow-Ausführung über das Gateway an den Konsumenten überstellt werden soll.

Damit sind nunmehr alle Kernkomponenten eines Vermittlers beschrieben. Unter Verwendung dieser Komponenten kann bereits ein einfacher Vermittlungsprozess ausgeführt werden, d.h. Produzenten können ihre Sensordaten über einen Vermittler veröffentlichen und von diesem nach eigenem Wunsch verarbeiten lassen und Konsumenten können ereignisbasierte Abfragen sub-

skribieren, deren Ergebnisse nach ihrer Maßgabe zu kontextbasierten Sichten aufbereitet und zugestellt werden können.

7.1.3. Interaktion der Komponenten

Das Zusammenspiel der einzelnen Komponenten wurde durch die Schnittstellen bereits angedeutet und soll im folgenden Abschnitt anhand eines Beispiels zunächst aus Sicht des Produzenten und im Anschluss aus Sicht des Konsumenten veranschaulicht werden.

7.1.3.1. Beispiel für die Registrierung und Publikation von Kontextdaten

Eine Wetterstation verfügt über Sensoren zum Messen der Temperatur und der Luftfeuchtigkeit sowie über eine kleine Kamera. Diese Daten werden von der Station mittels der ZigBee-Funktechnologie in einfachen Name/Wert-Paaren an einen nahegelegenen Computer (Stellvertreter bzw. Proxy) übertragen. Um die Daten auch anderen Nutzern (ggf. auch global) zur Verfügung stellen zu können, sollen sie über einen Vermittler publiziert werden. Allerdings soll das Kamerabild zuvor seitens des Vermittlers skaliert und zusätzlich per *File Transfer Protocol* (FTP) auf einen Webserver hochgeladen werden.

In diesem Beispiel ist es zunächst erforderlich, dass die Wetterstation beim Vermittler registriert wird, damit der Verarbeitungs-Workflow und einige Metadaten hinterlegt werden können. Der Inhalt einer solchen Registrierungsnachricht ist in Listing 7.1 dargestellt und besteht grundlegend aus einigen optionalen Metainformationen über den Sensor und die Sensordaten sowie einen Verarbeitungs-Workflow.

```
1  Registration {
2      Metadata {
3          Location=Hamburg,
4          SensorType=WeatherStation
5      }
6      Workflow=<bpmn.../>[...]</bpmn>,
7      EventTypeMap {
8          MapName=WeatherStationDataTypes,
9          Types {
10             Temperature=float,
11             Humidity=float,
12             Photo=binary
13         }
14     }
15 }
```

Listing 7.1: Beispiel einer Registrierungsnachricht

Da der Workflow mittels BPMN modelliert wird, existieren eine Reihe von Werkzeugen, um den Anwender bei der Umsetzung zu unterstützen. Seit

BPMN 2.0 [Obj11a] ist die Abbildung des graphischen Modells in eine XML-basierte Repräsentation standardisiert, sodass die XML-Beschreibung einem Vermittler einfach als Teil der Registrierungsnachricht übermittelt werden kann. In Abbildung 7.15 ist der Workflow für die Daten der Wetterstation dargestellt. Er besteht lediglich aus zwei Aktivitäten: das Skalieren des Bildes sowie das Hochladen des skalierten Bildes auf einen FTP-Server. Die beiden Aktivitäten werden durch Dienste realisiert, welche der Vermittler anbietet. Die Bezeichner der Dienste sowie deren Eingabeparameter müssen dem Anwender bekannt sein[3].

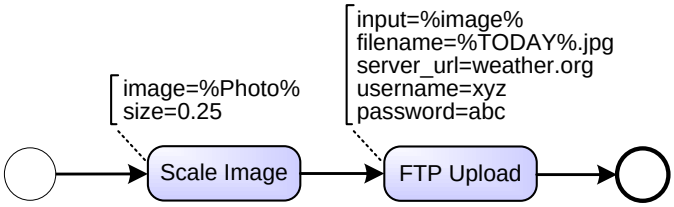


Abbildung 7.15.: Beispiel-Workflow zur vermittlerseitigen Verarbeitung

Das Binden von Werten an die Eingabeparameter geschieht in diesem Beispiel mittels BPMN-Kommentaren, welche über den Aktivitäten zu sehen sind. Neben Zuweisungen primitiver Datentypen (z.B. $size = 0.25$) ist es auch möglich, Referenzen, Variablen oder Ausdrücke zu verwenden, welche vom Vermittler zur Ausführungszeit gebunden bzw. ausgewertet werden. $\%Photo\%$ beispielsweise referenziert den Wert der entsprechenden Variablen aus der Sensordatennachricht (vgl. Listing 7.2), welcher mittels $image = \%Photo\%$ an den Eingabeparameter des Skalierungsdienstes gebunden wird. Nach demselben Prinzip weist $input = \%image\%$ dem Eingabeparameter des FTP-Dienstes das skalierte Bild der vorherigen Aktivität zu. $\%TODAY\%$ im Gegensatz stellt ein Schlüsselwort dar, welches bei der Ausführung durch einen entsprechenden Ausdruck, in diesem Fall das aktuelle Datum, substituiert wird.

In dem Listing 7.1 ist weiterhin ein spezieller Bezeichner $EventTypeMap$ aufgeführt. Dieser beschreibt für alle Felder der nachfolgenden Sensordatennachrichten deren Datentyp (z.B. int, $boolean$ oder $binary$). Da die Abfragen der Konsumenten datentypspezifische Ausdrücke enthalten können (z.B. $Wert_X < 0.1$), müssen dem Filterdienst vorab oder zusammen mit den eigentlichen Daten entsprechende Typinformationen als sogenannte Ereignistypabbildung (engl. *Event Type Map*) übergeben werden (vgl. Listing 7.1). Hierfür existieren zwei Möglichkeiten:

1. Ein Datensatz enthält neben den eigentlichen Datenwerten auch gleich Informationen über deren Typ. In diesem Fall spricht man von einem abgeschlossenen (engl. *self-contained*) Datensatz, dem keine weiteren Informationen hinzugefügt werden müssen. Dies ist beispielsweise bei Da-

[3] Ein Verzeichnis zum (automatischen) Auffinden der Dienstspezifikationen ist eine Erweiterung eines Vermittlers und gehört nicht zu den Kernkomponenten (siehe Abschnitt 5.4).

tensätzen, die dem *SensorML*-Standard der Sensor Web Enablement-Initiative (siehe Abschnitt 6.1.2) folgen [Ope07b], der Fall.

2. Falls ein Datensatz, wie in diesem Beispiel, nicht abgeschlossen ist, muss dem Filterdienst im Rahmen der Registrierung eine Ereignistypabbildung übergeben werden, sodass dieser die Werte entsprechend ihrer Datentypen interpretieren kann. Eine solche Abbildungsfunktion listet für jeden Wert seinen Datentyp auf, z.B. $Temperature \rightarrow float, Photo \rightarrow binary$, und kann über die Metadatenbereitstellung abgefragt werden.

Die Registrierung übernimmt ein Stellvertreter, z.B. der Computer, der auch die Daten der Wetterstation per ZigBee empfängt. Die Registrierungsnachricht wird über ein beliebiges (öffentliches) Gateway an den Vermittler gesandt (siehe Abbildung 7.16). Dieses leitet die Nachricht an den Registrierungsdienst weiter, welcher eine Reihe einzelner Schritte ausführt:

- Es wird, sofern noch nicht vorhanden, eine eindeutige Kennung für den Produzenten erzeugt.
- Der Workflow wird an die Workflow-Verwaltung übergeben.
- Metadaten werden an die Metadatenbereitstellung übermittelt.
- Referenzen auf Workflow und Metadaten, die Kennung sowie weitere Daten der Registrierungsnachricht werden persistiert.
- Die Beschreibung von Datentypen (engl. *Event Type Map*) wird beim Filterdienst für ereignisbasierte Abfragen (engl. *High-level Event Filter*) registriert.
- Das Ergebnis der Registrierung wird zusammen mit der Kennung an das Gateway zurückgegeben.

Die Kennung sowie die aktuelle Endpunktadresse des Senders bzw. Produzenten (in diesem Fall des Stellvertreters) meldet das Gateway beim Identitätsdienst an und übermittelt dem Sender schließlich dessen Kennung als Antwort auf die Registrierungsanfrage.

Nach erfolgreicher Registrierung können nun Daten der Wetterstation an den Vermittler übermittelt werden. Hierfür müssen entsprechende Nachrichten (dargestellt in Listing 7.2) an ein Gateway geschickt werden. Jede Nachricht enthält neben den eigentlichen Sensordaten auch die Kennung des Sensors, damit eine Zuordnung zu dem hinterlegten Workflow möglich ist, sowie ggf. den Namen der Event Type Map um eine nachfolgende Abbildung der Datentypen zu ermöglichen.

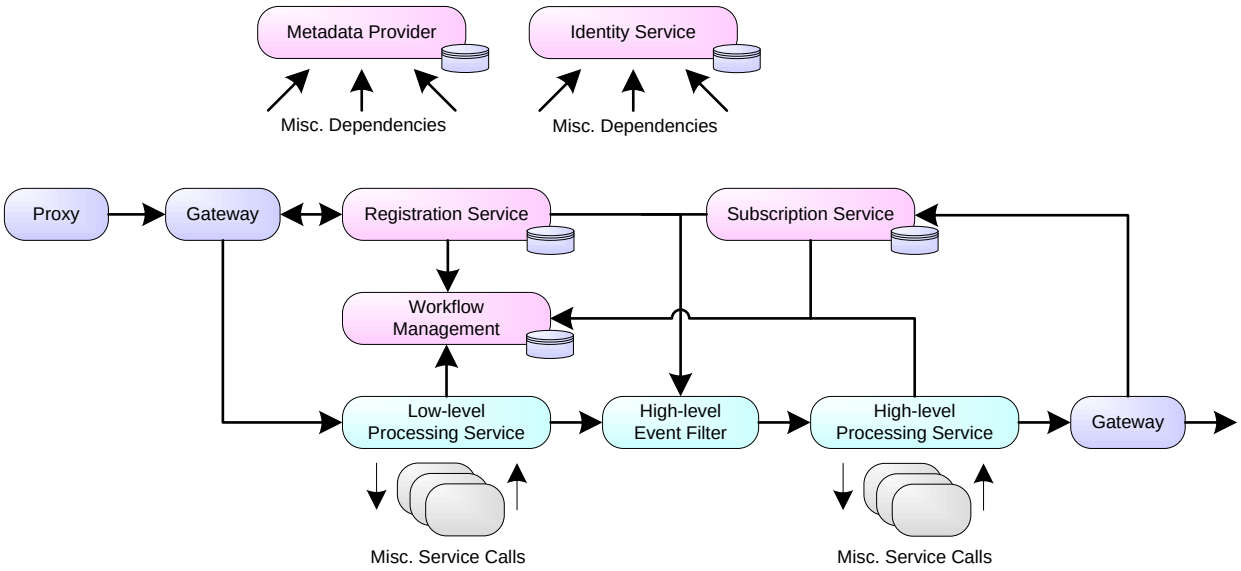


Abbildung 7.16.: Ablauf der Registrierung und Sensordatenübermittlung

```
SensorData {
    SensorId=Sensor1234,
    MapName=WeatherStationDataTypes
    Data {
        Temperature=20.3,
        Humidity=90.0,
        Photo=[...]
    }
}
```

Listing 7.2: Beispiel einer Sensordatennachricht

Sobald ein Gateway eine solche Nachricht empfängt (ggf. über einen Proxy), erzeugt dieses eine neue Instanz eines Verarbeitungsdienstes (engl. *Low-level Processing Service*), welcher für die weitergehende Verarbeitung zuständig ist. Sofern die Nachricht eine Kennung enthielt, wird der Workflow von der Workflow-Verwaltung abgerufen und es werden schließlich die Sensordaten an Variablen des Workflows gebunden. Für die Ausführung des Workflows wird eine eigenständige aktive Komponente erzeugt, welche diesen Schritt für Schritt abarbeitet und die aufgeführten Aktivitäten bzw. Dienste aufruft. Nach Abschluss der Bearbeitung übergibt der Verarbeitungsdienst die Sensordaten sowie eventuelle Ergebnisse der Ausführung des Workflows an den Filterdienst für ereignisbasierte Abfragen, wo alle Daten gegen die Abfragen der Konsumenten ausgewertet werden.

7.1.3.2. Beispiel für die Erstellung kontextbasierter Sichten

Ein Konsument interessiert sich für die Daten der Wetterstation aus vorherigem Beispiel. Sobald sich der erste jährliche Frost anbahnt, soll der Vermittler

eine Nachricht auf das Mobiltelefon des Konsumenten schicken. Da Mobiltelefone nicht ohne Weiteres aus dem Internet erreichbar sind [VHP06] und es sich hierbei um ein zukünftiges Ereignis handelt, was erst Wochen oder Monate später eintreten kann, soll die Nachricht der Einfachheit halber per SMS zugestellt werden.

Zunächst muss der Konsument eine Abfrage formulieren, welche das Ereignis eines nahenden Frostes repräsentiert. Listing 7.3 zeigt eine entsprechende Beispielanfrage in der Syntax der Esper *Event Query Language* [Esp13b]:

```
1 SELECT *
2 FROM pattern [[1] every (timer:interval(60 minutes)
3                          and not Sensor1234(temperature > 1.0))]
```

Listing 7.3: Beispiel einer Konsumentenanfrage

Diese Abfrage (angelehnt an [Esp13a]) ist erfüllt (d.h. sie „feuert“), sobald in einem Zeitintervall von 60 Minuten alle Temperaturmessungen der eintreffenden Sensordaten des Sensors mit der Kennung $Sensor1234$ (vgl. Listing 7.2) einen Wert kleiner als 1.0 aufweisen. Ist dieses Ereignis eingetreten, werden alle Daten des zuletzt empfangenen Datensatzes als Ergebnis an den konsumentenseitigen Verarbeitungsdienst zur Erstellung einer kontextbasierten Sicht weitergegeben.

Eine solche kontextbasierte Sicht wird nach Maßgabe des Konsumenten erzeugt. Sie soll alle relevanten Daten in einer geeigneten Repräsentation umfassen. Da in diesem Beispiel die Sicht als SMS an ein Mobiltelefon versendet werden soll, ist der Inhalt auf 160 Zeichen beschränkt. Der Konsument stellt sich die Nachricht folgendermaßen vor:

`Achtung Frost! Sensor1234 meldet 32.3° Fahrenheit.`

Sie besteht also aus statischen Textbausteinen, der Kennung des Sensors sowie der aktuellen Temperatur in Grad Fahrenheit. Um so eine Nachricht zu erstellen, muss der Konsument einen Workflow angeben, welcher aus dem Ergebnis des Filterdienstes obige Nachricht erzeugt. Der Workflow ist in Abbildung 7.17 dargestellt.

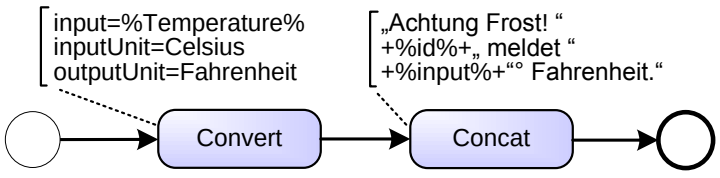


Abbildung 7.17.: Beispiel-Workflow zur Erstellung einer kontextbasierten Sicht

Zwei Aktivitäten sind im Rahmen des Workflows auszuführen: das Konvertieren von Grad Celsius nach Grad Fahrenheit (Aktivität „Convert“) sowie das Zusammenbauen der Nachricht (Aktivität „Concat“). Die Variablen, die durch %-Zeichen eingeschlossen sind, werden, wie im vorherigen Beispiel, wieder durch die entsprechenden Werte der Sensordaten substituiert. Kenntnis der Variablennamen und Aktivitäten seitens des Konsumenten werden vorausge-

setzt, bzw. der Konsument kann diese im Rahmen einer Metadatenabfrage vom Vermittler erfragen.

Nun kann eine Subskriptionsanfrage erstellt werden. Diese besteht aus obiger ereignisbasierter Abfrage, dem Workflow sowie zusätzlich der Angabe einer Endpunktadresse des Empfängers (z.B. $sms://+491701234567$). Letzteres ist erforderlich, da die Zustellung eines Ergebnisses in Form einer SMS an das Mobiltelefon übermittelt werden soll. Die komplette Subskriptionsanfrage kann nun an das Gateway versendet werden (vgl. Abbildung 7.16). Dieses kontaktiert den Identitätsdienst, um die Identität des Konsumenten zu hinterlegen und leitet daraufhin die Nachricht an den Subskriptionsdienst weiter. Dieser generiert eine eindeutige Kennung für die Subskription, den Subskribierenden sowie für den Empfänger, falls dieser nicht mit dem Subskribierenden identisch ist. Anschließend übergibt er den Workflow der Workflow-Verwaltung zur Validierung und Persistierung und subskribiert die ereignisbasierte Abfrage beim Filterdienst. Schließlich werden die Kennungen über das Gateway zurück an den Konsumenten gesendet.

Sobald die ereignisbasierte Abfrage beim Filterdienst subskribiert ist, wird sie gegen eintreffende Sensordaten abgeglichen. Sind alle Bedingungen der Abfrage (Kennung des Sensors, Schwellwert der Temperatur sowie das Zeitfenster) erfüllt, generiert der Filterdienst ein Ergebnis (auch als *komplexes Ereignis* bezeichnet) und instanziiert eine neue Komponente für den konsumentenseitigen Verarbeitungsdienst (engl. *High-level Processing Service*). Dieser kontaktiert zunächst die Workflow-Verwaltung, um den hinterlegten Workflow abzurufen, und bindet die Ergebnisdaten an die Variablen des Workflows. Eine neu instanziierte Komponente führt den Workflow aus und gibt das Ergebnis zurück an den Verarbeitungsdienst. Der letzte Schritt besteht darin, das Ergebnis dem Gateway zu übergeben, welches mittels des Identitätsdienstes die aktuelle Endpunktadresse des Empfängers ermittelt und diesem die Nachricht über einen SMS-Proxy zustellt.

7.1.4. Integration etablierter Standards

Der bisherige Entwurf der Architektur mit ihren Kernkomponenten ist zwar bereits in sich geschlossen, d.h. er repräsentiert die Basis zur Realisierung eines eigenständig lauffähigen Systems zur Vermittlung kontextbasierter Sichten, für einen Einsatz in der Praxis bedarf es jedoch der Unterstützung etablierter Standards, damit ein Vermittler in bestehende Systeme integriert werden kann.

Die Standards des Sensor Web sowie des Internet der Dinge verfolgen zwar grundlegend ähnliche Ziele und decken die wesentlichen funktionalen Aspekte ähnlich ab, jedoch unterscheiden sie sich in vielerlei Hinsicht. So standardisieren sie jeweils auf die entsprechenden Domänen zugeschnittene Daten- und Schnittstellenmodelle und sehen unterschiedliche funktionale Dienste vor. Daher sollen in diesem Abschnitt die wichtigsten Standards beider Welten aufgegriffen und in die Architektur eines Vermittlers integriert werden.

7.1.4.1. Sensor Web-Integration

Die *Sensor Web Enablement*-Initiative (SWE) des Open Geospatial Consortiums zeichnet sich für die Standardisierung von Daten- und Schnittstellenmodellen im Bereich des Sensor Webs verantwortlich (siehe Abschnitt 6.1.2). Die im Rahmen dieser Arbeit wesentlichen Standards beziehen sich auf die Schnittstellenmodelle des *Sensor Observation Service* (SOS) und des *Sensor Event Service* (SES) sowie auf die Datenmodelle *Sensor Markup Language* (SensorML) und *Observations & Measurements* (O&M).

Sensor Observation Service Der SOS dient dem Persistieren und Abfragen von Sensordaten und bietet zusätzlich Metainformationen über Sensoren und deren Sensordaten an [Ope12a]. Der SES erlaubt das Subskribieren ereignisbasierter Abfragen über flüchtige Sensordatenströme und hält ebenfalls Metainformationen über die erhebenden Sensoren und deren Daten vor [Ope08a]. Beide Dienstschnittstellen sehen die Verwendung der SensorML zur Repräsentation von Metadaten über Sensoren sowie die O&M-Spezifikation zur Beschreibung von Sensordaten vor. Diese Standards gilt es vermittlerseitig zu integrieren. Die Integration bezieht sich dabei auf das Anbieten derselben Dienstmethoden an der öffentlichen Schnittstelle eines Vermittlers, d.h. an der Schnittstelle des Gateways, sowie das Bereitstellen von Komponenten, welche die Anwendungslogik der Methoden realisieren und an welche entsprechende Anfragen seitens des Gateways delegiert werden können.

Die SOS-Schnittstelle spezifiziert (neben anderen, im Rahmen dieser Arbeit weniger relevanten Methoden) folgende Methoden [Ope12a]:

getCapabilities Beim Aufruf dieser Methode wird ein Capabilities-Dokument zurückgegeben, welches Metadaten über unterstützte Abfragefilteroperatoren und -operanden des Dienstes selbst sowie Metainformationen über alle vom Dienst verwalteten Sensordaten und deren Urheber enthält.

insertSensor Mittels dieser Methode kann ein neuer Sensor bei dem Dienst mittels einer SensorML-basierten Beschreibung registriert werden.

describeSensor Diese Methode liefert detaillierte Metadaten über einen einzelnen Sensor zurück.

insertObservation Mittels dieser Methode lassen sich neue Sensordaten in Form eines O&M-basierten Dokumentes hinzufügen.

getObservation Über diese Methode können Sensordaten gefiltert und abgefragt werden.

Normalerweise greift ein Klient direkt auf einen SOS zu, d.h. er versendet eine SOAP-Nachricht, welche direkt an die Web Service-Schnittstelle eines SOS adressiert ist, und bekommt im Fall einer Abfrage umgehend eine Antwort zurück. Bei einer Integration eines SOS in einen Vermittler muss dieses Verhalten nach außen hin beibehalten werden, damit auf Seite des Klienten keine

Änderungen vorgenommen werden müssen. Einzig adressiert ein Klient nun statt eines SOS ein Gateway, welches die SOS-Schnittstelle spiegelt und ankommende Nachrichten entsprechend weiterleitet.

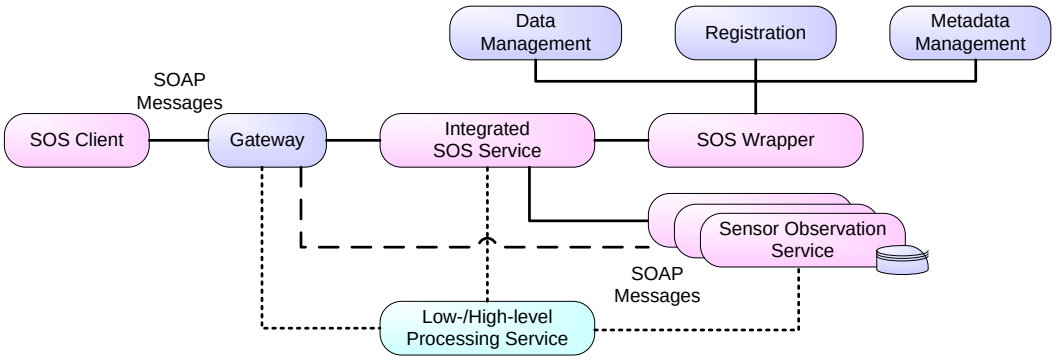


Abbildung 7.18.: Möglichkeiten zur Integration des Sensor Observation Service

Für die weitere Verarbeitung der Nachricht existieren verschiedene Möglichkeiten, welche in Abbildung 7.18 dargestellt und durch unterschiedliche Linienakzentuierungen hervorgehoben sind. Die wesentlichen Herausforderungen der Verarbeitung bestehen darin, dass die Datenquellen einerseits verteilt und andererseits heterogen sind. Als Datenquelle kommt nunmehr nicht nur ein einzelner SOS zum Einsatz, sondern unter Umständen mehrere SOS sowie nicht SWE-konforme Datenhaltungssysteme (z.B. die Datenverwaltung sowie die Metadatenbereitstellung). Es müssen also Abfrage- und Ergebnistransformationen angewendet, Ergebnisse unterschiedlicher Quellen zusammengefasst und Abfragen ggf. an mehrere Endpunkte weitergeleitet werden.

- Die durchgezogenen Linien zeigen den Nachrichtenfluss zwischen allen Beteiligten im Normalfall. Der Klient sendet eine SOAP-basierte SOS-Nachricht an die Web Service-Schnittstelle eines Gateways. Dieses leitet die Nachricht an eine spezielle Komponente, den *Integrierten Sensor Observation Service* (ISOS), weiter, welche für die Verteilung der Abfragen und das Zusammenfassen der Ergebnisse verantwortlich ist. Ein SOS kann in das System integriert werden, indem es direkt beim ISOS registriert wird[4]. Der ISOS wiederum kann dieselbe eintreffende Nachricht an alle ihm bekannten SOS weiterleiten, deren Antworten einsammeln und zusammenfassen.

  Allerdings ist ein SOS nicht die einzige Art von Datenquelle, in der Metadaten und Sensordaten gespeichert werden. Vielmehr werden Daten vermittlerseitig primär in dessen Datenverwaltung bzw. Registrie-

[4] Analog z.B. zur Metadatenbereitstellung kann auch der ISOS zustandslos realisiert und somit mehrere Male instanziiert werden, sofern dessen Datenhaltung entsprechend verteilt ausgelegt wird. Somit wirkt eine Registrierung bei einem beliebigen ISOS auf alle Instanzen. Statt ein SOS zu registrieren, kann auch ein *OpenGIS Catalogue Service* [Ope07a] registriert werden, welcher laufende SOS-Instanzen listet.

rung und Metadatenbereitstellung hinterlegt, welche aufgrund anderer Anforderungen nicht als SOS realisiert werden können. Aus diesem Grund sollten SOS-Abfragen von Klienten auch an diese Dienste weitergeleitet werden. Da diese jedoch andere Schnittstellen zum Abfragen, Hinzufügen und Löschen haben, müssen Übersetzungen vorgenommen werden. Hierfür wird vermittlerseitig eine SOS-Kapselung (engl. *SOS-Wrapper*) angeboten, d.h. ein SOS, dessen Datenbasis die Datenverwaltung sowie die Registrierung und Metadatenbereitstellung darstellen. Ein solcher Wrapper nimmt SOS-Abfragen entgegen, extrahiert deren Inhalte und verpackt diese in eine neue Nachricht an die entsprechenden Dienste. Da jedoch Datenverwaltung und Metadatenbereitstellung im Gegensatz zu der SOS-Datenhaltung lediglich semi-strukturierte Daten verwalten, kann bei der Übersetzung nur ein sogenannter *Best Effort-Ansatz* verfolgt werden, d.h. es kann keine Garantie für die korrekte Wiedergabe der Abfrage gegeben werden. Enthält beispielsweise eine SOS-Sensordatenabfrage ein räumliches Filterkriterium (z.B. alle Messungen im Areal zwischen lat_1, lon_1 und lat_2, lon_2) kann dieses von der Datenverwaltung nicht vollständig beantwortet werden, sofern keine räumliche Datenbank als Datenhaltungssystem verwendet wird.

- Die gestrichelte Linie in obiger Abbildung deutet die zweite Variante des Zugriffs auf ein SOS an. In diesem Fall leitet das Gateway die Nachricht nicht an ein ISOS, sondern an ein dediziertes SOS weiter. Falls ein Vermittler als geschlossenes System betrieben wird (z.B. von einer Institution oder Privatperson), kann ein einzelnes SOS vorausgesetzt und alle am Gateway eintreffenden SOS-Nachrichten an dieses weitergeleitet werden. Falls die Chance besteht, dass mehrere SOS-Instanzen (unterschiedlicher Betreiber) existieren, so muss dem Gateway explizit mitgeteilt werden, welche dieser Instanzen es ansprechen soll. Dies kann entweder durch entsprechende Konfiguration eines eigenen Gateways auf Seiten einer Institution geschehen oder durch Aufruf der entsprechenden $forward()$-Methode des Gateways, der sowohl die komplette SOS-Nachricht als auch der SOS-Endpunkt als Parameter übergeben werden. Bei Einsatz letzterer Möglichkeit verliert man jedoch klientenseitig die Zugriffskompatibilität auf das SOS, da der Aufruf einer Gateway-spezifischen Methode zwischengeschaltet ist.

- Eine dritte Möglichkeit, die Funktionen eines SOS in einen Vermittler zu integrieren, in der Abbildung durch gepunktete Linien angedeutet, stellen die Verarbeitungsdienste dar. Diese können nach Maßgabe eines Produzenten oder Konsumenten beliebig ausgewählt werden und werden von einem Vermittler im Rahmen einer Orchestrierung ausgeführt. Da der SOS eine Web Service-Schnittstelle anbietet, lässt sich dieser nahtlos als eine Aktivität in die Orchestrierungsbeschreibung eines Klienten einfügen. Somit lassen sich alle Methoden eines SOS auch indirekt über die Verarbeitungsprozesse aufrufen. Entsprechende Vor- und Nachverar-

beitung der Daten müssen hierbei durch weitere vor- bzw. nachgelagerte Dienste unterstützt werden. Auch der Zugriff auf den ISOS ist mittels dieser Variante möglich.

Es ist deutlich zu sehen, dass die Integration eines SOS in den Vermittlungsprozess nicht ohne Aufwand realisiert werden kann. Der SOS-Standard ist darauf ausgelegt, als einziger Dienst seiner Art in einem System betrieben zu werden. Eine Abfrage über mehrere SOS-Instanzen hinweg bedarf klientenseitig des Versands mehrerer Nachrichten, da ein SOS immer für sich alleine steht. Zwar ließe sich die Datenhaltung des SOS auch verteilt realisieren, jedoch bliebe der Dienst an sich als Flaschenhals und möglicher Single-Point-of-Failure bestehen. Außerdem speichert ein SOS sowohl Sensordaten als auch Metadaten, wodurch einerseits bei großen Datenmengen die Datenhaltung schwierig wird und andererseits die Forderung nach hoher Kohäsion verletzt wird. Der Aufwand, der durch Einführung neuer Komponenten und Schnittstellen in die Architektur eines Vermittlers entsteht, wird jedoch durch die Tatsache gerechtfertigt, dass der SOS-Standard in der Praxis eine bedeutende Rolle spielt und eine Integrationslösung daher unverzichtbar ist.

Sensor Event Service Die Integration eines SES gestaltet sich ähnlich, jedoch weitaus weniger aufwändig als die Integration eines SOS. Ein SES ist für das Verarbeiten ereignisbasierter Abfragen zuständig. Hierfür ist es notwendig, dass Abfragen (de-)subskribiert, Sensordaten übermittelt und die Abfragen gegen diese Daten abgeglichen werden können. Die Schnittstelle des SES sieht (neben anderen, im Rahmen dieser Arbeit weniger relevanten Methoden) folgende Methoden vor [Ope08a]:

getCapabilities Diese Methode erfüllt denselben Zweck wie beim SOS.

registerPublisher Diese Methode entspricht semantisch der $insertSensor$-Methode des SOS.

describeSensor Diese Methode entspricht semantisch der $describeSensor$-Methode des SOS.

notify Mittels dieser Methode kann ein Sensor einen neuen O&M-kodierten Datensatz übergeben. Im Gegensatz zum SOS wird dieser Datensatz jedoch nicht persistiert, sondern ausschließlich den Abfragefiltern des SES zugeführt.

subscribe Über diese Methode können Konsumenten Abfragefilter für Sensordaten beim SES registrieren. Stimmt ein Abfragefilter mit eintreffenden Sensordaten überein, so wird der entsprechende Konsument benachrichtigt.

Dieselben Aufgaben übernehmen im Vermittlungsprozess der Subskriptions- und der Verarbeitungsdienst sowie die Registrierung. Auch diese können aufgrund anderer Anforderungen hinsichtlich eines möglicherweise sehr hohen

Verarbeitungsaufkommens nicht direkt mittels der SWE-Standards realisiert werden, da der SES, genau wie der SOS, auf den Betrieb als alleinstehender Dienst ausgerichtet ist.

Zunächst muss die Schnittstelle der Gateways um das SES-Schnittstellenmodell erweitert werden. Produzenten, die ihre Daten SWE-konform veröffentlichen möchten, müssen lediglich die Nachrichten an ein Gateway adressieren. Dasselbe gilt für Konsumenten, die sich mit ihren ereignisbasierten Abfragen subskribieren oder Metadaten abfragen möchten.

Im Gegensatz zum SOS bringt es keinen besonderen Mehrwert, wenn mehrere SES in einem System vertreten sind, da dessen wesentliche Aufgabe die Detektion von Mustern in flüchtigen Daten ist. Ein SES selbst ist zwar zustandsbehaftet (Metadaten registrierter Produzenten sowie Abfragen der Konsumenten), jedoch ist es, im Gegensatz zu einem SOS mit allen enthaltenen Sensordaten, ohne großen Aufwand möglich, diesen Zustand zwischen Instanzen zu transferieren.

Die Integration des SES-Standards beschränkt sich daher auf eine Komponente (*SES Wrapper*), welche die Aufgaben eines SES an existierende Vermittlungskomponenten delegiert und dabei Datentransformationen durchführt (siehe Abbildung 7.19). Alle Abfragen, die über die SES-Schnittstelle eines Gateways eintreffen, können von einem solchen Wrapper bearbeitet werden.

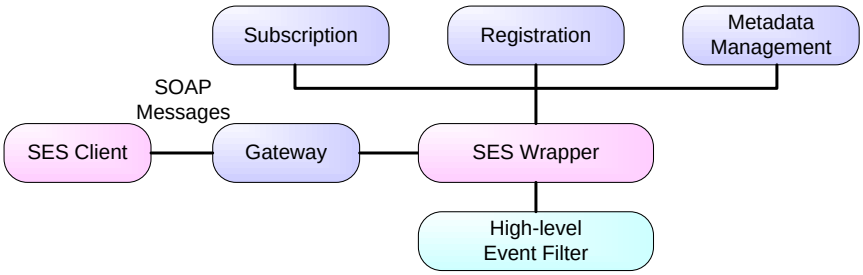


Abbildung 7.19.: Integration des Sensor Event Service

Beim Registrieren eines Produzenten (vgl. obige Methode $registerPublisher$) werden beispielsweise die Inhalte der SensorML-kodierten Registrierungsnachricht vom Wrapper extrahiert und an die Registrierung und Metadatenbereitstellung eines Vermittlers übergeben. Gleiches gilt für das Subskribieren einer ereignisbasierten Abfrage, welche gemäß des SWE-Standards in der Event Markup Language vorliegen muss, aber durch die Subskription in andere ereignisbasierte Abfragesprachen übersetzt werden kann (siehe [52N13]). Sendet ein Produzent Sensordaten im O&M-Format, können diese über den Wrapper an den Filterdienst (engl. *High-level Event Filter*) weitergegeben werden, welcher für die Auswertung der ereignisbasierten Abfragen verantwortlich ist. Führt eine SWE-konforme Abfrage zu einem Ergebnis, muss dieses dem Empfänger in Form einer O&M-kodierten Nachricht zugestellt werden. Dies entspricht dem Erstellen einer kontextbasierten Sicht und kann im Rahmen dieses Vorgangs durch entsprechende Transformationsdienste be-

werkstelligt werden. Natürlich steht diese Möglichkeit allen Konsumenten offen, unabhängig davon, ob sie SWE-konforme Abfragen registriert haben oder nicht.

7.1.4.2. Integration des Internet der Dinge

Da das Konzept des Internet der Dinge (engl. *Internet of Things*, IoT) unscharf verwendet wird, ist es schwierig, die Standards des IoT zu benennen. In Abschnitt 3.2.2 wurde mit der *Ding-orientierten Sichtweise* der Auto-ID Labs (bzw. EPCglobal und später GS1) ein Ansatz zur Standardisierung des Umgangs mit elektronischen Produktcodes (EPC) und für den Austausch von Ding-bezogenen Informationen vorgestellt [EPC10]. Im Rahmen des Vermittlungsprozesses gehören die Schnittstellen- und Datenmodelle des *EPC Information Services* (EPCIS) [EPC07] sowie des *Object Name Service* (ONS) [EPC08] zu den relevanten Standards, welche integriert werden sollen.

EPC Information Service Die Grundidee der Ding-bezogenen Sichtweise ist, dass Objekte mit einem oder mehreren RFID Chips ausgestattet werden, welcher mit einem eindeutigen EPC versehen ist. Somit bekommen die Objekte im Internet der Dinge eine eindeutige digitale Identität. Zu dieser Identität lassen sich beliebige zusätzliche Informationen in sogenannten *EPC Information Services* (EPCIS) hinterlegen [EPC07]. Es existiert jedoch kein zentrales EPCIS, welches Informationen zu allen Dingen enthält, sondern im Regelfall betreibt jede Institution, welche eine eigene *EPC Manager-Nummer* (ein eigenes Präfix im EPC-Namensraum) besitzt, auch ein eigenes EPCIS.

Die wesentliche Aufgabe eines EPCIS ist das Verwalten von EPCs und zu diesen hinterlegten Informationen. Zu solchen Informationen gehören beispielsweise Ereignisse, wie die Erfassung einer EPC durch ein Lesegerät, sowie auch Metainformationen, z.B. der Standort des Lesegeräts. Nutzer können Informationen einfügen und nach bestimmten Kriterien abfragen. Die Schnittstelle eines EPCIS ist entsprechend einfach und sieht (neben anderen, im Rahmen dieser Arbeit weniger relevanten Methoden) folgende Methoden vor [EPC07]:

capture Mittels dieser Methode können Klienten dem EPCIS eine Liste von Ereignissen oder Metadaten hinzufügen.

poll Diese Methode dient dem Abfragen von Ereignissen und Metadaten, wobei Abfrageparameter zum Eingrenzen der Ereignismenge angegeben werden können.

subscribe Es ist auch möglich, rudimentäre ereignisbasierte Abfragen beim EPCIS zu registrieren, welche periodisch ausgeführt werden. Nach jeder Ausführung wird ein subskribierter Klient über das Ergebnis informiert.

Das EPCIS unterscheidet sich somit nur unwesentlich vom Sensor Observation Service der Sensor Web Enablement-Initiative (siehe vorheriger Abschnitt 7.1.4.1). Aus diesem Grund verläuft auch die Integration sehr ähnlich.

Die Schnittstelle des Gateways wird dahingehend erweitert, dass dieser auch die Web Service-Schnittstelle für das Abfragen des EPCIS spiegelt. Das Hinzufügen neuer Daten geschieht gemäß der EPCIS-Standards mittels einfacher HTTP POST-Anfragen, sodass das Gateway eine weitere entsprechende Funktion anbieten muss.

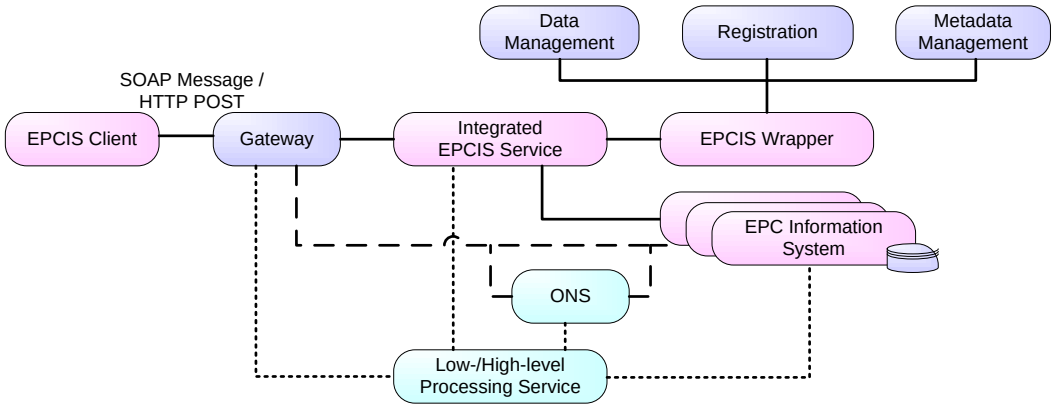


Abbildung 7.20.: Integration des EPC Information Service (EPCIS)

Für das Hinzufügen und Abfragen stehen vermittlerseitig mehrere Optionen zur Auswahl. Diese Optionen folgen der bereits im vorigen Abschnitt 7.1.4.1 erläuterten Integration eines Sensor Observation Service und sind in Abbildung 7.20 dargestellt.

- Ein integrierter EPCIS-Dienst (engl. *Integrated EPCIS Service*) nimmt Anfragen von Klienten über das Gateway entgegen (durchgezogene Linien in Abbildung 7.20). Diese Anfragen können einerseits an alle (oder ausgewählte) ihm bekannten EPCIS und andererseits an die vermittlerinterne EPCIS-Kapselung (engl. *EPCIS Wrapper*) weitergeleitet werden. Letztere transformiert die Anfrage in entsprechende Dienstaufrufe an die (Meta-)Datenverwaltung und Registrierung und übersetzt so zwischen den verwendeten Protokollen und Nachrichtenformaten. Im Falle einer Abfrage können die Ergebnisse von dem integrierten EPCIS Service aggregiert und an den Klienten zurückgeschickt werden.

- Bei der zweiten Option, dargestellt durch gestrichelte Linien, wird eine Anfrage lediglich an ein dediziertes EPCIS weitergeleitet. Das zu adressierende EPCIS kann vom Klienten entweder durch Angabe eines Parameters der $forward()$-Methode des Gateways festgelegt werden oder durch den *Object Naming Service* (siehe unten) anhand der EPC Manager-Nummer automatisch aufgefunden werden. Durch letztere Option bedarf es klientenseitig keinerlei Änderungen, sodass ein Vermittler transparent in bestehende Systeme integriert werden kann. Bei Abfrageoperationen funktioniert dies jedoch nur, sofern EPCs als Teil der Abfragefilter deklariert wurden. Ohne entsprechende Angaben können lediglich alle bekannten EPCIS abgefragt werden, wobei die Ergebnisse im

Anschluss durch den integrierten EPCIS-Dienst zusammengefasst werden müssen.

- Die gepunktete Linie deutet die Integration des integrierten EPCIS, des EPCIS und des Object Naming Service durch den Verarbeitungsdienst an. Klienten können im Rahmen eines Workflows beliebige Anfragen an diese Dienste stellen und die Ergebnisse durch den Verarbeitungsdienst weitergehend verarbeiten lassen.

Der EPCIS-Standard sieht zudem Abfragen vor, die periodisch oder in Abhängigkeit bestimmter Ereignisse ausgeführt werden. Bei Subskription solcher Abfragen muss der Klient einen Endpunkt angeben, an welchen zu späteren Zeitpunkten die Ergebnisse asynchron ausgeliefert werden. Hierfür wurde das sogenannte *Query Callback Interface* standardisiert [EPC07], welches eine Teilmenge der funktionalen Aufgaben des Gateways umfasst und somit ebenfalls von diesem realisiert werden kann.

Object Naming Service und Discovery Service Um zu einer beliebigen EPC ein (ggf. auch mehrere) EPCIS zu finden, welches weitergehende Informationen über das zugehörige Objekt vorhält, sehen die EPCglobal-Standards zwei verschiedene Möglichkeiten vor: den *Object Name Service* (ONS) [EPC08] und *EPC Discovery Services* (EPCDS) [EPC13].

Der ONS setzt auf dem bekannten *Domain Name System* (DNS) des Internets [PM87] auf, welches Domänennamen auf IP-Adressen abbilden kann. Die Grundidee des ONS ist, dass EPCs in Domänennamen (z.B. *000024.0614141.sgtin.id.onsepc.com*) konvertiert werden, mit welchen das normale DNS angefragt wird. Auch hier soll auf ein (oder mehrere) Adressen abgebildet werden, wobei diese Adressen Dienstendpunkte (z.B. ein EPCIS) repräsentieren. Der Domänenname wird von rechts nach links verarbeitet, wobei ein Segment jeweils von einem sogenannten *Nameserver* verarbeitet wird. Dieser kann eine Abfrage in der Hierarchie des DNS weiter nach unten reichen, sofern weitere Adresssegmente vorhanden sind. Von rechts gesehen das vorletzte Segment eines EPC-Domänennamens repräsentiert die EPC Manager-Nummer, d.h. den Präfix des Inhabers dieses Adressbereiches (welches durch die GS1-Organisation vergeben wird), z.B. eine bestimmte Firma. Diese Firma ist dafür verantwortlich, einen lokalen DNS-Server zu unterhalten, welcher für das letzte Segment, den Klassenbezeichner eines EPC, eine Liste von Dienstendpunktadressen vorhält, die weitere Informationen über die jeweilige Objektklasse liefern können.

Bei einer Abfrage an das ONS werden drei Parameter benötigt: i) die Adresse eines Nameservers, ii) der anzufragende Domänenname und iii) der angefragte Datensatz (engl. *Record*). Der erste Parameter ist meist durch den lokalen Internetanbieter vorgegeben und kann von einem Klienten außer Acht gelassen werden. Der zweite Parameter ist die zu einem Domänennamen konvertierte EPC und der dritte Parameter muss auf den *Naming Authority Pointer*

(NAPTR) gesetzt werden, was eine Art von Datensatz repräsentiert, in welchem mehrere Dienstendpunktadressen gespeichert werden können [EPC08].

Das Hinzufügen bzw. Ändern der Einträge geschieht ebenfalls über einfache DNS-Abfragen, wobei sich die Änderungen aufgrund der administrativen Zonen des Domain Name Systems im Wesentlichen auf der untersten Ebene der DNS-Hierarchie, also bei den Firmen, die für ihre EPCs eigene, lokale Nameserver betreiben, abspielen. Aus diesem Grund können im Rahmen des Vermittlungsprozesses zwar Abfragen an ein ONS gestellt werden, allerdings können keine Änderungen vorgenommen werden, da diese den jeweiligen Firmen vorbehalten sind. Ein eigenes lokales ONS, bzw. einen Nameserver, in einen Vermittler zu integrieren, ist daher nicht notwendig. Stattdessen wird ein Dienst angeboten, mittels dessen Abfragen an das ONS über das DNS ermöglicht werden.

Eine andere Möglichkeit, Informationen zu einer gegebenen EPC aufzufinden, stellen neben dem ONS die sogenannten EPC Discovery Services (EPCDS) dar. Beispiele hierfür sind die Arbeiten von Kürschner et al. [KCKT08] und Beier et al. [BGKR06]. Allerdings sind die EPCDS noch nicht durch EPCglobal standardisiert [EPC13], weshalb von einer Integration in die Architektur eines Vermittlers abgesehen wird.

7.1.5. Erweiterungen der Middleware

Nicht-funktionale Anforderungen an einen Vermittler betrafen unter anderem die Flexibilität und Erweiterbarkeit (vgl. Abschnitt 5.3.2). Flexibilität sollte u.a. hinsichtlich des Einsatzes in verschiedenen Konfigurationen ermöglicht werden, sodass ein Vermittler in unterschiedlichen Szenarien und unter Verwendung heterogener Ressourcenausstattung eingesetzt werden kann. Mit den Kernkomponenten wurde bereits eine minimale Konfiguration für ein lauffähiges System vorgestellt, welches die wesentliche Vermittlungsfunktionalität bietet. Um einen darüber hinausgehenden Mehrwert zu erbringen, lassen sich eine Vielzahl weiterer Funktionen zur Erweiterung eines Vermittlers vorstellen. Eine Auswahl funktionaler Erweiterungen (vgl. Abschnitt 5.3.1) soll im Folgenden vorgestellt werden.

7.1.5.1. Datenverwaltung

Die Kernfunktionalitäten eines Vermittlers sehen das Persistieren von Sensordaten nicht vor. Einerseits ist dies in vielen Anwendungsfällen nicht notwendig, da hier lediglich bestimmte zukünftige Ereignisse, über die ein Konsument informiert werden möchte, eine Rolle spielen. Andererseits kann das Persistieren von Sensordaten vieler Ressourcen bedürfen, was in manchen Systemkonfigurationen (z.B. wenn ein Vermittler auf einem mobilen Gerät ausgeführt wird) problematisch werden kann.

Aus diesen Gründen stellt die Datenverwaltung eine optionale Komponente dar, die je nach Bedarf in einen Vermittler eingebunden werden kann. Das Einbinden geschieht im Rahmen der Orchestrierungsbeschreibungen, welche

Produzenten und Konsumenten einem Vermittler zur Ausführung individueller Arbeitsschritte übergeben können. Das bedeutet, möchte ein Klient Kontextdaten persistieren lassen, muss er einen Workflow registrieren, welcher eine entsprechende Persistierungsaktivität beinhaltet. Bei Ausführung der Aktivität durch den Verarbeitungsdienst können auf diese Weise zu persistierende Daten an die Datenverwaltung übergeben werden. Ähnliches gilt für die Abfrage von persistierten Daten. Diese können entweder im Rahmen der Workflow-Ausführung über die Datenverwaltung abgerufen werden oder direkt mittels der $forward$-Methode eines Gateways, welche die logische Adresse der Datenverwaltung als Parameter verlangt.

Die Schnittstelle der Datenverwaltung (dargestellt in Abbildung 7.21) sieht hierfür eine Reihe generischer Methoden vor, welche die typischen *CRUD-Funktionen* (Create, Read, Update und Delete) von Datenhaltungssystem widerspiegeln. Diese Schnittstelle ist weitestgehend neutral hinsichtlich des eingesetzten Datenhaltungssystems, was der (optionalen) funktionalen Anforderung der Unterstützung unterschiedlicher Datenbanksysteme (vgl. Abschnitt 5.3.1.3) nachkommt.

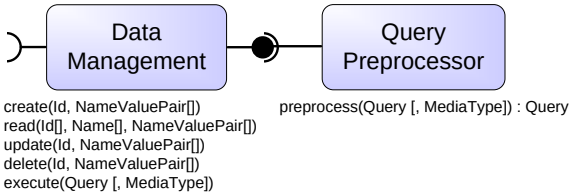


Abbildung 7.21.: Datenverwaltung: Angebotene Dienste

Eine Untersuchung verschiedenster Sensor Web-Infrastrukturen und deren Abfragesprachen in [Gue11] zeigt, dass sich viele Abfragesprachen auf ein gemeinsames abstraktes Schema abbilden lassen. Dieses besteht im Wesentlichen aus drei Teilen: i) der Eingabe-, ii) der Filter- und der iii) Ausgabedefinition. Am einfachsten lässt sich dieses Schema am Beispiel der *Structured Query Language* (SQL) nachvollziehen. Die Eingabedefinition entspricht dem *FROM*-Teil einer SQL-Abfrage, die Filterdefinition der Selektion und somit dem *WHERE*-Teil und die Ausgabe wird durch die Projektion, d.h. dem *SELECT*-Teil, dargestellt.

In [Gue11] werden mögliche Abbildungen für die *EPL*/*EQL* von Esper [Esp13a], die SQL-ähnliche Abfragesprache des *Global Sensor Networks* von Aberer et al. [AHS06b], die Ereignisfilter der *ALE Middleware* von EPCglobal [EPC11], das u.a. von der SWE-Initiative verwendete *XPath* [Wor10] sowie das ISO-standardisierte *OGC Filter Encoding* (FES) [Ope10b] und die *Event Pattern Markup Language* (EML) [Ope08a] dargestellt. Anzumerken ist, dass die Sprachen unterschiedlich ausdrucksstark bzw. mächtig sind und eine eindeutige Abbildung auf das dreiteilige Schema oft nur in einfachen Fällen möglich ist (siehe auch [Gue11]).

Die Abfrageparameter an der Schnittstelle der Datenverwaltung ist an dieses dreiteilige Schema angelehnt: Die Ein- und Ausgabedefinitionen werden

durch kommaseparierte Listen vorgenommen, die Filterbedingung mittels der Angabe von Name/Wert-Paaren. Eine entsprechende Abfrage sieht somit beispielsweise folgendermaßen aus:

```
read('temperature,humidity','Sensor1234','temperature>35')
```

Bei der Realisierung der Datenverwaltung müssen entsprechende Adapter diese allgemeine Form der Abfrage auf das konkret verwendete Datenhaltungssystem abbilden, d.h. zum Beispiel eine SQL-Abfrage oder einen XPath-Ausdruck konstruieren. Das Gleiche gilt auch für Datenbankmanagementsysteme, die keine deklarative Abfragesprache unterstützen. Operationen der Datenverwaltung zum Hinzufügen, Ändern und Löschen von Datensätzen erfordern ebenfalls Parameter, welche obiger Syntax folgen.

Bei komplexen Abfragen, welche nicht dem obigen Schema entsprechen oder nicht einfach durch kommaseparierte Listen und Name/Wert-Paare dargestellt werden können, bedarf es klientenseitig der Kenntnis des von der Datenverwaltung verwendeten Datenbankmanagementsystems und einer Abfrage in entsprechend unterstützter Syntax. Es obliegt dem Klienten eine syntaktisch korrekte Abfrage zu formulieren und diese über die generische *execute*-Methode ausführen zu lassen.

Diese Methode bietet noch einen zweiten, optionalen Parameter für multimediale Abfragen. Wie in Abschnitt 5.3.1.6 gefordert, sollen auch Abfragen in Form von Audio- oder Video-/Bilddaten unterstützt werden. Derlei Abfragen werden in Abhängigkeit von ihrem Medientyp durch die Abfragevorverarbeitung (engl. *Query Preprocessing*) in eine textuelle Form umgewandelt (bspw. mittels Spracherkennung) und für die letztendliche Abfrage der Datenverwaltung aufbereitet.

7.1.5.2. Mobilitätsunterstützung und push-basierte Nachrichtenzustellung

Mobile Geräte kommunizieren meist mittels drahtloser Kommunikationstechniken. Wesentliches Charakteristikum hierbei ist, dass die Verbindungsqualität erheblichen Schwankungen unterliegt. Das bedeutet, Datenraten und Nachrichtenlaufzeiten können beträchtlich schwanken und Verbindungen unvorhersehbar unterbrochen werden. Letzteres führt dazu, dass mobile Klienten nach wiederholtem Verbindungsaufbau häufig eine neue physische Endpunktadresse besitzen. Beim Publizieren von Sensordaten seitens der Produzenten spielen Adressänderungen keine Rolle, da die Kommunikation vom Produzenten initiiert wird und dieser bei einem plötzlichen Verbindungsabbruch für das wiederholte Versenden einer Nachricht verantwortlich ist. Bei allen Kommunikationsakten jedoch, die eine unmittelbare Antwort seitens eines Vermittlers erwarten (z.B. dem Abfragen von Sensor- oder Metadaten) sowie allen Akten, bei denen ein Vermittler proaktiv eine Verbindung initiiert, d.h. die Nachrichten *push-basiert* übermitteln möchte (z.B. dem Zustellen von Ergebnisdaten ereignisbasierter Abfragen oder dem Tasking), stellen Verbindungsabbrüche

und Adresswechsel eine Herausforderung dar, welche ein Vermittler adäquat unterstützen muss. Dies umfasst im Wesentlichen zwei Aspekte:

1. Ein Vermittler muss die aktuelle physische Endpunktadresse eines mobilen Gerätes kennen, damit bei proaktivem Verbindungsaufbau das richtige Gerät adressiert wird.

2. Ein Vermittler muss Nachrichten, die einem Klienten nicht zugestellt werden können, zwischenspeichern und zu einem späteren Zeitpunkt erneut zustellen.

Der erste Aspekt wird bereits durch den Identitätsdienst realisiert, welcher für die Verwaltung physischer und logischer Endpunktadressen zuständig ist. Für das Zwischenspeichern und wiederholte Zustellen von Nachrichten bedarf es einer Komponente zur Nachrichtenverwaltung. Diese muss, ähnlich z.B. der Metadatenbereitstellung, zustandslos und mehrfach instanziierbar sein, Nachrichten mittels eines verteilten Datenhaltungssystems persistieren und diese unter Verwendung verschiedener Strategien zu einem späteren Zeitpunkt erneut zustellen können.

Für das Zustellen von Nachrichten an die Klienten zeigen sich grundsätzlich die Gateways verantwortlich. Bei Auftreten eines Kommunikationsfehlers müssen diese die Nachricht an die Nachrichtenverwaltung übergeben, wo sie zunächst persistiert wird. Für das wiederholte Zustellen können verschiedene Strategien angewandt werden:

- Die Nachrichtenverwaltung könnte in festgelegten Zeitintervallen die Nachricht zur erneuten Zustellung an das Gateway übergeben, welches seinerseits versucht, den Klienten unter der letztbekannten physischen Endpunktadresse zu erreichen. Dies entspricht einer *push-basierten* Nachrichtenübermittlung.

- Jedes Mal wenn das Gateway eine Nachricht von einem Klienten erhält und dieses die Adressinformationen beim Identitätsdienst aktualisiert, fragt es bei der Nachrichtenverwaltung nach, ob für den Klienten Nachrichten vorliegen und überstellt diese gegebenenfalls.

- Klienten fragen über das Gateway in bestimmten Zeitintervallen explizit nach, ob aktuell Nachrichten für sie vorliegen. Dies entspricht einem *pull-basierte* Abruf.

Die push-basierte Strategie zur Wiederholten Zustellung kann jedoch nicht nur bei fehlgeschlagenen Übertragungen hin zu mobilen Geräten sinnvoll sein. Auch andere Geräte können eine solche Art der Übermittlung bevorzugen, da push-basierte Kommunikation eine zeitnahe Zustellung von Nachrichten ermöglicht und kein wiederholtes Abfragen von Klienten, ob Nachrichten verfügbar sind, notwendig ist.

Abbildung 7.22 zeigt die Schnittstelle der Nachrichtenverwaltung. Diese definiert eine Methode zum push-basierten Senden von Nachrichten, welche das

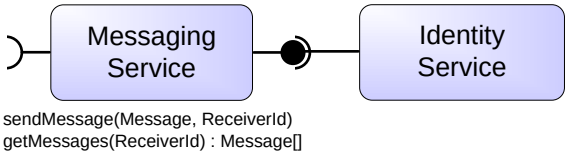


Abbildung 7.22.: Nachrichtenverwaltung: Angebotene und benötigte Dienste

Gateway aufruft, sofern der Nachrichtenversand seinerseits fehlgeschlagen ist und eine weitere Methode zum Abrufen von Nachrichten, welche ebenfalls durch das Gateway aufgerufen wird, sofern eine Verbindung von einem Klienten aufgebaut wurde und Nachrichten pull-basiert abgefragt werden.

7.1.5.3. Semantische Abbildungen und Ableitungen

Sensordaten und Metadaten haben zunächst keine eigene Semantik. Möchte man die Daten interpretieren und semantische Interoperabilität zwischen Diensten sicherstellen, so bedarf es einer gemeinsamen Übereinkunft von Produzenten und Konsumenten über die Bedeutung dieser [JSB+09]. Das geschieht zunächst implizit, d.h. diese Übereinkunft wird den Daten nicht explizit angehaftet, sondern folgt entweder einer Konvention oder geschieht durch die Namensgebung von Attributen. Zum Beispiel kann ein Konsument das Tupel $(Hamburg, 24.12.2012, 10, Grad)$ dahingehend interpretieren, dass die Außentemperatur in Hamburg (Deutschland) am 24.12.2012 10° Celsius beträgt, es könnte aber auch die Stadt Hamburg in Pennsylvania adressiert sein, die mit 10° Fahrenheit einen kalten Wintertag erlebt. Es könnte jedoch auch gemeint sein, dass die Stadt Hamburg (Deutschland) am 10. Grad östlicher Breite zu finden ist.

Es gibt eine Reihe von Möglichkeiten um Daten explizit eine Bedeutung zu verleihen. In der Praxis des *Semantic Webs* haben sich hierfür insbesondere das *Resource Description Framework* (RDF) [W3C04d] zur Formulierung logischer Aussagen (z.B. Beziehungen) über Dingen sowie *RDF Schema* [W3C04c] und die *Web Ontology Language* (OWL) [W3C04b] zur Beschreibung von Ontologien und die Abfragesprache *SPARQL* [W3C08] durchgesetzt. Ontologien definieren dabei explizite Konzeptualisierungen, Beziehungen zwischen diesen sowie Inferenz- und Integritätsregeln. Annotiert man obiges Tupel mit entsprechenden Referenzen auf die Konzepte einer Ontologie, ist eine zweifelsfreie Interpretation möglich. Zusätzlich kann man weiteres Wissen aus semantisch annotierten Tupeln ableiten, sofern entsprechende Regeln definiert werden, z.B. dass Weihnachten 2012 in Hamburg (Deutschland) ungewöhnlich warm war.

Um derartige semantische Abbildungen und Ableitungen unterstützen zu können, muss ein Vermittler dahingehend erweitert werden, dass dieser mit semantisch annotierten Sensor- und Metadaten umgehen kann. Zwei Ansätze aus dem Bereich des Sensor Webs zeigen grundsätzliche Möglichkeiten auf.

Semantic Sensor Observation Service In [HPST09, PHPS10] wird mit dem *Semantic Sensor Observation Service* (SemSOS) eine Erweiterung der Standards der Sensor Web Enablement-Initiative (siehe Abschnitt 6.1.2) beschrieben. Dieser Dienst kann Konsumenten auf Anfrage semantisch annotierte Sensorbeschreibungen und Messwerte zurückliefern. Hierfür haben die Autoren Ontologien zur Konzeptualisierung von Messwerten und Metadaten für Sensoren entworfen, sowie eine Wissensbasis, in der einerseits semantisch annotierte Daten sowie andererseits Regelsätze für semantische Ableitungen gespeichert sind. Eine Abfrage an den SemSOS resultiert in semantisch annotierten Sensordaten, welche dem *Observations & Measurements*-Datenmodell folgt. Dieses Modell kann ein Klient zweifelsfrei interpretieren. Darüber hinaus kann mit Hilfe eines Inferenzmechanismus zusätzliches Wissen abgeleitet und in der Antwort mitgeschickt werden. Liegen beispielsweise keine Messwerte für die Temperatur in Hamburg vor, könnten die Daten einer nahegelegenen Messstation zurückgeliefert werden oder einem Datensatz zusätzlich ein Wahrscheinlichkeitswert für das Auftreten von Bodenfrost angehängt werden.

Da SemSOS vollständig kompatibel zu den OGC-Standards ist, könnte dieses genau wie ein normales SOS in den Vermittlungsprozess integriert werden (siehe Abschnitt 7.1.4).

Semantic Enablement Layer Während SemSOS lediglich eine Lösung für den Sensor Observation Service darstellt, stellt die *Semantic Enablement Layer* (SEL) eine semantische Schicht dar, welche über allen Diensten der Sensor Web Enablement-Initiative steht [JSB+09]. Sie sorgt für den transparenten Zugriff auf semantisch angereicherte Daten nicht nur des SOS, sondern auch der anderen SWE-Dienste. Sie abstrahiert von den konkreten Technologien des Semantic Webs und ist daher flexibler einsetzbar als SemSOS.

Bestandteile der SEL sind der *Web Ontology Service* (WOS) und der *Web Reasoning Service* (WRS). Aufgabe des WOS ist die Verwaltung und das Bereitstellen von Ontologien, einerseits für die standardisierten Datenmodelle wie SensorML und O&M und andererseits für domänenabhängige, konkrete Anwendungen. Der Dienst ist als ein Profil des *Web Catalog Service* [Ope07a] definiert und nutzt somit existierende OGC-Standards. Der WRS realisiert Schlussfolgerungs- und Inferenzfunktionen und ist als ein Profil des OGC-*Web Processing Service* (WPS) [Ope07c] definiert.

Zur Laufzeit kann der WRS mit Hilfe des WOS semantische Abbildungen und Ableitungen vornehmen und so eine gegebene Abfrage mit semantisch ähnlichen Konzepten anreichern und die Ergebnisse mit semantischen Annotationen versehen, welche verwendete Ontologien referenzieren.

Für eine Integration in den Vermittlungsprozess müssten Instanzen des WOS und WRS aufgesetzt werden, welche mittels eines Web Processing Services koordiniert werden. Für die Klienten müsste die Schnittstelle des WPS nach außen hin durch die Gateways gespiegelt und eintreffende Nachrichten von diesen entsprechend weitergeleitet werden. Die Integration erfolgt somit analog der Integration des Sensor Observation Service (vgl. Abschnitt 7.1.4).

7.1.5.4. Typische und benutzerdefinierte Verarbeitungsfunktionen

Ein Vermittler kann einfach um Verarbeitungsfunktionen für Kontextdaten erweitert werden. Hierfür muss lediglich eine Komponente (aktive Komponente oder Web Service) bereitgestellt werden, welche die entsprechenden Funktionen realisiert. Typische Verarbeitungsfunktionen werden dabei direkt vom Vermittler angeboten und können von diesem bei Bedarf auch instanziiert und innerhalb der eigenen Infrastruktur ausgeführt werden. Im Gegensatz dazu unterliegen benutzerdefinierte Verarbeitungsfunktionen nicht der Verantwortung eines Vermittlers. Sie werden extern von Dritten angeboten und lediglich von einem Vermittler im Rahmen der Ausführung von Verarbeitungs-Workflows aufgerufen.

Sofern Verarbeitungsfunktionen öffentlich zugreifbar sein sollen, müssen diese bei der Metadatenbereitstellung registriert werden, damit ein Klient Kenntnis über die Funktionen und die Art ihrer Nutzung erlangen kann. Bei typischen Verarbeitungsfunktionen, welche durch einen Vermittler angeboten werden sollen, muss zudem der Programmcode zugreifbar sein, damit der Vermittler diesen bei Bedarf ausführen kann. Ein Klient kann bei der Metadatenbereitstellung alle angebotenen Verarbeitungsfunktionen abfragen. Gewünschte Funktionen können im Rahmen des Verarbeitungs-Workflows referenziert werden und werden aufgerufen, sobald dieser vom Verarbeitungsdienst ausgeführt wird.

7.1.5.5. Benutzerdefinierte Verarbeitungsfunktionen

In Abschnitt 5.4.3 wurden verschiedene Möglichkeiten der Injektion benutzerdefinierter Verarbeitungsfunktionen in einen Verarbeitungs-Workflow präsentiert. Hierbei sind zwei Ansätze zu unterscheiden: i) Spezifikation externer Dienstaufrufe (z.B. Web Services, Active Component Services oder direkt über Socket-Verbindungen) im Rahmen der Workflow-Beschreibung und ii) Bereitstellen von Programmbibliotheken zur Laufzeit, sodass der Programmcode bei Bedarf geladen und lokal ausgeführt werden kann.

In ersterem Fall muss für jede Art von Dienst (Web Service, Jadex Service etc.) ein sogenannter *Wrapper* zur Verfügung gestellt werden, welcher den eigentlichen Dienstaufruf kapselt. Dessen Schnittstelle muss dahingehend generisch sein, dass alle für den Aufruf erforderlichen Informationen (z.B. Endpunktadresse, Ein- und Ausgabeparameter etc.) über die entsprechende Aktivität im Workflow bereitgestellt werden können. Der eigentliche Dienstaufruf erfolgt dann transparent durch den Wrapper bzw. die Laufzeitumgebung, die den Workflow ausführt, evtl. notwendige Zusatzartefakte erzeugt, den Dienst aufruft und das Ergebnis schließlich an den spezifizierten Ausgabeparameter bindet.

Das Bereitstellen eigener Programmbibliotheken zur Instanziierung durch einen Vermittler kann auf unterschiedliche Weisen erfolgen. Im trivialen Fall fügt man die Bibliothek einem Knoten hinzu, der an dem Vermittlungsprozess beteiligt ist, passt ggf. dessen Konfiguration an und startet den Knoten neu.

Dieser Ansatz wird als invasiv bezeichnet, da manuelle Anpassungen an der Konfiguration bzw. dem Code eines Vermittlers erforderlich sind. Außerdem müssen alle Dienstaufrufe an diesen Knoten delegiert werden, da nur hier der Programmcode vorhanden ist (siehe auch unten, „Zustandsbehaftete Verarbeitungsdienste“).

Eine andere Möglichkeit ist das Hinzufügen der Bibliothek zur Laufzeit durch einen dynamischen Lademechanismus. Hierbei muss die Bibliothek über das Netzwerk zugreifbar sein und der Ausführungsumgebung die Netzwerkadresse bekanntgemacht werden. Sobald die Bibliothek zur Ausführung benötigt wird und lokal nicht aufgefunden werden kann, wird sie von der entfernten Adresse über das Netz geladen. Auch dieser Ansatz ist problematisch, sofern das Binden nicht über eindeutige Kennungen, sondern über Namen geschieht (wie z.B. bei der Java-Laufzeitumgebung). So könnten zwei Anwender unterschiedliche Funktionen mit gleichem Namen bereitstellen, wobei die eine Funktion die andere überdecken und daher unter Umständen fälschlicherweise ausgeführt würde. Insbesondere betrifft dies auch den Umgang mit verschiedenen Versionen derselben Bibliothek. In beiden Fällen müsste eine Referenz auf den Knoten, welcher über die Bibliothek verfügt, für die Dienstsuche in der Metadatenbereitstellung eingetragen werden, da bei verteilter Ausführung nicht sichergestellt werden kann, dass der Knoten, bei dem die Bibliothek initial hinzugefügt wurde, auch der den Workflow ausführende Knoten ist. Das bedeutet der Aufruf der benutzerdefinierten Funktion muss zunächst über eine Suche in der Metadatenbereitstellung an den entsprechenden Knoten delegiert werden.

Eine dritte Möglichkeit zeigt das *Jadex Active Components*-Rahmenwerk (vgl. Abschnitt 7.2.1.1): Hier können externe Bibliotheken (und Workflows) als sogenannte *Maven-Artefakte* bereitgestellt werden. Maven ist ein *Build- und Konfigurations-Management*-Werkzeug [Son09], welches unter anderem für die Verwaltung von Programmressourcen (z.B. Bibliotheken) und deren Abhängigkeiten verantwortlich ist. Mittels dieses Werkzeugs ist es möglich, einem Programm zur Laufzeit neue Abhängigkeiten (z.B. zu den benutzerdefinierten Verarbeitungsfunktionen) bekanntzumachen. Deklariert und veröffentlicht man Programmbibliotheken als Maven-Artefakte, so können diese zur Laufzeit in ein mit Maven verwaltetes System automatisch integriert werden.

7.1.5.6. Zustandsbehaftete Verarbeitungsdienste

Im Normalfall sind Verarbeitungsdienste zustandslos, d.h. sendet ein konkreter Produzent zwei Ereignisse an einen Vermittler, wird für jedes Ereignis ein neuer Verarbeitungs-Workflow und im Zuge dessen auch für jede darin definierte Verarbeitungsfunktion eine neue Instanz erzeugt.

In manchen Anwendungsfällen kann es jedoch wünschenswert sein, zustandsbehaftete Verarbeitungsdienste zu verwenden. Ein Beispiel hierfür wären Dienste, die mittels Techniken des unüberwachten Lernens von sich

aus Sensordaten klassifizieren können. Ein solcher Dienst könnte intern beispielsweise mittels eines künstlichen neuronalen Netzes realisiert werden, welches neu eintreffende Ereignisse auf Basis bereits bekannter Ereignisse klassifiziert. Hierbei werden unter anderem die Kantengewichtungen zwischen den Neuronen adaptiert, welche sozusagen den Zustand des Dienstes repräsentieren.

Um einen solch zustandsbehafteten Dienst realisieren zu können, gibt es zwei Möglichkeiten: i) der Dienst an sich ist nicht zustandsbehaftet, sondern der Zustand wird nach jedem Aufruf persistiert und vor jedem Aufruf wiederhergestellt oder ii) von dem zustandsbehafteten Dienst existiert lediglich eine Instanz und jeder Aufruf muss an diese delegiert werden.

Ersterer Fall wurde in [Fol11] umgesetzt. Hier wurden Zustandsinformationen in einem Datenhaltungssystem hinterlegt und bei jedem Dienstaufruf neu geladen. Problematisch ist dieser Ansatz, sofern die Gefahr des nebenläufigen Dienstaufrufs besteht, da eine Synchronisierung der Aufrufe nur schwerlich umzusetzen ist und die Gefahr besteht, dass gleichzeitige Aufrufe unterschiedliche Zustände fest- bzw. überschreiben.

Der zweite Fall kann ähnlich der Realisierung benutzerdefinierter Verarbeitungsfunktionen umgesetzt werden. Von einem Vermittler angebotene Dienste müssen der Metadatenbereitstellung bekanntgemacht werden. Soll eine Instanz global einmalig existieren, reicht die Registrierung lediglich dieser einen Instanz. Sollen mehrere Instanzen desselben Diensttyps erlaubt werden, so müssen deren Dienstbeschreibungen bei initialer Instanziierung mit einer entsprechenden eindeutigen Markierung in der Metadatenbereitstellung versehen werden. Bei externen Diensten ist der jeweilige Betreiber für die Fortführung des Zustands sowie die Synchronisierung der Zugriffe verantwortlich.

7.1.5.7. Protokollierung

Bei der Protokollierung (engl. *Logging*) geht es zunächst darum, bedeutende Systemereignisse aufzuzeichnen. Solche Ereignisse können z.B. das Versenden oder Empfangen einer Nachricht, der Aufruf einer bestimmten Methode oder eines Dienstes, das Ändern einer Variablenbelegung oder das Auftreten eines Fehlers darstellen. Die jeweils zu protokollierenden Informationen sind dabei abhängig von der Art des Ereignisses. Beispielsweise sind beim Versenden einer Nachricht der Sender und Empfänger sowie ggf. bestimmte Inhalte der Nachricht relevant. Beim Aufruf einer Methode oder Dienstes sind deren Name sowie die Aufrufparameter von Interesse. Außerdem sind grundsätzlich Zeitstempel sowie ggf. eine Art von Vorgangsbezeichner für jedes Ereignis zu protokollieren. Zeitstempel können dabei entweder näherungsweise die absolute Zeit oder eine logische Zeit (zur Bestimmung der sogenannten *Happened-Before*-Beziehung [TS07]) repräsentieren. Vorgangsbezeichner dienen der Zuordnung eines Logbuch-Eintrags zu einem bestimmten Vorgang (z.B. das Ent-

gegennehmen und Verarbeiten einer Sensormessung) und der späteren Filterung von Einträgen bei Abfragen.

Einträge in einem Logbuch können für verschiedene Anwendungszwecke hilfreich sein. Häufig werden sie bei der Fehleridentifikation verwendet, da sie das Systemverhalten zu bestimmten (logischen) Zeitpunkten widerspiegeln und ggf. weiterführende Zustandsinformationen einzelner Entitäten enthalten. Bei einem Fehler kann so nachvollzogen werden, um was für eine Art Fehler es sich handelt und anhand vorheriger Ereignisse sowie den Zustandsinformationen ggf. auch warum dieser aufgetreten ist. Unter Umständen ist es auch möglich, anhand des Logbuchs das Systemverhalten zu reproduzieren, um an bestimmten Haltepunkten einen detaillierten Blick auf den Zustand einzelner Entitäten zu werfen.

Das Protokollieren von Systemereignissen kann jedoch einen erheblichen Einfluss auf die Leistung des Systems haben. An der Verarbeitung von Sensordaten oder bei der Erstellung kontextbasierter Sichten sind eine Vielzahl an Komponenten beteiligt. Protokolliert jede dieser Komponenten die eigenen Aufrufe sowie ggf. weitere Verarbeitungsschritte und sendet diese Information an einen (zentralen) Protokollführer, besteht ein Großteil der Verarbeitungslast womöglich aus derlei Logging-Vorgängen, was die Verfügbarkeit von Ressourcen für die eigentliche Verarbeitung der Sensordaten stark einschränkt. Aus diesem Grund gehört das Logging zu den Erweiterungen der Middleware und Logging-Strategien sollten mit Bedacht ausgewählt werden. Eine ressourcenschonende Strategie wäre z.B., lediglich Fehler zu protokollieren und erst bei Auftreten eines Fehlers weitere Protokollfunktionen zu aktivieren.

Ein adäquater Protokollmechanismus im Rahmen des Vermittlungsprozesses sollte hinsichtlich der Anforderungen an Skalierbarkeit und Robustheit verteilt ausgelegt sein. Für die Realisierung eines verteilten Protokollmechanismus existieren mehrere Möglichkeiten, die jeweils unterschiedlich starke Auswirkungen auf den Leistungsüberhang, die Robustheit und die Ordnung des Protokolls haben. Leistungsüberhang betrifft hierbei den Verbrauch an Rechen-, Speicher- und Kommunikationsressourcen. Robustheit bezieht sich darauf, ob das Protokoll bei Ausfall einzelner beteiligter Komponenten noch sinnvoll zu lesen ist und die Ordnung bestimmt, ob die Einträge in einer bestimmten zeitlichen oder logischen Reihenfolge geordnet werden können.

Eine naheliegende Möglichkeit ist die Umsetzung eines entsprechenden Protokolldienstes analog der Metadatenbereitstellung, d.h. eine zustandslose Komponente, welche beliebig häufig instanziiert werden kann, deren Instanzen über ein verteiltes Datenhaltungssystem miteinander verbunden sind und welche Protokollierungsfunktionen über eine Dienstschnittstelle anbietet. Abbildung 7.23 stellt die Schnittstelle eines solchen Dienstes dar. Die generischen Methoden erlauben das Hinzufügen und Abfragen typisierter Protokolleinträge.

Die Vorteile dieser Lösung liegen in der Skalierbarkeit, wobei jedoch durch den zusätzlichen Nachrichtenversand ein signifikanter Mehraufwand entsteht. Robustheit ist gegeben, sofern mehrere Instanzen dieser Komponente

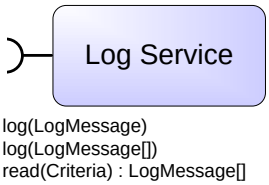


Abbildung 7.23.: Protokolldienst: Angebotene Dienste

im Netz verteilt werden. Es ist allerdings nicht möglich, eine partielle Ordnung der Logeinträge sicherzustellen, da keine globale Sicht auf das System existiert, einzelne Komponenten gänzlich unabhängig von anderen sind und keine expliziten Ordnungsinformationen in den ausgetauschten Nachrichten enthalten sind.

Eine andere Möglichkeit wäre die Integration eines (verteilten) Datenhaltungssystem in jede einzelne Komponente, sodass Protokolleinträge einfach der lokalen Datenhaltungsinstanz hinzufügt werden können. Im Fall einer verteilten Datenhaltung kann das komplette Logbuch einfach durch entsprechende Abfrage bei einer beliebigen Datenhaltungsinstanz ausgelesen werden. Liegt lediglich eine lokale Datenhaltung vor, so muss zur Abfrage des kompletten Logbuchs jede Komponente einzeln abgefragt werden. Je nach Art des verwendeten Datenhaltungssystems weisen beide Varianten einen verhältnismäßig hohen Ressourcenbedarf auf, da eine Vielzahl an Datenhaltungsinstanzen laufen müssen. Auf der anderen Seite existieren jedoch durch die Wahl geeigneter Datenhaltungssysteme eine Reihe von Möglichkeiten zur Optimierung der Protokollierungsleistung (z.B. durch Verwendung lokaler In-Memory-Datenbanken). Doch auch bei diesen Varianten ist keine partielle Ordnung der Logeinträge möglich, da sich die Komponenten zeitlich nicht genau synchronisieren können.

Um eine partielle Ordnung der Einträge zu erhalten, können in verteilten Systemen logische Zeitstempel verwendet werden. Diese beruhen darauf, dass alle Komponenten Zählvariablen führen, die den Logeinträgen hinzugefügt werden. Die Variablen werden beim Auftreten von Ereignissen hochgezählt und den auszutauschenden Nachrichten angehängt, wodurch sich die Komponenten untereinander synchronisieren können. Die *Lamport-Uhr* [Lam78] sowie deren Erweiterung zur *Vektoruhr* [TS07] stellen prominente Verfahren zur Bestimmung einer partiellen Ordnung von Ereignissen dar. Für beide Verfahren müssten jedoch die Schnittstellen aller Komponenten, bzw. die Signaturen aller angebotenen Dienstmethoden, dahingehend erweitert werden, dass diese den Austausch von Zeitstempeln ermöglichen. Da diese Art der Erweiterung hochgradig invasiv ist und einen signifikanten Überhang erzeugt, soll sie nicht weiter berücksichtigt werden.

Zur Realisierung einer effizienten Protokollierung eignet sich eine Kombination oben erwähnter Aufzeichnungsvarianten. Jede Komponente führt ein eigenes lokales Logbuch, in welchem alle Ereignisse innerhalb eines bestimmten Zeitraums festgehalten, jedoch nicht persistiert werden (ein sogenanntes

Rolling Event Log). Zudem wird die Schnittstelle aller Komponenten um eine Methode zum entfernten Abruf dieses Logbuchs erweitert. Über oben beschriebenen Protokolldienst (vgl. Abbildung 7.23), welcher Methoden zur Aufzeichnung und Persistierung von Logeinträgen anbietet, können dann Logeinträge oder Logbücher abgerufen und festgeschrieben werden. Methoden zum Festschreiben werden von Komponenten jedoch erst im Fehlerfall aufgerufen, wobei eine Komponente ihr gesamtes lokales Logbuch an den Dienst übermittelt.

Bei der Fehlersuche können unterschiedliche Protokollebenen verwendet werden. In gängigen Protokollierungsrahmenwerken (z.B. *log4j* [Apa13c], der *Java Logging API* [Ora13a] oder *Logback* [Qua13a]) finden sich hier beispielsweise Ebenen wie *Debug*, *Info*, *Warnung* und *Fehler*, welche unterschiedliche Protokollgranularitäten erlauben.

Auf diese Weise ist es möglich, eine effiziente Protokollierung zu realisieren, welche im normalen Betrieb lediglich geringe Leistungseinbußen mit sich bringt, im Bedarfs- bzw. Fehlerfall jedoch eine detaillierte Übersicht über das Systemverhalten erlaubt.

7.1.5.8. Leistungsüberwachung

Aufgabe der Leistungsüberwachung ist das Erheben leistungsrelevanter Eckdaten (engl. *Benchmarking*). Diese Daten haben im Wesentlichen zweierlei Nutzen: zum einen dienen sie der Auswahl wenig ausgelasteter Dienstinstanzen, um einen Lastausgleich im System zu ermöglichen, und zum anderen unterstützen sie die Detektion von Flaschenhälsen, d.h. stark ausgelasteten Dienstinstanzen bzw. Verarbeitungsknoten, die einen signifikant negativen Einfluss auf die Vermittlungsleistung haben, Lastspitzen oder allgemein der Verletzung von Quality of Service-Vereinbarungen. Auch das Accounting (vgl. Abschnitt 5.3.1.4) kann auf solchen Eckdaten aufsetzen. Sind beispielsweise einzelne Dienste entgeltpflichtig, so kann anhand von Aufrufprotokollen in Verbindung mit Vorgangsnummern und der Identitätsinformation der aufrufenden Entität ein entsprechender Abrechnungsposten erstellt werden.

Zur Leistungsüberwachung muss jede Komponente beim Aufruf von Methoden ihrer öffentlichen Schnittstelle entsprechende Kennzahlen aufzeichnen. Hierzu gehören der Beginn der Verarbeitung, der Zeitpunkt der Rückgabe des Kontrollflusses an den Klienten sowie bedingt durch die Asynchronität von Aufrufen aktiver Komponenten auch das Bereitstellen eines Ergebnisses. Anhand dieser drei Zeitstempel lässt sich auf die Verweildauer von Aufträgen innerhalb einer Komponente schließen. Unter Berücksichtigung der Historie solcher Eckdaten lassen sich auch Lastspitzen erkennen und mit Hilfe zusätzlicher Zählvariablen Flaschenhälse detektieren. Die Erhebung weiterer Leistungsindikatoren wie Speicherverbrauch und Datenübertragungsvolumen pro Komponente müssen auf Ebene der Ausführungsplattform erhoben werden, was nicht mehr Gegenstand dieser Arbeit ist.

Da die Erhebung und Aufzeichnung von Eckdaten, ähnlich wie bei der Protokollierung, einen Einfluss auf die Leistung selbst haben kann, bedarf es auch

hier entsprechender Strategien zur Minimierung des Überhangs. Angelehnt an die Lösung bei der Protokollierung sollen die Eckdaten lediglich eines bestimmten Zeitfensters lokal bei jeder Komponente vorgehalten werden. Durch eine Ergänzung der Schnittstelle mit einer entsprechenden Methode zum Abruf der Leistungsdaten, können feingranulare Daten bei Bedarf entfernt zugegriffen werden. Zusätzlich erfolgt zu bestimmten Zeitpunkten eine Zusammenfassung der Daten, welche an die Metadatenbereitstellung übermittelt wird. Diese Zusammenfassung unterstützt andere Komponenten bei der Auswahl wenig ausgelasteter Dienstinstanzen.

7.1.5.9. Zugriffskontrolle

Der Vermittlungsprozess, so wie er im Rahmen dieser Arbeit verstanden wird, ist ein offener Prozess, welcher beliebige Ressourcen und Dienste zur Optimierung der Leistungsfähigkeit und zur Erweiterung seiner Funktionalität einbinden kann. Demgegenüber steht die Notwendigkeit, den Zugriff auf Daten und Dienste auf jeweils nur bestimmte Klienten einschränken zu können. Eine sichere Zugriffskontrolle zu realisieren, gestaltet sich jedoch schwierig, insbesondere wenn keine Anforderungen an die Klienten hinsichtlich verfügbarer Ressourcen gestellt werden sollen. Sensoren beispielsweise können häufig keine sichere Authentifizierung vornehmen oder Nachrichteninhalte mit Hilfe kryptographischer Funktionen signieren.

Daher ist es in einem ersten Schritt ausreichend einen Vermittler lediglich dahingehend zu erweitern, dass dieser einfache Berechtigungsnachweise (engl. *Credentials*) für Klienten verwalten kann. Das Prinzip beruht darauf, dass eine vertrauenswürdige Instanz (z.B. der Administrator in einer Firma) einen entsprechenden Berechtigungsnachweis erzeugt, welcher Angaben zur Zugriffskontrolle enthält, zum Beispiel die Art des Merkmals, welches Klienten zur Authentifizierung vorlegen müssen, und für welche Dienste sie anschließend autorisiert sind. Dieser Nachweis muss beim Vermittler registriert werden, was mit einer entsprechenden Kennung quittiert wird. Klienten, die sich auf diese Berechtigung berufen möchten, müssen sowohl diese Kennung als auch ihr Authentifizierungsmerkmal in ihren Nachrichten an den Vermittler mitführen. Dieser überprüft vor Aufruf bestimmter Dienste bzw. dem Zugriff auf Daten, ob das übergebene Merkmal authentisch ist, und kann die Aktion ggf. verweigern.

```
1  Credential {
2      IssuerId=<Id>,
3      IssuerPassphrase=<String>,
4      ValidUntil=<DateTime>,
5      ValidForClient {
6        <AuthenticationFeature>,
7        [...]
8      },
9      ValidForService {
10       <ServiceId>,
```

```
11        [...]
12      },
13      FurtherCredentials {
14        <String>=<String>,
15        [...]
16      }
17  }
```

Listing 7.4: Aufbau eines Berechtigungsnachweises

Listing 7.4 zeigt den Aufbau eines solchen Berechtigungsnachweises. Der Nachweis enthält die Kennung ($IssuerId$) des Erzeugers (z.B. des Administrators) sowie ein von diesem selbst bestimmtes Passwort ($IssuerPassphrase$). Wird der Nachweis nicht explizit durch den Erzeuger unter Angabe des Passwortes invalidiert, so kann dies auch nach Ablauf eines Gültigkeitszeitraumes ($ValidUntil$) automatisch erfolgen. Um zu bestimmen, für welche Klienten der Nachweis gültig ist, muss eine Liste ($ValidForClient$) mit Authentifizierungsmerkmalen ($AuthenticationFeature$) angegeben werden. Ein Merkmal kann beispielsweise einfach die Kennung eines Klienten sein, ein selbstgewähltes Passwort oder ein öffentlicher kryptografischer Schlüssel. Bei Verwendung von Letzterem wird erwartet, dass der Klient alle Nachrichten mit seinem privaten Schlüssel signiert und ein Vermittler diese Signaturen mit Hilfe des öffentlichen Schlüssels überprüft, um die Authentizität festzustellen.

Weitere Angaben des Berechtigungsnachweises beziehen sich auf eine Liste mit Dienstkennungen ($ValidForService$), für welche der Nachweis gültig ist, sowie die optionale Angabe weiterer Berechtigungsnachweise ($FurtherCredentials$). Diese Name/Wert-Paare können beispielsweise vom Verarbeitungsdienst im Zuge einer Workflow-Ausführung angefordert werden und für die Autorisierung des Zugriffs auf externe Dienste, z.B. die firmeneigene Datenbank, verwendet werden.

Der vermittlerseitige Schutz von Daten vor dem Zugriff Unberechtigter ist prinzipiell schwierig, da die Daten von verschiedenen Diensten, die jeweils unter der Kontrolle unterschiedlicher Organisationen stehen können, verarbeitet werden. Unter der Annahme, dass nur vertrauenswürdige Organisationen an der Verarbeitung teilnehmen, gilt es, den Zugriff einerseits auf dem Übertragungsweg zwischen den Diensten abzusichern, und andererseits die Datenverwaltung um entsprechende Konzepte zur Zugriffskontrolle zu erweitern. Da die Datenverwaltung eines Vermittlers prinzipiell beliebige Datenhaltungssysteme unterstützen soll und nicht jedes konkrete System eine Zugriffskontrolle bietet, wird von einem entsprechenden Schutz im Rahmen dieser Arbeit abgesehen, zumal das Persistieren der Daten im Rahmen des Vermittlungsprozesses optional ist. Das sichere Persistieren von Daten kann jedoch durch Delegation während der Datenverarbeitung an ein externes, vertrauenswürdiges Datenhaltungssystem erfolgen, welches unter Verwendung des oben erwähnten Berechtigungsnachweises Zugriffskontrollen durchsetzen

kann. Auch können die Daten durch Einfügen kryptographischer Funktionen in den Verarbeitungs-Workflow verschlüsselt abgelegt werden.

Ähnliches gilt auch für den Schutz von Diensten. Die Kernkomponenten und hier vorgestellten Erweiterungen eines Vermittlers sind zwar im Wesentlichen von allen Klienten nutzbar, jedoch könnte der Zugriff auf bestimmte Dienste nur einer Gruppe von Klienten gewährt werden oder ein Betreiber der Infrastruktur könnte ein Entgelt für den Zugriff verlangen wollen. Der Aufruf des Verarbeitungsdienstes oder das Abrufen von Kontextdaten aus der Datenverwaltung sind Beispiele hierfür. Möchte ein Klient solche Dienste in Anspruch nehmen, muss dieser ein entsprechendes Authentifizierungsmerkmal vorlegen, welches dann von einem Zugriffskontrolldienst überprüft wird. Bei erfolgreicher Authentifizierung wird der Aufruf dann entsprechend über den Protokolldienst (z.B. für eine spätere Abrechnung) festgeschrieben.

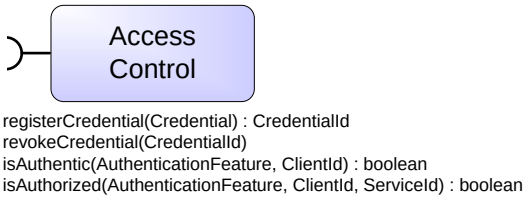


Abbildung 7.24.: Zugriffskontrolle: Angebotene Dienste

Die Zugriffskontrolle (engl. *Access Control*) ist, analog der Metadatenbereitstellung, zustandslos und somit mehrfach instanziierbar. Berechtigungsnachweise werden in einem verteilten Datenhaltungssystem hinterlegt und sind somit von jeder Instanz zugreifbar. Die Schnittstelle, dargestellt in Abbildung 7.24, bietet grundlegende Funktionen zum Registrieren und Invalidieren von Berechtigungsnachweisen sowie zur Authentifikation von Klienten und Autorisierung von Zugriffen.

7.1.6. Datenmodelle

Datenmodelle stellen eine formale Repräsentation der in einem System verwendeten Daten dar. Sie dienen nicht nur der Strukturierung der Daten durch Beschreibung von Objekttypen, deren Attribute und Beziehungen, sondern auch dem einheitlichen Austausch von Daten über Systemgrenzen hinweg und fördern die Wiederverwendbarkeit von Komponenten [FS06, Wis09, SLP04].

Insbesondere für den Bereich der Kontextdaten existieren eine Reihe von Modellierungsansätzen (vgl. [SLP04, CK00, HA07]). Zu unterscheiden sind hierbei die konzeptionellen Ansätze von den Implementationsmodellen [Wis09]. Erstere werden meist zur Dokumentation und strukturellen Analyse verwendet und dienen einerseits dem Verständnis durch den Menschen und andererseits als Basis zur Erstellung von Implementationsmodellen. Bekannte Vertreter dieser Klasse sind beispielsweise die *Unified Modeling Language* [Obj11b] und die *Context Modeling Language* [HI05] sowie verschiedene aktivitätstheoretische Ansätze (siehe [Wis09]).

Die Implementationsmodelle haben einen höheren formalen Grad als die konzeptionellen Modelle und können direkt in einem System eingesetzt und verarbeitet werden [Wis09]. Zu diesen Modellen gehören die *Key/Value-Modelle*, verschiedene Varianten sogenannter *Auszeichnungssprachen*, *objekt- und datenflussorientierte Modelle*, *relationale und logische Modelle* sowie *Ontologien* [SLP04, Wis09].

Die Wahl von Implementationsmodellen für die prototypische Realisierung eines Vermittlers basiert einerseits auf Evaluationen der Modelle hinsichtlich einer Reihe von Kriterien, die insbesondere in Ubiquitous Computing-Szenarien gelten [SLP04, Wis09], und andererseits auf den spezifischen Anforderungen des Vermittlungsprozesses. Zu Letzteren gehören unter anderem:

- performante De-/Serialisierung, da die Daten im Rahmen des Vermittlungsprozesses unter einer Vielzahl verteilter Komponenten ausgetauscht werden,
- Berücksichtigung ressourcenschwacher Produzenten und Konsumenten,
- Berücksichtigung der Erfordernisse insbesondere der Datenverwaltung und des Filterdienstes sowie
- Berücksichtigung existierender Standards des Sensor Webs und des Internet der Dinge.

Unter anderem um ressourcenschwache Produzenten und Konsumenten unterstützen zu können, wird zunächst ein einfaches Key/Value-Modell unterstützt. Klienten können ihre Sensor- oder Metadaten in Form einfacher Name/Wert-Paare an einen Vermittler übergeben. Da dieses Modell jedoch keine Unterstützung hierarchischer Daten (z.B. für Metadaten) bietet, können Produzenten im Rahmen des Registrierungsprozesses ein eigenes Key/Value-basiertes Schema ihrer Kontextdaten übermitteln. Dieses Schema enthält zum Beispiel die Datentypen der Sensordaten und wird unter anderem von der Datenverwaltung und vom Filterdienst zur Auswertung von Abfragen verwendet.

Um auch komplexe Datenstrukturen bei der Interaktion zwischen Klienten und Vermittler verwenden zu können, sind zudem einfache XML- und JSON-basierte Datenmodelle für alle Anfragen (z.B. Publikation von Sensordaten, Subskribieren von Abfragen etc.) definiert. Diese Modelle erlauben eine beliebige Schachtelung von Attributen sowie im Falle von XML auch das Hinzufügen von Metadaten zu den eigentlichen Werten. Auch die Datenmodelle des Open Geospatial Consortiums für das Sensor Web und von Global Standards One für das Internet der Dinge beruhen auf Auszeichnungssprachen. Zwar ist die durchgängige Verwendung der Standards erstrebenswert, allerdings würde man Leistung zugunsten von Interoperabilität opfern. Aus diesem Grund sollten die Standards am letztmöglichen Ende der Verarbeitungskette angewandt werden [FMF09]. Allerdings darf man nicht davon ausgehen, dass alle Produzenten und Konsumenten die Standards unterstützen. Aus diesem

Grund ist die Verwendung von Standards optional. Gateways und der High-level Verarbeitungsdienst bieten entsprechende Möglichkeiten zur Transformation des intern verwendeten Modells in die Auszeichnungssprachen bzw. Datenmodelle der Standards.

Intern verwendet ein Vermittler ein objektorientiertes Datenmodell. Dieses ist jedoch sehr einfach gehalten und erlaubt lediglich den Zugriff auf die konkreten Datenstrukturen. Bei Bedarf kann dieses Modell einfach in eine entsprechende Auszeichnungssprache überführt werden. Der interne Filterdienst (basierend auf Esper, siehe Abschnitt 7.2.1.2), welcher die komplexe Ereignisverarbeitung durchführt, unterstützt sowohl Key/Value-Modelle, objektorientierte und XML-basierte Modelle. Auch alle Dienste, die auf einer verteilten Datenbank beruhen (z.B. die Metadatenbereitstellung, die Workflow-Verwaltung und der Identitätsdienst), unterstützen diese drei Arten von Modellen.

7.2. Prototypische Realisierung eines Vermittlers

Im vorherigen Abschnitt wurde der Entwurf wesentlicher Kernkomponenten sowie zusätzlicher Erweiterungen eines Vermittlers mit den jeweiligen Aufgaben und Schnittstellen vorgestellt. Ausgehend von diesem technologieneutralen Entwurf sollen nun die wichtigsten Aspekte einer exemplarischen, prototypischen Realisierung eines Vermittlers erläutert werden, bevor im Anschluss in Kapitel 8 eine qualitative und quantitative Evaluation des Ansatzes präsentiert werden.

7.2.1. Verwendete Technologien

Zur Realisierung eines Vermittlers bedarf es der Auswahl konkreter Technologien zur Umsetzung von Kernfunktionalitäten, da diese direkten Einfluss auf die Architektur des Prototypen, wie sie in Abschnitt 7.2.2 beschrieben ist, haben. Hierzu gehören eine Ausführungsumgebung für aktive Komponenten, ein System zur Verarbeitung ereignisbasierter Abfragen, eine Beschreibungssprache zur Erstellung kontextbasierter Sichten und eine konkrete Technologie zur verteilten Datenhaltung.

7.2.1.1. Aktive Komponenten

In Abschnitt 7.1.1 wurden eine Reihe verbreiteter Softwareentwicklungsparadigmen hinsichtlich ihrer Eignung für den Entwurf und die Umsetzung eines Vermittlers miteinander verglichen. Das Paradigma der *aktiven Komponenten* [PB13] erwies sich hierbei aufgrund der gebotenen Abstraktionen als besonders gut geeignet für die Entwicklung verteilter Anwendungen in mobilen, ubiquitären Systemen mit hoher Dynamik.

Für die Realisierung der konstituierenden Bausteine eines Vermittlers existiert mit dem *Jadex Active Components Framework* [BP13] ein Rahmenwerk[5], welches den Entwickler hinsichtlich der wesentlichen Aspekte des Software-Lebenszyklus unterstützt. Hierzu gehören insbesondere die Implementierung und das Testen sowie das spätere Anpassen einer Anwendung im laufenden Betrieb an sich ändernde Umstände [Som12].

Jadex erlaubt die Entwicklung aktiver Komponenten mit unterschiedlichen internen Architekturen, sogenannten *Kernels*. Einfache Komponenten können beispielsweise als sogenannte *Micro-Agenten* realisiert werden, welche zwar Zugriff auf alle Funktionen der Laufzeitumgebung haben, jedoch nur wenig Ressourcen zur Ausführung bedürfen. Mit Hilfe von *BDI-Agenten* lassen sich „intelligente“ Komponenten entwickeln. Sie verfügen über eigene Schlussfolgerungsmechanismen, welche für die selbstbestimmte Auswahl von Plänen zum Erreichen definierter Ziele dienen. Interne Architekturen können jedoch auch durch BPMN- und GPMN[6]-basierte Workflows realisiert werden, mittels derer strukturierte Prozesse modelliert und durch die Jadex-Laufzeitumgebung direkt ausgeführt werden können.

Die Implementierung wird durch eine Reihe von Bibliotheken unterstützt, welche den zu entwickelnden Komponenten Zugriff auf wesentliche Funktionen der Jadex-Laufzeitumgebung erlauben. Hierzu gehören beispielsweise die Verwaltung des Lebenszyklus von Komponenten sowie das Anbieten und Auffinden benötigter Dienste. Alle Funktionen des Jadex-Rahmenwerkes, deren synchroner Aufruf den Aufrufenden potentiell über längere Zeit blockieren könnten (z.B. die Suche nach Diensten), werden asynchron ausgeführt. Hierzu wird dem Aufrufenden zunächst ein sogenanntes *Future* zurückgeliefert, dessen interner Zustand (zum Beispiel das Resultat der Ausführung oder eine Fehlermeldung) erst im späteren Verlauf gesetzt wird. Mittels einer Variante des Beobachter-Entwurfsmusters [GEG09] kann dieser schließlich informiert werden sobald das Future seinen internen Zustand ändert und unterdessen weitere Anweisungen ausführen. Da der Aufrufende nicht blockiert wird, können auf diese Weise Verklemmungen (engl. *Deadlocks*) konzeptionell verhindert werden [PB11].

Außerdem ermöglicht Jadex den verteilungstransparenten Zugriff auf entfernte Plattformen und Komponenten bzw. die von diesen angebotene Dienste. Die Laufzeitumgebung kann sich ihrer logischen Umwelt dahingehend gewahr werden, dass entfernte Plattformen automatisch aufgefunden werden. Mittels eines speziellen Relay-Servers kann dies sogar (durch Firewalls hindurch) über das Internet erfolgen. Sofern die Sicherheitsrichtlinien von Jadex entsprechend eingestellt sind, können so auch Dienste entfernter Plattformen aufgefunden und gebunden sowie die entfernten Plattformen mittels der gebotenen Werkzeuge administriert werden.

[5] Da Jadex das bisher einzige Rahmenwerk für aktive Komponenten ist, entfällt eine Evaluation von Alternativen.

[6] Goal-oriented Process Modeling Notation [JBP+11]

Jadex bietet zudem eine Reihe von Werkzeugen zur Modellierung aktiver Komponenten, zur Verwaltung von Anwendungen, zur Überwachung der Komponenten und ihrer Kommunikation, zum Auffinden entfernter Laufzeitumgebungen, zur Unterstützung beim Testen und bei der Fehlersuche etc. Die Werkzeugunterstützung und die bereitgestellten Bibliotheken des Rahmenwerkes bieten zusammen mit den Entwurfszielen, welche die Unterstützung verschiedener Kernel, die Kernel-Wiederverwendbarkeit, die Anwendungsportabilität und die Unterstützung heterogener Anwendungen umfassen [PB13], eine ausgezeichnete Basis zur Realisierung der Vermittlerfunktionalitäten.

7.2.1.2. Ereignisbasierte Abfragen auf Datenströmen

Produzenten publizieren kontinuierlich Sensordaten in Form eines Daten- oder Ereignisstroms, d.h. in einer partiellen Ordnung von Nachrichten. Konsumenten können Abfragen bei einem Vermittler subskribieren, um informiert zu werden, sobald in diesen Strömen für sie relevante Ereignisse oder Ereignismuster zu finden sind. Da diese Datenströme jedoch nicht zwangsläufig persistiert werden, das Datenvolumen sehr groß und das Auffinden komplexer Muster aufwendig sein kann und darüber hinaus die Reaktionszeit möglichst niedrig sein sollte, eignen sich herkömmliche Datenbanktechniken zur Datenabfrage nur bedingt [Luc06].

Für diesen Einsatzzweck haben sich Systeme zur Datenstromverarbeitung (engl. *Event Stream Processing*, ESP) und zur Verarbeitung komplexer Ereignisse (engl. *Complex Event Processing*, CEP) etabliert. Während beim ESP Abfragen kontinuierlich gegen eintreffende Daten eines Datenstroms ausgewertet werden, bezieht sich das CEP auf die (datenstromübergreifende) Detektion von komplexen Mustern in ungeordneten Strömen, sogenannten Ereigniswolken [Luc06], durch Filterung, Korrelation, Kontextualisierung und Analyse der Daten [Luc02, GRHY09].

Das grundsätzliche Vorgehen ähnelt dabei einer Art umgekrempelten Datenbank: die Abfragen werden in der Datenbank vorgehalten während die Daten gegen die Abfragen laufen. Eine Abfrage kann man sich dabei wie einen Zustandsautomaten vorstellen [Esp13b]. Knoten repräsentieren einen bestimmten Abfragezustand während die Kanten die Übergangsbedingungen (z.B. $Temperature = 23.5$) darstellen. Trifft ein passendes Ereignis ein, wird der Zustand des Automaten entsprechend fortgeschrieben. Ist eine Abfrage komplett erfüllt, d.h. der Automat in einem entsprechenden Endzustand, wird ein neues, sogenanntes komplexes Ereignis ausgelöst und der Subskribierte hierüber benachrichtigt.

In der Praxis existieren eine Reihe verschiedener (teils kommerzieller) ESP/CEP-Systeme. Bedeutende Vertreter kommen unter anderem von *SAP/-Sybase*, *TIBCO*, *IBM*, *Oracle* und *EsperTech*. In [GRHY09] werden eine Reihe von Systemen hinsichtlich verschiedener Kriterien wie zum Beispiel Laufzeitarchitektur, Plattformverwaltung, Funktionen zur Ereignisverarbeitung, Werkzeugunterstützung und Marktchancen etc. evaluiert. *Esper*, wenn

auch nicht so leistungsfähig wie kommerzielle Produkte, stellt das einzige frei verfügbare System in der Bewertung dar und wurde als *"Strong Performer"* eingeordnet, d.h. dem System wird aufgrund sehr guter Bewertungen hinsichtlich obiger Kriterien eine gute Entwicklung und Marktdurchdringung attestiert bzw. vorausgesagt. Ein weiterer Leistungsvergleich rein technischer Aspekte von CEP-Systemen, unter anderem *AMIT* [Hai13], *Drools Fusion* [Com13a] und Esper [Esp13a], in [GSS10] attestierte Esper eine sehr niedrige Latenz und einen hohen Durchsatz. Da es alle wesentlichen funktionalen Anforderungen hinsichtlich der Ereignisstromverarbeitung und der Detektion komplexer Ereignisse erfüllt, findet es in der prototypischen Realisierung eines Vermittlers Verwendung.

Genau wie Jadex stellt auch Esper ein Rahmenwerk zur Programmierung sowie eine Laufzeitumgebung zur Ausführung bereit. Den Kern bildet die sogenannte *Esper Engine*, welche für den Abgleich von Abfragen zu eintreffenden Ereignissen verantwortlich ist. Die Esper Engine bildet somit den Kern des Filterdienstes (vgl. Abschnitt 7.1.2). Hier werden eintreffende primitive Ereignisse von der Esper Engine gegen die subskribierten Abfragen ausgewertet. Zur Formulierung von Abfragen wird die Esper-eigene *Event Query Language* (EQL) bzw. *Event Pattern Language* (EPL) verwendet [Esp13b]. Bei erfolgreichem Abgleich generiert der Filterdienst ein neues komplexes Ereignis und übergibt dies einer hierfür neu erzeugten Instanz eines High-level Verarbeitungsdienstes (vgl. Abschnitt 7.1.2.9), welcher für die weitere Verarbeitung und die Benachrichtigung des Klienten verantwortlich ist.

7.2.1.3. Prozessmodellierung mittels BPMN

Bei der Vermittlung kontextbasierter Sichten müssen Prozesse auf unterschiedlichen Ebenen unterschieden werden. Hierbei sollen Prozesse nach Melzer [Mel10] als eine „Zusammenstellung von Aktivitäten zu Arbeitsabläufen" verstanden werden. Auf der obersten Ebene findet sich der Vermittlungsprozess an sich. Im Rahmen der Vermittlung werden eine ganze Reihe von Aktivitäten (z.B. das Verarbeiten, Filtern und Veröffentlichen von Kontextinformationen) ausgeführt, wobei die Koordination der Ausführung durch aktive Komponenten (vgl. Abschnitt 7.2.1.1) erfolgt und somit nicht explizit modelliert wird.

Die andere Art von Prozess betrifft speziell die Verarbeitung von Kontextdaten bzw. die Erstellung kontextbasierter Sichten. Dieser als *Verarbeitungsprozess* bezeichnete Vorgang stellt eine Aktivität im übergeordneten Vermittlungsprozess dar und beschreibt die einzelnen Arbeitsschritte, welche nach Maßgabe der Produzenten bzw. Konsumenten auf Kontextdaten ausgeführt werden sollen (vgl. Abschnitte 4.4.3 und 5.4.3). Damit ein Vermittler einen solchen Prozess ausführen kann, müssen Klienten ein Prozessmodell anfertigen und diesem übergeben, welches alle auszuführenden Verarbeitungsschritte sowie deren logische Reihenfolge vorgibt.

Ein solches Prozessmodell muss serialisierbar sein und eine Ausführungssemantik besitzen. Idealerweise ist es sowohl für Menschen als auch für Maschinen leicht verständlich und kann mit Hilfe geeigneter Werkzeuge erstellt werden. In der Praxis finden sich unterschiedliche Beschreibungssprachen, welche diese Anforderungen erfüllen. Zu den prominentesten Vertretern in der Praxis gehören die *Business Process Execution Language* (BPEL) [Org07] sowie die *Business Process Model and Notation 2.0* (BPMN) [Obj11a] (vgl. Abschnitt 5.4.3).

Im Rahmen dieser Arbeit findet BPMN Verwendung, da es insbesondere aus Sicht des Klienten eine abstraktere und somit einfachere Modellierung von Prozessen erlaubt. Zudem unterstützt das Jadex Active Components-Rahmenwerk (siehe Abschnitt 7.2.1.1) die Ausführung BPMN-basierter Prozesse, sodass eine zusätzliche *Workflow Engine* obsolet ist.

Die BPMN-Beschreibungssprache basiert auf wenigen grundlegenden Elementen mit jeweils einer begrenzten Anzahl an Ausprägungen [Obj11a]. Sogenannte *Pools* und *Swimlanes* repräsentieren Teilnehmer, Organisationen oder Rollen, welche einzelne Verarbeitungsschritte ausführen. Diese Aktivitäten sowie Kontrollflusselemente und Ereignisse werden als *Flow Objects* in den Swimlanes modelliert und mittels *Connecting Objects*, d.h. verschiedenen Arten von Kanten, miteinander verbunden, um den Kontrollfluss und Nachrichtenaustausch darzustellen. Zudem existieren noch *Artifacts*, welche Datenobjekte, Annotationen etc. repräsentieren können.

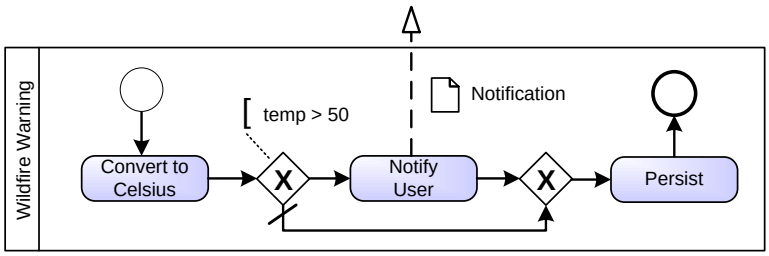


Abbildung 7.25.: BPMN-Beispiel: Waldbrandwarnung

Abbildung 7.25 zeigt ein einfaches BPMN-Beispiel eines Prozesses zur Waldbrand-Überwachung. Der leere Kreis links oben repräsentiert ein einfaches Startereignis. Hier beginnt der Kontrollfluss bei der Prozessausführung (und endet bei dem rechten Kreis mit dickerem Rand). Die drei abgerundeten Kästen stellen verschiedene Aktivitäten dar und die X-Rauten dienen der Steuerung des Kontrollflusses: in gegebenem Beispiel eine Exklusiv-Oder-Aufspaltung (bzw. -Zusammenführung) hinsichtlich eines Temperaturgrenzwertes. Alle diese Flowing Objects sind mittels durchgezogener Pfeile miteinander verbunden. Lediglich die gestrichelte Linie sowie das nebenstehende Dokument deuten an, dass hier ein Nachrichtenversand stattfindet.

Um einen solchen abstrakten Prozess ausführen zu können, bedarf es weiterer technischer Informationen. In BPMN lassen sich diese beispielsweise als Attribute der Elemente oder mittels Kommentaren annotieren (vgl. Abbildun-

gen 7.15 und 7.17). Auf diese Weise werden im Jadex Active Components-Rahmenwerk zum Beispiel Ein- und Ausgabeparameter der Aktivitäten sowie die Auswertungsbedingungen an den Kontrollflusselementen definiert.

Zudem unterstützt Jadex die Modellierung durch einen eigenen BPMN 2.0-kompatiblen Editor, welcher Anwendern das Erstellen von Verarbeitungs-Workflows wesentlich erleichtert (vgl. Abschnitt 7.2.2.4).

7.2.1.4. Nicht-relationale Datenbanken

Im Rahmen des Vermittlungsprozesses müssen unter Umständen große Mengen heterogener Daten vorgehalten werden. Hierzu gehören neben den Registrierungs- bzw. Subskriptionsinformationen von Produzenten bzw. Konsumenten z.B. auch Identitätsinformationen und Leistungskennzahlen von Komponenten, Aufruf- und Fehlerprotokolle sowie unter Umständen auch die Sensordaten der Produzenten.

Wie in Abschnitt 7.1.2 beschrieben, soll die horizontale Skalierbarkeit und Robustheit eines Vermittlers unter anderem dadurch realisiert werden, dass dessen Komponenten zustandslos und somit beliebig häufig auf unterschiedlichen Knoten instanziierbar sind. Somit kann die Last unter allen Instanzen verteilt werden und der Ausfall eines Knotens hat keine gravierenden Auswirkungen auf das Fehlerverhalten des Gesamtsystems.

Da die Komponenten jedoch zustandslos sind, müssen Informationen, die zur Laufzeit benötigt werden, aus einem Datenhaltungssystem geladen werden. Aufgrund der zu erwartenden hohen Last, darf dieses nicht zentralisiert sein, sondern muss verteilt realisiert werden. Zusätzlich ist aufgrund der hohen Heterogenität der Daten gefordert, dass die Datenhaltung zudem den Umgang mit semi-strukturierten Daten beherrscht, da die wenigsten Komponenten ein festes Schema für die zu persistierenden Daten vorsehen (können).

Im Zuge der Datenexplosion in der Web 2.0-Ära haben sich für ähnliche Anforderungen die sogenannten *NoSQL*[7]-Datenbanken etabliert. Unter diesem Oberbegriff werden eine Reihe von Datenbanken bzw. Datenbanktypen zusammengefasst, die im Vergleich zu den relationalen Datenbanken mit der standardisierten Abfragesprache *SQL* [GW10], gänzlich andere Aspekte in den Vordergrund stellen [Edl13, EBF+10, SF12]. Hierzu gehören beispielsweise ein nicht-relationales Datenmodell, weitestgehende Schemafreiheit, eine starke Ausrichtung auf Verteilung und horizontale Skalierbarkeit und damit einhergehend die Unterstützung von Datenreplikation. Statt des *ACID*-Konsistenzmodells[8] verwenden NoSQL-Datenbanken meist das alternative *BASE*-Modell[9], welches geringere Konsistenzanforderungen zugunsten der Verfügbarkeit hat und damit das *CAP*-Dilemma [EBF+10] (siehe Abschnitt 7.1.2.2) entschärft.

Aufgrund dieser Eigenschaften eignen sich Vertreter nicht-relationaler Datenbanken prinzipiell besser für die Realisierung oben angesprochener Da-

[7] *NoSQL* wird häufig einfach als Akronym für „Not only SQL“ verstanden [SF12]
[8] ACID steht für *Atomicity*, *Consistency*, *Isolation* und *Durability* [KE11]
[9] BASE steht für *Basically Available*, *Soft state*, *Eventual consistency* [EBF+10]

tenhaltung im Rahmen des Vermittlungsprozesses. Jedoch gibt es innerhalb der NoSQL-Familie eine Reihe von weiteren Unterteilungen, die im Folgenden kurz vorgestellt werden sollen [SF12, EBF+10]:

Key/Value Stores Key/Value-Datenbanken gehören aufgrund ihres Datenmodells zu den einfachsten NoSQL-Datenbanken. Klienten können zu einem Namen einen beliebigen (auch komplexen) Wert hinterlegen und diesen später wieder abfragen. Einige Datenbanken (z.B. *Redis* [SN13]) erlauben zudem mit Bereichseingrenzungen, Vereinigungen und Schnittmengenbildung auch komplexere Arten von Abfragen. Da Namen die Primärschlüssel repräsentieren und indiziert werden, ist der Zugriff sehr performant und die Daten lassen sich einfach über mehrere Knoten hinweg verteilen und somit horizontal skalieren.

Wide Column Stores oder Column Family Stores Während relationale Datenbanken die Daten meist zeilenweise speichern, werden diese bei spaltenorientierten Datenbanken physisch nach Spalten organisiert. Dies führt zwar zu langsameren Schreibvorgängen, in vielen Anwendungen wird jedoch häufiger lesend zugegriffen, wobei Lesevorgänge insbesondere nur wenige Spalten, aber viele Zeilen betreffen. Die Datenmodelle sehen daher meist einen eindeutigen Bezeichner als Primärschlüssel vor, welcher auf ein zweidimensionales Array von Schlüssel/Wert-Paaren (den Spalten) verweist. Da jeder Primärschlüssel auf ein eigenes Array verweist, ist es nicht notwendig, dass jede Zeile auch die gleichen Spalten definiert. Unter anderem des effizienteren Zugriffs wegen, ist es jedoch auch möglich Spalten, welche von vielen Zeilen genutzt werden, in Spaltenfamilien (engl. *Column Families*) zu gruppieren. Die Abfragemächtigkeit dieser spaltenorientierten Datenbanken ist jedoch ähnlich wie bei den Key/Value-Datenbanken eingeschränkt.

Document Stores Dokumenten-Datenbanken sind mit den Key/Value-Datenbanken vergleichbar: zu einem Schlüssel wird ein Wert gespeichert. Der wesentliche Unterschied besteht darin, dass hier der Wert ein selbstbeschreibendes, hierarchisch-strukturiertes Dokument (z.B. ein XML-, JSON- oder BSON-Dokument) darstellt, welches aus beliebigen Attributen und komplexen Datentypen bestehen kann. Dokumente unterschiedlicher Schlüssel müssen nicht dieselbe Struktur, Datentypen oder Bezeichner haben. Klienten können beliebige Dokumente speichern und es obliegt schließlich dem Abfragenden, die Struktur zu kennen bzw. zu interpretieren. Im Rahmen von Abfragen kann diese Struktur traversiert werden, wodurch z.B. alle Dokumente mit dem gleichen Wert eines bestimmten Attributs ermittelt werden können.

Graphdatenbanken Graphdatenbanken erlauben das Speichern vernetzter Informationen, d.h. von Entitäten und Beziehungen zwischen diesen mit jeweiligen Attributen. Die Stärke dieser Art von Datenbanken liegt insbe-

sondere in der Flexibilität der Abfragestellung und dem effizienten Traversieren des Graphen zur Beantwortung von Abfragen.

Die Wahl einer Datenbank hängt von vielerlei Faktoren ab [EBF+10]:

- Art, Umfang und Komplexität der Daten, das Daten- bzw. Speichermodell und ob und wie innerhalb der Daten navigiert werden muss,
- Transaktionsanforderungen und Isolationsebenen für Transaktionen,
- Leistungsaspekte (z.B. Latenzzeiten, vorhersagbares Antwortverhalten, Durchsatz) und Skalierbarkeitsanforderungen (Scale-up vs. Scale Out),
- Komplexität und Art der Abfragen, Analysemöglichkeiten für Daten,
- Architektur, Grad der Verteilung und Datenzugriffsmuster sowie
- weiterer nicht-funktionaler Anforderungen wie die Art der Replikation, Zugriffsschutz, Sicherungsstrategien, Unterstützung von Standards, Erweiterbarkeit, Quelloffenheit etc.

Cassandra Hinsichtlich der genannten Auswahlfaktoren bietet sich *Apache Cassandra* [Apa13a] mit der Erweiterung *Astyanax* [Lan13] für die Realisierung der verteilten Datenhaltung eines Vermittlers an. Ursprünglich von *Facebook* nach dem Vorbild von Googles *BigTable* entwickelt, wurde Cassandra 2008 als Open Source freigegeben und 2010 in die *Apache Foundation* aufgenommen. Cassandra gehört der Familie der Column Store-Datenbanken an, d.h. sie arbeitet spaltenorientiert, bietet jedoch zusätzlich eine relativ flexible Schemaunterstützung [EBF+10].

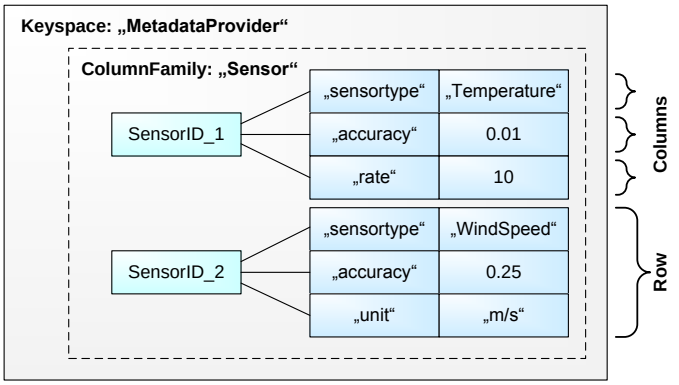


Abbildung 7.26.: Datenmodell von Cassandra

Das Datenmodell von Cassandra, dargestellt in Abbildung 7.26, verwendet als kleinste Einheit Name/Wert-Paare. Diese repräsentieren die Spalten (in der Abbildung z.B. *sensortype*, *accuracy* und *rate*) und können prinzipiell Daten beliebigen Typs enthalten, wobei für Name und Wert jedoch explizit auch Typen definiert und zur Laufzeit überprüft werden können.

Eine Zeile besteht aus einem beliebigen, aber eindeutigen Namen, dem sog. *Key* (in der Abbildung z.B. $SensorID\_1$ und $SensorID\_2$), dessen Wert eine Liste von Spalten ist. Dabei ist es unerheblich, ob eine Zeile nur wenige Spalten (sog. *Skinny Row*) oder viele, ggf. tausende Spalten (sog. *Wide Row*) besitzt. Zeilen mit annähernd ähnlichen Spalten können zu Spaltenfamilien (sog. *Column Families*) gruppiert werden. Hier wird bereits deutlich, dass der Zugriff zeilenbasiert über den eindeutigen Namen einer Zeile, aber auch spaltenbasiert erfolgen kann. In der Abbildung 7.26 haben beispielsweise beide Zeilen die Spalten $sensortype$ und $accuracy$, jedoch jeweils auch eine weitere andere Spalte. Dennoch kann man sie derselben Spaltenfamilie zuordnen. Zur Laufzeit können zudem beliebige Spalten hinzugefügt oder gelöscht werden.

Zusätzlich zu den normalen Spalten lassen sich zudem Super-Spalten (sog. *Super Columns*) definieren. Statt eines normalen Wertes referenziert eine Super-Spalte wiederum eine Menge an Spalten (ähnlich einer neuen Zeile). Auch Super-Spalten lassen sich zu Super-Spaltenfamilien zusammenfassen. Alle Spaltenfamilien gehören zudem einem Schlüsselraum, dem sog. *Keyspace* (in der Abbildung $MetadataProvider$) an. Dieser bildet die höchste Ebene der Struktur und kapselt die Daten unterschiedlicher Anwendungsbereiche. Mit Hilfe der vielen Möglichkeiten zur Strukturierung lässt sich ein flexibles Schema für den Datenraum erzeugen.

Abfragen werden typischerweise über die Netzwerkschnittstelle in einer proprietären Abfragesprache gestellt. Sie können Vergleichsoperatoren, Bereichsangaben und Zählfunktionen enthalten. Zudem existiert mit der *Cassandra Query Language* (CQL) [Apa13b] eine an SQL angelehnte Abfragesprache, die jedoch weitaus weniger mächtig ist.

Eine der wesentlichen Kerneigenschaften von Cassandra stellt das Peer-to-Peer-Modell dar [Tiw11]. Die meisten anderen NoSQL-Datenbanken folgen einem Master/Slave-Modell, bei dem Schreibzugriffe ausschließlich an den Master gerichtet sind und von diesem zu den Slaves propagiert werden. Auf diese Weise können zwar Lese-, nicht jedoch die Schreibzugriffe skalieren. Das Peer-to-Peer-Modell ermöglicht bei Cassandra eine einfache horizontale Skalierbarkeit beider Zugriffsarten. Zur weitergehenden Steigerung des Schreibdurchsatzes verwendet Cassandra zudem ein *In-Memory Commit Log*, welches sehr hohe Schreibraten erlaubt [Tiw11]. Unter anderem aus diesen Gründen eignet sich Cassandra daher insbesondere für Anwendungen mit höherem Schreib- als Leseaufkommen, wozu auch die Vermittlung kontextbasierter Sichten und insbesondere das Persistieren von Sensordaten gehören.

Die Knoten werden bei Cassandra in einem Ring angeordnet, wobei alle Knoten prinzipiell gleich sind. Es gibt keine zentrale Instanz zur Koordination der Knoten, dies geschieht untereinander mittels eines *Gossip*-Protokolls [Tiw11]. Soll das System erweitert werden, so kann dem Ring einfach ein neuer Knoten hinzugefügt werden. Nach definierbaren Strategien werden daraufhin die Schlüsselbereiche einzelner oder aller Knoten neu vergeben und die Daten entsprechend verschoben. Fällt ein Knoten aus, so können die Daten, die in den Schlüsselbereich des Knotens fallen, nicht mehr zugegriffen werden.

Um dies zu vermeiden, lassen sich Replikationsfaktoren angeben, welche die Anzahl Replikate bestimmen. Zur Wahrung der Konsistenz können für Lese- und für Schreibzugriffe unterschiedliche Konsistenzebenen bestimmt werden [EBF+10, SF12].

7.2.2. Jadex Event Stream Processing Architecture

Nachdem nun die wesentlichen Schlüsseltechnologien vorgestellt wurden, soll im Folgenden eine exemplarische, prototypische Realisierung eines Vermittlers beschrieben werden. Dem Entwurf aus Abschnitt 7.1 folgend, wurde eine Middleware realisiert, welche die Vermittlung kontextbasierter Sichten zwischen Produzenten und Konsumenten in mobilen, ubiquitären Systemen ermöglicht. Die *Jadex Event Stream Processing Architecture* (JESPA) [Bad09a, Bad09b, BL09, BL12] realisiert die wesentlichen Aufgaben eines Vermittlungsprozesses und wurde im Kontext verschiedener Projekte (z.B. [BIK09a, IBK09, KFZB11]) eingesetzt. Im Folgenden sollen die konstituierenden Komponenten der Architektur, deren Zusammenspiel sowie weitere Aspekte der Implementierung im Detail erläutert werden.

7.2.2.1. Komponenten

Eine exemplarische, prototypische Implementierung erfordert zur Vermittlung kontextbasierter Sichten die Realisierung der wesentlichen Kernkomponenten aus Abschnitt 7.1.2 und muss darüber hinaus so angelegt sein, dass mögliche Erweiterungen (z.B. aus Abschnitt 7.1.5) einfach zu integrieren sind. Deshalb besteht JESPA aus einer Reihe stark kohäsiver, lose gekoppelter Komponenten, welche mittels des Jadex-Rahmenwerkes als aktive Komponenten, basierend auf dem Jadex Micro-Kernel, realisiert wurden. Jede dieser Komponenten implementiert Methoden zur Verwaltung ihres Lebenszyklus und deklariert explizit angebotene und benötigte Dienste. Die von einer Komponente angebotenen Dienste wurden in Form einer speziellen Dienstklasse realisiert, wobei generische Methoden (z.B. zum Suchen nach Diensten sowie für die Protokollierung und Leistungsüberwachung) von einem zugrundeliegenden Basisdienst geerbt werden, um Wiederverwendbarkeit des Codes zu gewährleisten.

In Abbildung 7.27 ist der grobe Aufbau einer Komponente anhand eines Beispiels dargestellt. Der $RegistrationAgent$ ist eine aktive Komponente, welche mit den Methoden $onCreate$, $executeBody$ und $agentKilled$ ihr Verhalten (z.B. das Initialisieren bzw. Finalisieren der angebotenen Dienste und des Datenhaltungssystems) in bestimmten Phasen des Lebenszyklus definiert. Mit dieser Komponente ist eine Dienstklasse $RegistrationService$ assoziiert, welche die Anwendungslogik für die Diensterbringung enthält. Diese Klasse stellt eine Spezialisierung des generischen $BaseService$ dar, welcher allgemein relevante Logik zur Diensterbringung kapselt. Der BaseService implementiert das Interface $IBaseService$, welches die öffentlichen Methoden dieses Basisdienstes explizit definiert. Analog hierzu implementiert der RegistrationService das Interface $IRegistration$, welches eine Spezialisierung des IBaseService-Interfaces

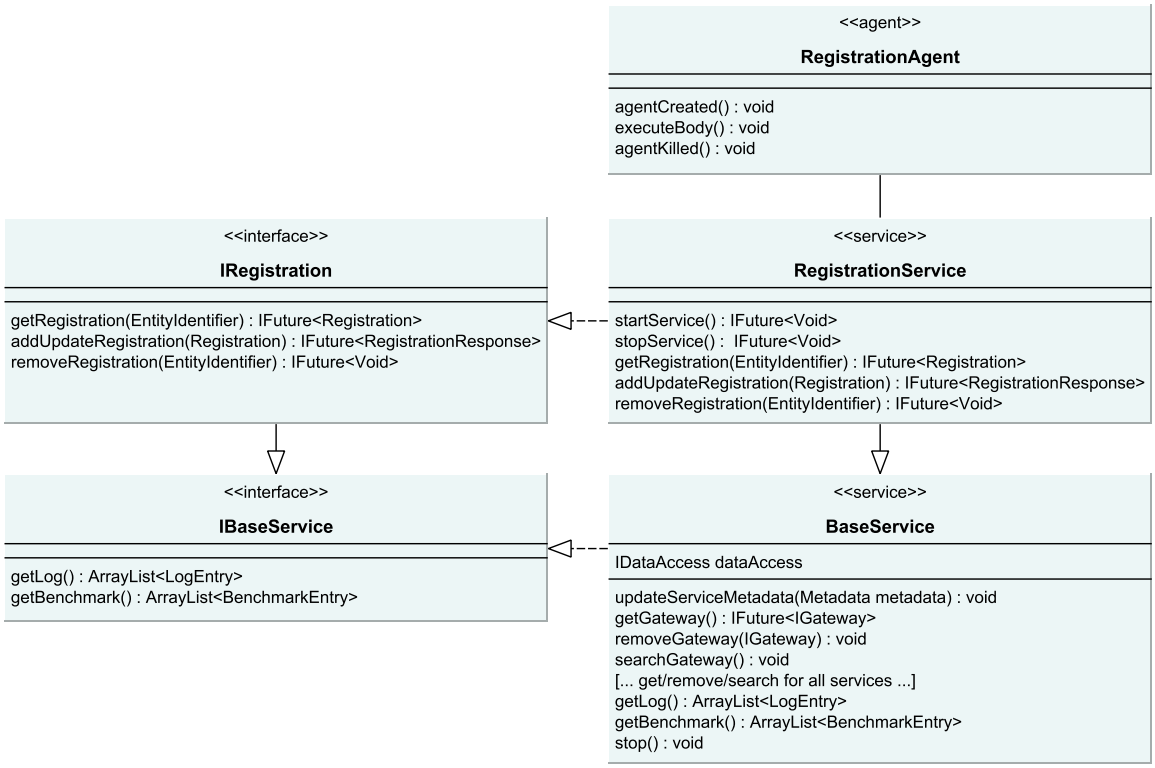


Abbildung 7.27.: Aufbau von Komponenten eines Vermittlers

darstellt und von Klienten für die Dienstsuche und das asynchrone Aufrufen der Dienstmethoden verwendet wird.

Alle Kernkomponenten (vgl. Abschnitt 7.1.2) eines Vermittlers können auf diese Weise umgesetzt werden. Das wurde exemplarisch mit dem Gateway, dem Identitätsdienst, der Metadatenbereitstellung, der Registrierung und dem Subskriptionsdienst, der Workflow-Verwaltung und dem Filterdienst gezeigt. Lediglich der Low-level- und der High-level-Verarbeitungsdienst folgen nicht diesem Aufbau, da sie keine öffentlichen Methoden anbieten, sondern individuell für die Verarbeitung eines Ereignisses erzeugt werden.

7.2.2.2. Zusammenspiel der Komponenten

Da eine starke Kohäsion der Komponenten gefordert ist, realisiert jede Komponente nur eine begrenzte Funktionalität und delegiert Aufgaben, die nicht in ihren Verantwortungsbereich fallen, an andere Komponenten. Die Registrierung beispielsweise nimmt die Registrierungsdaten eines Produzenten entgegen, prüft diese, erzeugt eindeutige Kennungen etc. Das Persistieren der Metadaten sowie des Workflows werden hingegen an die Metadatenbereitstellung bzw. die Workflow-Verwaltung delegiert.

Die Delegation erfolgt über den Aufruf von Diensten. Bei Aufruf lokaler Dienste werden Fachobjekte aus Gründen höherer Leistung als Referenzen, bei entfernten Diensten in serialisierter Form als XML-Repräsentation oder

in binärer Darstellung übergeben. Aufrufe sind netzwerktransparent, d.h. der Aufrufende ist sich nicht gewahr, ob der Empfänger lokal oder entfernt ausgeführt wird. Da ein Aufruf über das Netzwerk, bedingt durch die Nachrichtenlaufzeiten, eine erheblich höhere Latenz hat, werden sogenannte *Futures* verwendet (vgl. Abschnitte 7.1.1 und 7.2.1.1). Sie erlauben einen asynchronen Aufruf, ohne den Aufrufenden zu blockieren. Das bedeutet, der Aufrufende kann in der Zeit bis zur Bearbeitung seines Auftrags selbst weitere Aufträge annehmen und verarbeiten. Abbildung 7.28 veranschaulicht die Interaktion von Komponenten am Beispiel einer Subskription. Ein Konsument versendet eine Subskriptionsanfrage an ein Gateway, welches lediglich eine Durchleitungsfunktion hin zum Subskriptionsdienst übernimmt. Dieser generiert eine Reihe von Subskriptionskennungen, welche in einem Datenhaltungssystem hinterlegt werden, und extrahiert die wesentlichen Bestandteile der Abfrage: die Abfrage an den Filterdienst, die Identität des Klienten bzw. des Endpunktes, welcher die Ergebnisse empfangen soll, sowie den Workflow zur Erstellung kontextbasierter Sichten. Diese Informationen werden an andere Dienste delegiert und von diesen weitergehend verarbeitet und gespeichert. Die Bestätigung des Vorgangs bzw. Informationen über etwaige Fehler (z.B. Syntaxfehler in der ereignisbasierten Abfrage) werden schließlich über das Gateway zurück an den Konsumenten geleitet.

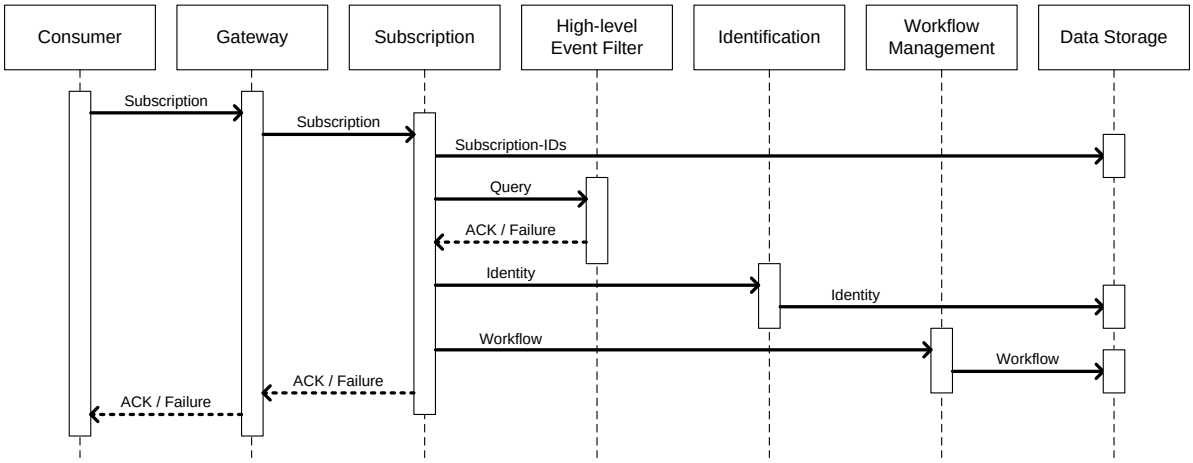


Abbildung 7.28.: Interaktion der Komponenten am Beispiel einer Subskription

Um einen passenden Dienst zur Delegation von Aufgaben aufzufinden, gibt es zwei verschiedene Möglichkeiten: Die Jadex-Laufzeitumgebung bietet von sich aus unterschiedliche Varianten zum Auffinden von Diensten aktiver Komponenten: Hierzu gehören unter anderem die Suche auf der lokalen Plattform, die Suche per Multicast im Netzwerk sowie die Suche über einen speziellen Relay-Dienst im Internet. Externe Dienste, die von Klienten angeboten werden und nicht über den Relay-Dienst zugreifbar sind, sowie Dienste, die nicht als aktive Komponenten realisiert wurden (z.B. Web Services), können über die

Metadatenbereitstellung unter Angabe bestimmter Auswahlkriterien (Diensttyp, Auslastung etc.) aufgefunden werden.

Die Suche nach Diensten sowie die konkrete Auswahl einer Dienstinstanz wird von oben erwähntem BaseService unterstützt. Dieser verwaltet einen für alle Komponenten einer Plattform gemeinsam genutzten Zwischenspeicher in dem alle aufgefundenen Dienste bzw. deren Metadaten und Endpunktreferenzen für den schnellen Zugriff vorgehalten werden. Kann eine Abfrage nicht aus dem Zwischenspeicher bedient werden oder ist ein Dienstendpunkt nicht mehr erreichbar, delegiert der Dienst die Suche an die Jadex-Laufzeitumgebung bzw. die Metadatenbereitstellung. Auf diese Weise sind die Komponenten lediglich lose miteinander gekoppelt, da erst zur Laufzeit ein konkreter Endpunkt für die Ausführung ermittelt wird. Dadurch wird auch ermöglicht, dass zur Laufzeit beliebig neue Instanzen für die Verarbeitung integriert oder vorhandene Instanzen entfernt werden können.

Insbesondere interessant ist dieser Aspekt in Bezug auf skalierbare Infrastrukturen. Das *Cloud-Paradigma* steht für ein verteiltes System aus virtualisierten Verarbeitungsressourcen wie Speicher, Prozessoren und Diensten, welche dynamisch vorgehalten und einem Klienten als eine vereinigte Ressource nach Bedarf zur Verfügung gestellt werden können [BBG10] (vgl. Abschnitt 5.4.6). Das Besondere hierbei ist, dass der Klient nur für die tatsächlich in Anspruch genommenen Ressourcen zahlt und diese zur Laufzeit beliebig hinzufügen und wieder freigeben kann. Auf diese Weise kann ein verteiltes System horizontal skaliert werden. Für einen Vermittler könnten beispielsweise bei hoher Auslastung weitere virtuelle Computer gestartet werden. Anhand der Metadaten der aktuellen Dienste kann dann festgestellt werden, welche Dienste besonders stark ausgelastet sind und weitere Instanzen dieser Dienste auf den neuen Computern angeboten werden. Durch das Umweltgewahrsein ist sichergestellt, dass diese Dienste automatisch aufgefunden und fortan in den Vermittlungsprozess integriert werden können, um punktuelle Lastspitzen auszugleichen.

7.2.2.3. Realisierung der Datenhaltung

Da ein Großteil der Komponenten zustandslos ist und somit der Skalierbarkeit halber beliebig häufig im Gesamtsystem instanziiert werden kann, bedarf es einer gemeinsam genutzten Datenbasis, damit jede Komponente Zugriff auf die von ihr benötigten und verwalteten Informationen (z.B. Registrierungsinformationen oder Metadaten) hat. Die Wahl einer konkreten Technologie hängt dabei von den jeweiligen Anforderungen der Anwendung ab und es wurde gefordert, dass prinzipiell verschiedene Datenhaltungssysteme eingesetzt werden sollen (vgl. Abschnitt 5.3.1.3). Im Rahmen dieser Arbeit wurden zwei verschiedene Optionen realisiert: eine leichtgewichtige Implementierung auf Basis von Hash-Tabellen und eine schwergewichtige Variante auf Basis von Apache Cassandra (vgl. Abschnitt 7.2.1.4).

Die Implementierung auf Basis von Hash-Tabellen dient als eine schnelle In-Memory-Lösung, d.h. sie persistiert die Daten nicht und unterstützt lediglich den lokalen Zugriff. Ihr Einsatz eignet sich, sofern ein Vermittler kontextbasierte Sichten nur zwischen lokalen Produzenten und Konsumenten vermitteln muss, sofern dieselbe Art von Komponente mehrfach auf einem Knoten instanziiert werden soll (zum Beispiel um mehrere CPU-Kerne auszunutzen) oder wenn dieselbe Komponente nur einmal im Gesamtsystem existiert.

Für die Datenhaltung mittels Apache Cassandra kommt eine angepasste Version eines eingebetteten Cassandra-Servers [Hec13] zum Einsatz. Dieser wird gestartet sobald die erste Komponente auf das Datenhaltungssystem zugreift. Das *Singleton-Entwurfsmuster* [GEG09] verhindert, dass jede Komponente einer Plattform eine eigene Instanz startet und stattdessen alle lokalen Komponenten dieselbe Server-Instanz verwenden. Für die Interaktion mit Cassandra wurde das *Astyanax*-Rahmenwerk [Lan13] verwendet, welches von den grundlegenden Methoden der Cassandra API abstrahiert und eine einfache Integration in bestehende Anwendungen erlaubt.

In Cassandra bekommt jede Art von Komponente einen eigenen Keyspace zugewiesen, sodass die Daten logisch von den Daten der anderen Komponenten getrennt werden. Eine Unterteilung der Keyspaces in verschiedene Column Families erlaubt zudem eine weitere Trennung der fachlichen Informationen auch innerhalb der Komponenten. So definiert zum Beispiel die Metadatenbereitstellung unterschiedliche Column Families für allgemeine Metadaten und Ereignistyp-Abbildungen. Letztere werden bspw. für die Deserialisierung von Abfrageergebnissen verwendet, da Cassandra intern Byte-Arrays zur Speicherung nutzt und zur Laufzeit keine Informationen über Datentypen bietet.

Beide Varianten der Datenhaltung implementieren ein gemeinsames Interface, welches die konkrete Realisierung vor den Komponenten versteckt und einen transparenten Zugriff ermöglicht. Auf diese Weise ist es nicht nur möglich, weitere Datenhaltungsvarianten in einen Vermittler zu integrieren, sondern auch komponentenweise unterschiedliche Varianten gleichzeitig einzusetzen.

7.2.2.4. Erstellung kontextbasierter Sichten

Eine kontextbasierte Sicht, wie sie in Abschnitt 5.1 definiert wurde, enthält alle für einen bestimmten Anwendungszweck relevanten Informationen in einer geeigneten Repräsentation. Ein Konsument, der eine solche Sicht als Ergebnis einer Abfrage zugestellt bekommt, kann diese im Idealfall ohne weitere Aufbereitung für den eigentlichen Zweck weiterverwenden. Hiervon profitieren insbesondere ressourcenbeschränkte Geräte, da die Verarbeitungslast auf leistungsfähigere Knoten in der Infrastruktur verschoben werden kann. Außerdem können auf diese Weise auch eingebettete Systeme (z.B. programmierbare Fernseher oder allg. intelligente Haushaltsgegenstände) einfach in übergeordnete Prozesse integriert werden, sofern sie über eine Kommunikationsschnittstelle Anweisungen empfangen können.

Zur Erstellung einer kontextbasierten Sicht bedarf es der Spezifikation von Verarbeitungsschritten, welche beschreiben, wie aus primitiven Kontextdaten eine solche Sicht zu erzeugen ist. Seien beispielsweise die aktuelle Zeit, ein Terminkalender sowie der Aufenthaltsort einer Person der Eingabekontext. Die gewünschte kontextbasierte Sicht wäre zum Beispiel eine Routenbeschreibung samt Stauinformationen, welche der Person kurz vor Abfahrtszeitpunkt präsentiert werden soll, sodass der Termin rechtzeitig wahrgenommen werden kann. Die Verarbeitungsschritte geben vor, welche zusätzlichen Informationen zu den vorliegenden Kontextdaten abgerufen und wie all diese Informationen zur Erstellung der Sicht verarbeitet werden müssen.

Die Verarbeitungsschritte werden in JESPA mittels BPMN-basierter Verarbeitungsprozesse spezifiziert, welche eine Sequenz von Aktivitäten und Kontrollflussstrukturen beschreiben und von der Jadex-Laufzeitumgebung ausgeführt werden können [PB13]. Hierfür bietet Jadex ein grafisches Modellierungswerkzeug, welches bei der Erstellung der Prozesse unterstützt (dargestellt in Abbildung 7.29).

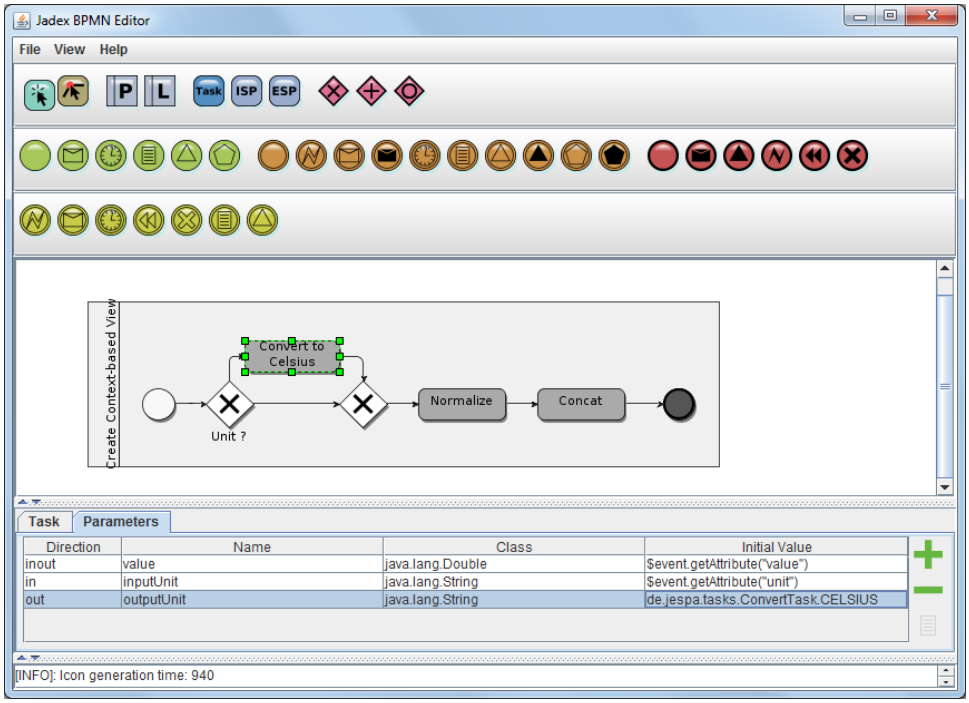


Abbildung 7.29.: Screenshot des Jadex BPMN-Editors

JESPA bietet eine Reihe typischer, parametrisierbarer Aktivitäten, welche ein Klient in seinen Prozessen verwenden kann. Über die Metadatenbereitstellung können Informationen, z.B. eine Beschreibung notwendiger Parameter, abgefragt werden. Zudem können beliebige eigene Aktivitäten mittels externer Dienstreferenzen eingebunden werden. So könnten beispielsweise Stauinformationen über den Web Service eines externen Anbieters geholt werden.

Der resultierende Prozess, auch als *Workflow* bezeichnet, kann von Produzenten im Rahmen der Registrierung sowie von Konsumenten bei

der Subskription an JESPA übermittelt werden und wird von der Workflow-Verwaltung persistiert. Die Produzenten-Workflows werden bei jedem übermittelten Sensorwert des jeweiligen Produzenten ausgeführt, die Konsumenten-Workflows werden entweder bei Abfrage historischer Zeitreihen oder der Detektion komplexer Ereignisse ausgeführt. Hierzu werden die serialisierten Workflows von der Workflow-Verwaltung abgerufen und als neue Jadex-Komponenten instanziiert. Die Kontextdaten, auf denen der Workflow operiert, werden dabei als Parameter an den Workflow übergeben. Ein Interpreter sorgt schließlich für die schrittweise Abarbeitung des Workflows und ruft die entsprechenden Aktivitäten zur Erstellung der kontextbasierten Sicht dem spezifizierten Kontrollfluss folgend auf.

7.2.2.5. Unterstützung heterogener Geräte

Eine typische Eigenschaft mobiler, ubiquitärer Systeme ist insbesondere die Heterogenität der Geräte, die im Rahmen des Vermittlungsprozesses als Produzenten, Konsumenten oder als Bestandteile eines Vermittlers auftreten. Hierbei spielt vor allem die unterschiedliche Ausstattung an Ressourcen eine wesentliche Rolle, die es bei der Realisierung eines Vermittlers zu berücksichtigen gilt.

An Produzenten und Konsumenten sollen dabei geringstmögliche Anforderungen gestellt werden. Da die gesamte Interaktion zwischen Klienten und Vermittler über ein Gateway (bzw. Proxy) verläuft, bedarf es an dieser Stelle der Unterstützung einerseits unterschiedlicher Kommunikationstechnologien und -protokolle sowie andererseits verschiedener Enkodierungen von Nachrichten. Für das Gateway wurden daher Adapter realisiert, die neben der Jadex-spezifischen Protokolle und Enkodierungen auch XML- bzw. JSON-kodierte Nachrichten interpretieren und transformieren und diese über SOAP- bzw. RESTful Web Services, TCP- und UDP Sockets sowie Bluetooth empfangen bzw. versenden können[10].

Um eine technologieneutrale Adressierung zu ermöglichen, besitzen alle Teilnehmer logische Adressen in Form von URIs. Bei Produzenten und Konsumenten besteht diese aus der eindeutigen Kennung, die ihnen bei der Subskription oder Registrierung zugeteilt wurde. Auch einzelne Dienstinstanzen lassen sich auf diese Weise adressieren, da Dienste jedoch zustandslos sind und es somit unerheblich ist, welche Instanz eine bestimmte Nachricht erhält, lassen sich Dienste darüber hinaus mittels ihrer Diensttypenkennung adressieren. Die Umsetzung von logischen auf physische Adressen geschieht durch die Gateways, welche beim Identitätsdienst die zuletzt bekannte physische zu gegebener logischer Adresse abfragen. Durch diese Art der Adressierung und Adressumsetzung ist es einem Vermittler möglich, mit einer Vielzahl heterogener Produzenten und Konsumenten zu interagieren.

[10] Weitere Adapter, z.B. zur Kommunikation über Web Services, Cloud-basiertes Push Messaging, SMS, ZigBee, gesprochene Sprache etc. sind möglich, wurden jedoch nicht realisiert.

Sofern Geräte ihre Ressourcen für den Vermittlungsprozess selbst zur Verfügung stellen sollen, z.B. indem sie einen bestimmten JESPA-Dienst ausführen, wachsen die Anforderungen an die Geräte. Insbesondere muss die Jadex-Laufzeitumgebung auf den Geräten ausführbar sein, da JESPA für die Kommunikation der Dienste untereinander auf Jadex aufsetzt. Als Beweis der prinzipiellen Ausführbarkeit von JESPA auf ressourcenbeschränkten Geräten wurde im Rahmen dieser Arbeit bereits 2010 zunächst eine Portierung von Jadex für die *Android*-Plattform [Ope13] vorgenommen [Bad10b]. Im Zuge dessen wurde auch eine Portierung einer eingeschränkten Version der Esper ESP/CEP-Laufzeitumgebung zur Verarbeitung komplexer Ereignisse durchgeführt [Bad10a]. Mit lediglich geringfügigen Anpassungen wurde schließlich auch der gesamte Vermittlungsprozess, der durch JESPA realisiert wird, auf Android-basierten Endgeräten zur Ausführung gebracht [Bad10c].

Die Jadex-Portierung wurde durch eine Implementierung von [Kal12] abgelöst, welche inzwischen als offizielle Jadex-Distribution verfügbar ist [BP13]. Da es jedoch nicht möglich war, die Android-Version der Esper-Laufzeitumgebung in den Hauptentwicklungszweig von Esper zu integrieren und eine kontinuierliche Portierung aktualisierter Esper-Versionen sehr aufwändig wäre, blieb es bei einem prototypischen Demonstrator für die Android-Plattform. Das bedeutet, es ist zum aktuellen Zeitpunkt möglich, JESPA-Dienste (mit Ausnahme des Filterdienstes auf Basis von Esper) auch unter Android auszuführen und in den Vermittlungsprozess einzubinden. Konsumenten und Produzenten können ungeachtet dessen weiterhin als Klienten eine Vermittlung kontextbasierter Sichten nutzen, sofern sie oben erwähnte Kommunikationstechnologien, Protokolle und Kodierungen unterstützen.

7.2.2.6. Werkzeugunterstützung

Um einerseits die Entwicklung und andererseits die Nutzung von JESPA angemessen zu unterstützen, wurden im Rahmen dieser Arbeit eine Reihe von Werkzeugen entwickelt. Diese dienen im Wesentlichen der Testunterstützung und der Leistungskontrolle, können jedoch auch als Basis für weitere Werkzeuge, z.B. zur Unterstützung der Abfrageformulierung oder der Administration eines Vermittlers, dienen.

Ereignisgeneratoren Ereignisgeneratoren dienen im Wesentlichen zwei Zwecken: i) Sie können einfache Last in Form von Ereignissen bzw. Sensordaten erzeugen, mit Hilfe derer eine Leistungsüberwachung und Optimierung eines Vermittlers durchgeführt werden kann und ii) sie erlauben die Simulation spezifischer Ereignisse zum Überprüfen der Semantik von Abfragen. Letzteres ermöglicht zum Beispiel die Simulation eines Wohnungsbrands oder Unwetters, für welche ein Konsument eine ereignisbasierte Abfrage subskribiert hat, und die Kontrolle, ob die Abfrage semantisch korrekt gestellt wurde.

Zur Leistungskontrolle und Optimierung wurde im Rahmen dieser Arbeit der *DejaVecuPlayer* als eine Erweiterung des *Replayers* von Kunz, Müller, Fa-

bian und Ziekow von der Humboldt-Universität zu Berlin implementiert. Der DejaVecuPlayer liest eine Datei mit Ereignissen und Zeitstempeln ein und erlaubt das zeitgesteuert wiederholte Abspielen der Ereignisse zu immer denselben relativen Zeitpunkten. Der DejaVecuPlayer kam im Rahmen des *AEPIO*/-*SOLA*-Projekts (siehe Abschnitt 8.2) zum Einsatz und erlaubte das Testen und die Evaluation des Projektes mittels unterschiedlicher Szenarien im Bereich des Internet der Dinge.

Einen ähnlichen Ansatz verfolgt *HALASS*, ein grafischer Generator für RFID-Leseereignisse, der von [Sch09] entwickelt und im Rahmen dieser Arbeit leicht angepasst wurde (vgl. Abbildung 7.30). Mit Hilfe dieses Generators ist es möglich, das Ein- und Austreten von RFID Tags in den Einzugsbereich entsprechender Lesegeräte zu modellieren und auf diese Weise komplexe Szenarien wie den Einkauf in Supermärkten oder Warenströme in einer Lagerhalle zu erstellen. Die Ereignisse, die ein Lesegerät beim Ein- und Austritt eines RFID Tags erzeugt, können mit Zeitstempeln persistiert und später mittels des DejaVecuPlayers wieder abgespielt werden.

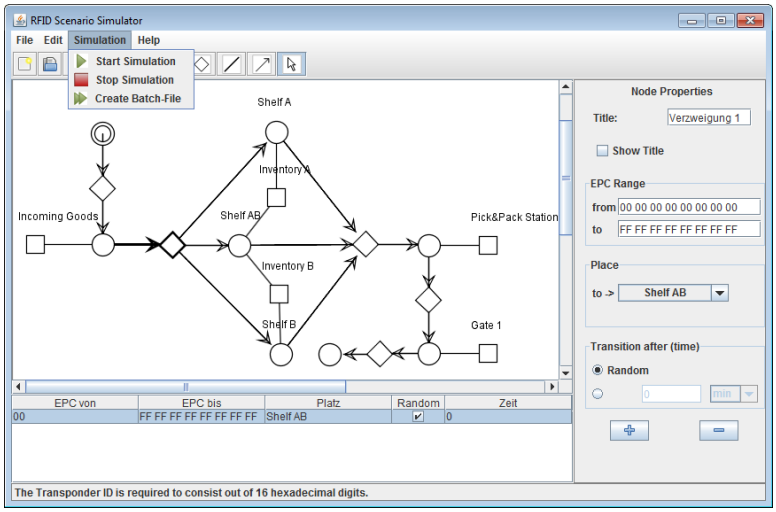


Abbildung 7.30.: Screenshot des HALASS-Ereignisgenerators

Zum Testen der JESPA Middleware mit Daten einer realen Umgebung wurden zudem Adapter für einen Quadrokopter sowie kleine Roboter implementiert. Der AR.Drone 2.0-Quadrokopter [Par12] ist ein Fluggerät, welches über verschiedene Sensoren (Beschleunigung, Gyroskop, Kamera etc.) verfügt, und aus der Ferne über eine WLAN-Funkverbindung kontrolliert werden kann. Das im Kontext dieser Arbeit entwickelte *YADrone*-Rahmenwerk [Bad12] für Desktop-Computer und Android-basierte Geräte (basierend auf dem *ARDroneForP5*-Rahmenwerk [Yos13]) bietet die Möglichkeit, den Quadrokopter als Produzent bei einer JESPA-Instanz zu registrieren, um anschließend kontinuierlich Sensordaten von diesem an den Vermittler zu senden. Abbildung 7.31 zeigt einen Screenshot des *YADrone Control Centers* zur Steuerung des Quadrokopters und zur visuellen Kontrolle der übermittelten Informationen durch den Benutzer.

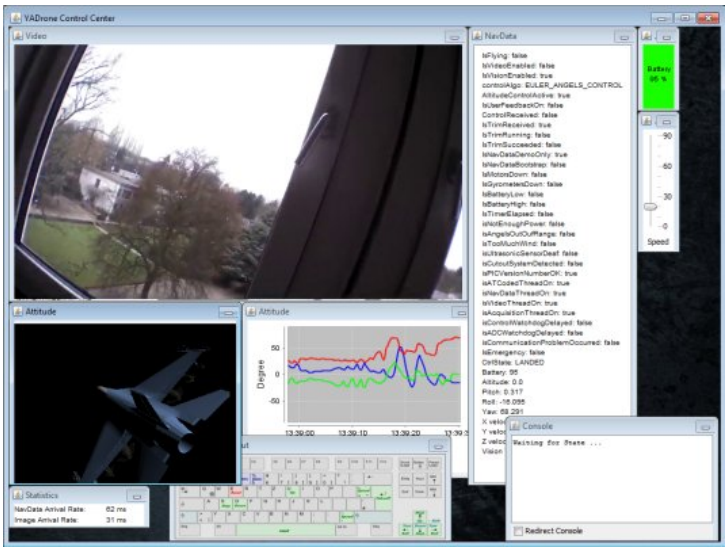


Abbildung 7.31.: Screenshot des YADrone Control Centers

Auch für *LEGO Mindstorm*-Roboter [Wik13a] wurde eine entsprechende Anbindung an JESPA realisiert. Diese Roboter verfügen über einen kleinen Baustein zur Ausführung primitiver Java-Programme mit Hilfe dessen Sensoren abgefragt und Aktuatoren gesteuert werden können. Im Rahmen dieser Arbeit wurde ein Sensor-/Aktuator-Proxy entwickelt, der auf dem Baustein der Roboter ausgeführt wird und per Bluetooth Sensordaten überträgt und Steuerbefehle entgegennimmt. Auf der Gegenseite abstrahiert eine aktive Komponente von einem konkreten Roboter und bietet Dienstschnittstellen zur Abfrage der Sensordaten und zur Steuerung des Roboters. Mit dieser Komponente wurde ebenfalls ein *Control Center* (dargestellt in Abbildung 7.32) sowie die Anbindung als Produzent an die JESPA Middleware realisiert.

Subscription Manager Ein Konsument kann ereignisbasierte Abfragen bei einem Vermittler subskribieren, um sich bei Auftreten bestimmter zukünftiger Ereignisse informieren zu lassen. Die Abfragen werden kontinuierlich gegen eintreffende Sensordaten ausgewertet und im Falle einer Übereinstimmung mit einem komplexen Ereignis beantwortet. Eine Abfrage bleibt so lange aktiv, bis sie explizit durch den Konsumenten invalidiert (gelöscht oder geändert) wird.

Um den Konsumenten bei der Erstellung und Verwaltung seiner Abfragen zu unterstützen wurde im Rahmen dieser Arbeit ein *Subscription Manager* (SM) entwickelt, welcher schließlich im *MagicTracker*-Projekt [BIK09a, IBK09] eingesetzt wurde. Der SM leitet den Benutzer bei der Spezifikation von Ereignismustern an, erlaubt die Angabe von Ereignistypabbildungen, führt ein Repository mit subskribierten Abfragen, übernimmt den Subskriptionsvorgang, erlaubt die Überwachung komplexer Ereignisse etc. Screenshots in Abbildung 7.33 geben einen Eindruck des SM. Mit Hilfe dieses Werkzeugs vereinfacht sich das Erstellen und Verwalten von Subskriptionen für den Konsumenten.

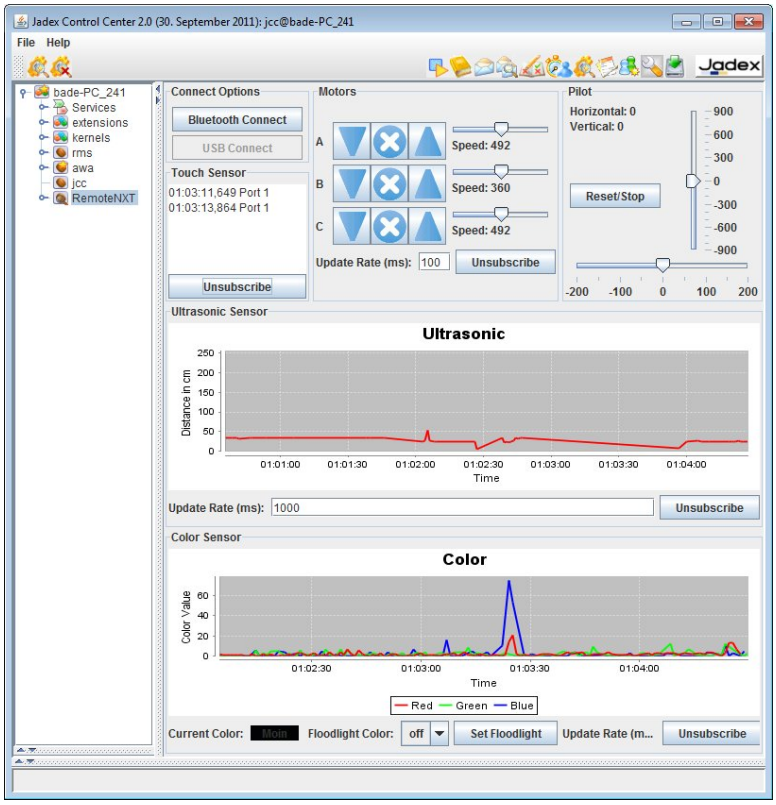


Abbildung 7.32.: Screenshot des Mindstorm Control Centers

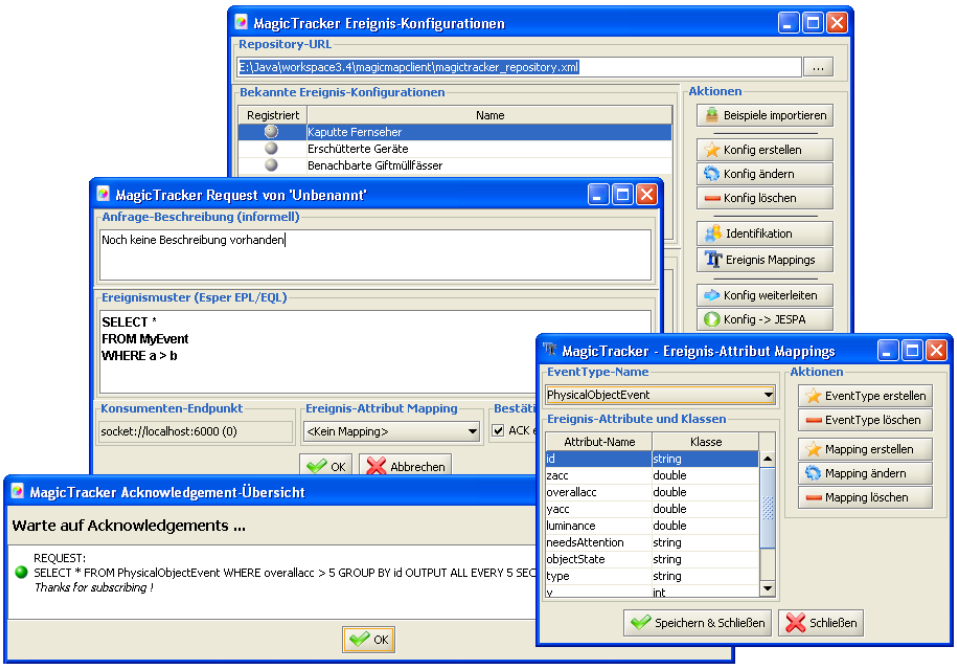


Abbildung 7.33.: Screenshot des JESPA Subscription Managers (BIK09b)

Deployment Monitor Die einzelnen Komponenten, die am Vermittlungsprozess beteiligt sind, können beliebig häufig instanziiert und im Netz verteilt werden, um so die Verarbeitungslast zu balancieren. Auch können zur Laufzeit jederzeit neue Verarbeitungsknoten hinzugefügt oder existierende Knoten heruntergefahren werden. Auf diese Weise ist es möglich den Prozess horizontal und vertikal skalieren zu lassen.

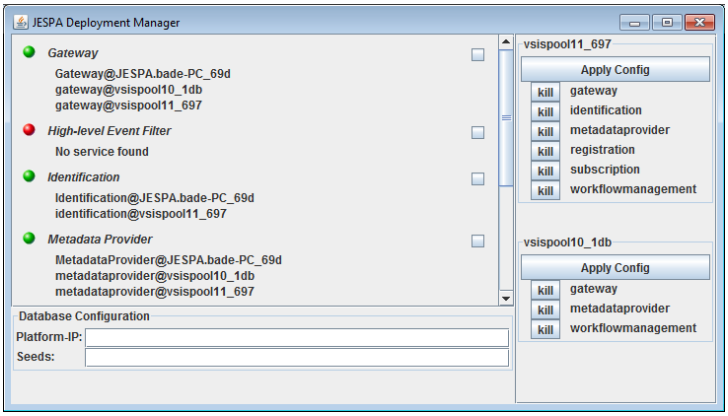


Abbildung 7.34.: Screenshot des JESPA Deployment Monitors

Jedoch muss von jeder Kernkomponente mindestens eine Instanz laufen, ansonsten kann der Vermittlungsprozess nicht vollständig durchgeführt werden. Um hier den Überblick zu behalten, wurde ein *Deployment-Monitor* entwickelt. Dieser durchsucht in regelmäßigen Abständen das Netz nach laufenden Instanzen und zeigt diese in einer Übersicht (vgl. Abbildung 7.34) an. Hier kann ein Administrator einzelne Dienstinstanzen manuell beenden oder neu starten. Ein spezieller Modus erlaubt zudem das automatische Starten von Diensten, sobald der Ausfall einer Kernkomponente festgestellt wurde.

Data Access Query Support Eine Reihe von Kernkomponenten (z.B. die Metadatenbereitstellung, der Identitätsdienst, die Registrierung etc.) persistieren Informationen von Produzenten und Konsumenten, Abfragen, Kennzahlen etc. in einer verteilten Datenhaltung. Zum Testen wurde daher ein Werkzeug entwickelt, welches die Inhalte der verteilten Keyspaces der einzelnen Komponenten anzeigen kann und welches das Filtern von Inhalten unterstützt. Dieses Werkzeug kann jedoch auch klientenseitig zum Beispiel für die Suche nach geeigneten Diensten für Verarbeitungs-Workflows, zur Abfrage von Metainformationen über registrierte Produzenten oder dem gezielten Abfragen von persistierten Sensorinformationen eingesetzt werden. Ein Screenshot des Werkzeugs ist in Abbildung 7.35 dargestellt.

Profiler Mit Hilfe sogenannter *Profiling*-Werkzeuge lässt sich das Laufzeitverhalten einer Anwendung analysieren. Gängige Werkzeuge erlauben beispielsweise einen detaillierten Blick auf die Speicherbelegung, Aufrufzeiten, Thread-Lebenszyklen etc. einer lokal laufenden Anwendung. Um derartige Kennzah-

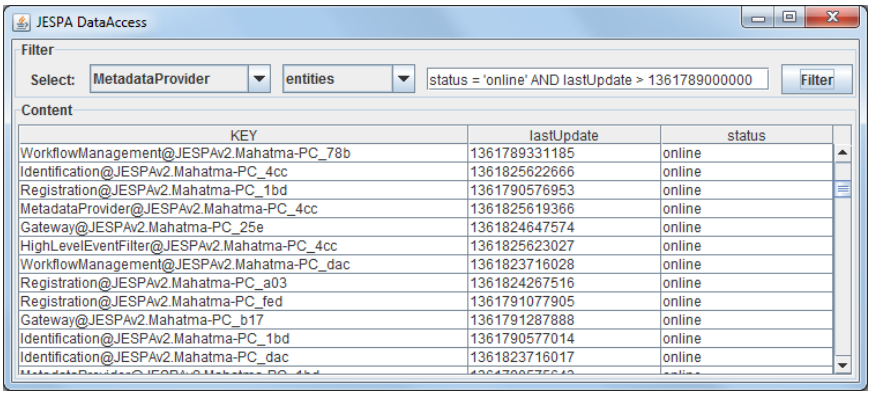


Abbildung 7.35.: Screenshot des JESPA Data Access Query Support

len auch für den verteilten Vermittlungsprozess zu erheben, wurde ein eigenes, einfaches Profiling-Werkzeug entwickelt, welches bei Bedarf Benchmark-Daten von Diensten einsammeln und verschiedene Statistiken zu Durchlaufzeiten von Methodenaufrufen generieren kann (dargestellt in Abbildung 7.36). Mit Hilfe dieser Statistiken lassen sich beispielsweise Knoten identifizieren, die aufgrund geringer Ressourcenausstattung einen Flaschenhals darstellen, oder es lässt sich die Last einzelner Knoten abschätzen sowie die Wirkung von Änderungen im Quellcode auf die Systemleistung ablesen etc. Dieses Werkzeug dient entsprechend dem Testen, der Kontrolle und der strategischen Planung, welche Dienste auf welchen Knoten angesiedelt werden sollen.

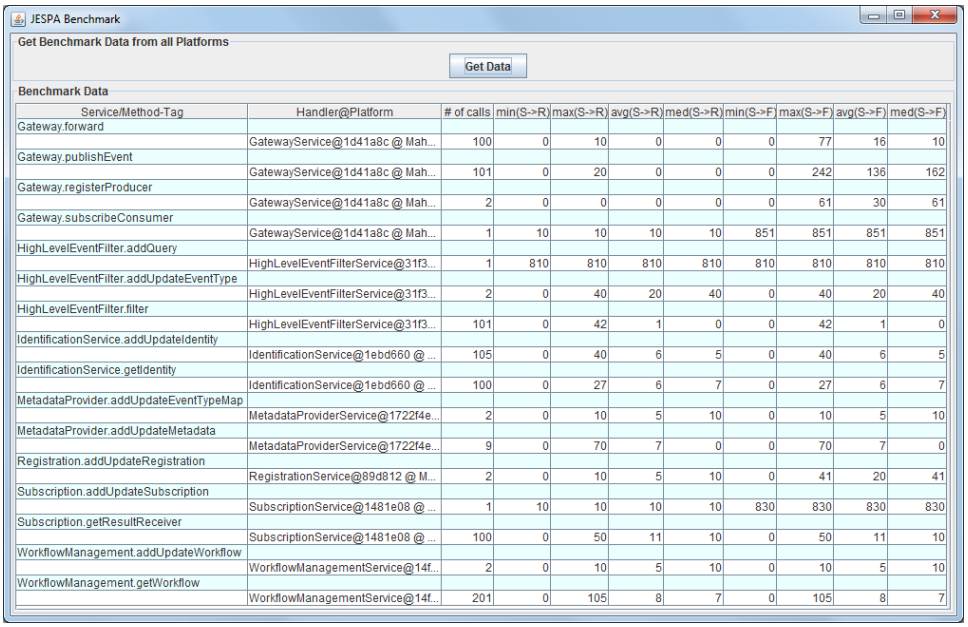


Abbildung 7.36.: Screenshot des JESPA Profilers

7.2.2.7. Zusammenfassung der prototypischen Realisierung

Die prototypische Realisierung eines Vermittlers baut auf einer Reihe existierender Technologien wie dem Jadex Active Components-Rahmenwerk, der Esper Complex Event Processing-Laufzeitumgebung, der Spezifikation von Verarbeitungsanweisungen mittels BPMN-basierter Workflows sowie dem verteilten, nicht-relationalen Datenbanksystem Cassandra auf.

Unter Verwendung dieser Technologien wurden eine Reihe stark kohäsiver, lose gekoppelter Komponenten entwickelt, welche sämtliche Kernfunktionalitäten sowie einige optionale Funktionen für die Vermittlung kontextbasierter Sichten bereitstellen. Die Komponenten können rein lokal auf Standardcomputern und Android-basierten Geräten ausgeführt sowie beliebig in lokalen und globalen Netzen verteilt werden, wodurch der Vermittlungsprozess problemlos in mobilen und ubiquitären Systemen mit ihren besonderen Herausforderungen umgesetzt werden kann.

Zusätzlich wurden eine Reihe von Werkzeugen zur Unterstützung von Produzenten und Konsumenten sowie zur Administration, zum Testen und zur Leistungsüberwachung eines Vermittlers entwickelt, was den praktischen Einsatz vereinfacht und unterstützt.

7.3. Zusammenfassung

In diesem Kapitel wurden der Entwurf und eine exemplarische Implementierung einer Middleware für die Vermittlung kontextbasierter Sichten in mobilen, ubiquitären Systemen präsentiert. Diese besteht aus einer Reihe lose gekoppelter, stark kohäsiver aktiver Komponenten, die kooperativ den Vermittlungsprozess realisieren. Der Vorteil des Softwareentwicklungsparadigmas der aktiven Komponenten liegt in dessen Eignung insbesondere für die Realisierung von Anwendungen aus dem Bereich des Ubiquitous Computing, zu denen auch die Vermittlung kontextbasierter Sichten gehört. Den hiermit einhergehenden Herausforderungen an die Verteilung, Nebenläufigkeit, nichtfunktionalen Anforderungen und die Dynamik der Umwelt kann mit Hilfe aktiver Komponenten auf hohem Abstraktion begegnet werden.

Die Middleware untergliedert sich in eine Reihe von Kernkomponenten, die so ausgelegt sind, dass ein grundlegender Vermittlungsprozess durchgeführt werden kann, jedoch lediglich geringe Anforderungen an die Ausführungsumgebung gestellt werden. Auf diese Weise können auch ressourcenbeschränkte Geräte an dem Prozess mitwirken. Zusätzlich wurden eine Reihe optionaler Erweiterungen vorgeschlagen, welche für Klienten, Entwickler und Administratoren eines Vermittlers Mehrwertdienste erbringen. Um einen solchen Vermittler in existierende Systeme integrieren zu können, müssen auch etablierte Standards aus den Bereichen des Internet der Dinge sowie des Sensor Webs berücksichtigt werden. Hierfür wurden eine Reihe unterschiedlicher Integrationsvarianten vorgestellt und diskutiert.

Für die exemplarische, prototypische Realisierung eines Vermittlers als Proof-of-Concept eignet sich das *Jadex Active Components*-Rahmenwerk, welches eine passende (verteilte) Laufzeitumgebung für aktive Komponenten bereitstellt. Mittels der gebotenen Programmierschnittstellen wurden alle Kernkomponenten sowie ausgewählte Erweiterungen und eine Reihe von Werkzeugen im Rahmen des Promotionsprojekts exemplarisch realisiert. Der daraus resultierende Prototyp kann Sensordaten von Produzenten entgegennehmen, nach deren Maßgaben verarbeiten (anreichern, konvertieren, persistieren etc.) und anschließend einem System zur komplexen Ereignisverarbeitung zuführen. Konsumenten können ihrerseits Abfragen bei einem Vermittler subskribieren, deren Ergebnis nach ihrer Maßgabe verarbeiten und die daraus resultierenden kontextbasierten Sichten mittels unterschiedlicher Kommunikationstechnologien zustellen lassen. Damit wurde gezeigt, dass die vorgeschlagene Systemarchitektur mit geeigneten Systemkomponenten umsetzbar und (prinzipiell) mit den geforderten Eigenschaften realisierbar ist.

Im nächsten Kapitel folgen eine qualitative und eine quantitative Evaluation des Prototypen, welche jeweils die Erfüllung der wesentlichen funktionalen und nicht-funktionalen Anforderungen aus Abschnitt 5.3 aufzeigen.

8. Evaluation des eigenen Ansatzes

In diesem Kapitel wird eine Evaluation des vorgeschlagenen Konzepts zur Realisierung der Vermittlung kontextbasierter Sichten präsentiert. Diese Evaluation soll aufzeigen, welche Anforderungen erfüllt bzw. nicht erfüllt werden konnten. Hierzu werden in Abschnitt 8.1 zunächst die Evaluationskriterien vorgestellt. Im Anschluss daran erfolgt in Abschnitt 8.2 eine qualitative Evaluation anhand zweier Forschungsprojekte, welche in deskriptiver Weise das Konzept und dessen Umsetzung bewertet. Während sich die meisten Anforderungen ausschließlich qualitativ beschreiben lassen, gibt es einige insbesondere nicht-funktionale Anforderungen, die sich auch quantitativ, das heißt durch Erhebung bestimmter Kennzahlen im Betrieb der Middleware, evaluieren lassen. Der hierfür verwendete Versuchsaufbau, die jeweils durchgeführten Versuchsreihen sowie die Ergebnisse werden in Abschnitt 8.3 erläutert.

8.1. Evaluationskriterien

Für die Evaluation des eigenen Ansatzes werden dieselben Kriterien verwendet, wie für die Bewertung verwandter Arbeiten aus Abschnitt 6.3. Ausgehend von den funktionalen und nicht-funktionalen Anforderungen, die in den Abschnitten 5.3.1 bzw. 5.3.2 identifiziert wurden, erfolgt zunächst eine *qualitative Evaluation*, im Rahmen derer qualitative Aussagen über die JESPA Middleware getroffen werden. Diese beziehen sich auf die Frage „was das System leistet“ (funktionale Anforderungen) und „welches Verhalten zu erwarten ist“ (nicht-funktionale Anforderungen). Wie gut oder schlecht eine Leistung im Einsatz wirklich erbracht wird, ist Teil der *quantitativen Evaluation*. Hierbei werden Laufzeitdaten evaluiert, die während der Ausführung von Versuchsreihen erhoben wurden.

Während die qualitative Evaluation beispielsweise zu dem Ergebnis kommen könnte, die Middleware müsse gut horizontal skalieren, da die konstituierenden Komponenten einfach über mehrere Knoten hinweg verteilt werden können, müsste die quantitative Evaluation dieses Ergebnis durch Laufzeitanalysen bestätigen können. Laufzeitanalysen betreffen jedoch im Wesentlichen die Leistungsaspekte des Systems, da diese quantitativ erfassbar sind. Zum Beispiel lässt sich die Erfüllung nicht-funktionaler Anforderungen wie Erweiterbarkeit oder Flexibilität schwerlich quantitativ belegen.

8.2. Qualitative Evaluation

Wie bereits erwähnt, geht es bei der qualitativen Evaluation um die Fragen „was das System leistet“ bzw. „ob es die gestellten Anforderungen erfüllt“. Zur

exemplarischen Beantwortung dieser Fragen wurden zwei komplexere Anwendungsbeispiele umgesetzt, anhand derer evaluiert werden kann, ob die aufgestellten Anforderungen erfüllt werden. Das Ergebnis besteht im Wesentlichen aus einer Beschreibung des Systems sowie dessen Eigenschaften und Verhalten. Jedoch werden nicht alle aufgestellten Anforderungen von den Anwendungsbeispielen abgedeckt. Auf der einen Seite werden diese durch Top Down-Analyse erhobenen Anforderungen (vgl. Abschnitt 5.3) mittels einer Beschreibung der Eigenschaften und der konkret umgesetzten Funktionen des Systems bewertet. Auf der anderen Seite existieren auch eine Reihe optionaler, durch eine Bottom Up-Analyse erhobene Anforderungen, deren Umsetzung im Rahmen der Beispiele und des Vermittlungsprozesses im Allgemeinen nicht notwendig waren und den Umfang einer exemplarischen, prototypischen Implementierung übersteigen würden - und teils (z.B. im Fall semantischer Abbildungen und Ableitungen) auch noch erheblicher Forschungsanstrengungen bedürfen.

Im Folgenden werden zunächst die beiden Evaluationsprojekte *MagicTracker* und *AEPIO/SOLA* vorgestellt, bevor im Anschluss die Ergebnisse präsentiert werden.

8.2.1. Evaluationsprojekt MagicTracker

Der *MagicTracker* wurde als Erweiterung für die Funkortungslösung *MagicMap* entwickelt, welche der Ermittlung und Visualisierung von Zustandsdaten (z.B. Position) mobiler Objekte dient [BZIS07, IBK09]. Für die Erfassung von Zustandsdaten können eine Reihe unterschiedlicher Sensoren an MagicMap angebunden werden, welches die erhobenen Sensordaten auf einer Karte darstellen und mit anderen MagicMap-Klienten austauschen kann. Die Aufgabe des MagicTracker besteht darin, objektbezogene Prozesse zu überwachen und bei Abweichung des aktuellen Ist-Zustandes von einem durch den Prozess definierten Soll-Zustand den Anwender zu informieren und ggf. Handlungsoptionen in Form alternativer Prozesspfade vorzuschlagen. Hierfür muss der MagicTracker die Sensordaten, die von MagicMap bereitgestellt werden, aggregieren und korrelieren, nach komplexen Mustern in den Daten suchen und bei erkannten Abweichungen bestimmte Zustandsattribute der Objekte setzen, damit diese in MagicMap entsprechend visualisiert werden können.

Abbildung 8.1 veranschaulicht dies an einem fiktiven Beispiel. In dem hier dargestellten Screenshot eines Logistik-Szenarios, ist die Karte eines Warenhauses zu sehen, in welchem eine Reihe von Paletten, Arbeitern und Vehikeln visualisiert sind. Am linken Rand der Karte befindet sich ein LKW, der gerade Paletten mit Fernsehern entlädt. Die Fernseher sind mit Beschleunigungssensoren ausgestattet, sodass starke Erschütterungen, die zu einer Beschädigung und somit zu einer Eskalation im zugehörigen Prozess führen können, erfasst und dokumentiert werden können.

In der MagicMap-Anwendung laufen die Sensor- sowie Positionsdaten aller Objekte zusammen und werden zur Auswertung an das MagicTracker-Plugin

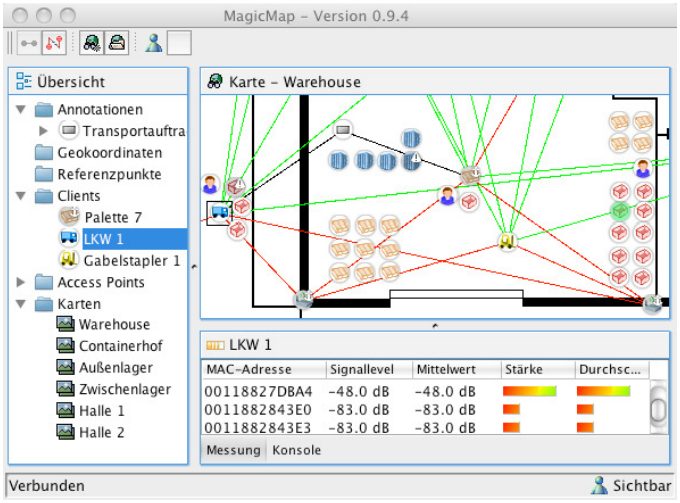


Abbildung 8.1.: Beispiel: Screenshot der MagicMap-Funkortungslösung (aus (IBK09))

übergeben, dessen Kernfunktion durch die JESPA Middleware realisiert wird. Hier sind für alle Objekte bzw. Objekttypen entsprechende Ereignismuster, d.h. ereignisbasierte Abfragen, definiert. Für die oben erwähnten Fernseher liefert zum Beispiel folgende Abfrage den Zeitpunkt, die Position und die Kennung einer Palette sowie des entsprechenden Gabelstaplers, sobald eine Gesamtbeschleunigung von mehr als 3.0 m/s^2 festgestellt wurde:

```
SELECT current_timestamp(), pc.id, pc.pos, p.id, p.sensor.acc
FROM PalletEvent AS p, PalletCarrierEvent AS pc
WHERE pc.load.id = p.id AND p.sensor.acc > 3.0
```

Liefert die Abfrage ein Ergebnis, so wird durch JESPA ein Workflow ausgeführt, welcher dem von MagicMap verwendeten Objektmodell weitere Attribute hinzufügt, sodass die Objektvisualisierung auf der Karte entsprechend angepasst werden kann (in der Abbildung z.B. durch Graufärben des Symbols und Hinzufügen eines 'Achtung'-Zeichens). Auf diese Weise bildet JESPA die Basis zur Unterstützung der Nutzerinteraktionen mit MagicMap, indem vorhandene Kontextinformationen aufbereitet und der MagicMap-Software in Form einer kontextbasierten Sicht (aktualisiertes Objektmodell von MagicMap) zur Verfügung gestellt werden. Die Empfehlung alternativer Handlungsoptionen für den weiteren Prozessverlauf eines Objektes wurde in diesem Projekt lediglich konzeptionell behandelt.

8.2.2. Evaluationsprojekt AEPIO / SOLA

Das zweite Projekt für eine qualitative Evaluation der Umsetzung des Konzepts zur Realisierung der Vermittlung kontextbasierter Sichten basiert auf einem Anwendungsszenario aus dem Bereich des Internet der Dinge (vgl. Abschnitt 3.2.2). Ausgehend von der Feststellung, dass sich Forschung und praktische Ansätze in diesem Kontext in erster Linie auf den Umgang mit

Objektdaten konzentrieren, wurde ein neuer prozessfokussierter Ansatz entwickelt, der die dynamische Adaptierung von objektbezogenen (Geschäfts-)Prozessen auf Basis von Ereignissen im Lebenszyklus intelligenter Objekte erlaubt [KFZB11].

Ein Objekt wird als „intelligent“ bezeichnet, sofern es eine digitale Identität besitzt (z.B. durch einen angebrachten RFID Chip), den eigenen Zustand sowie bestimmte Parameter der Umwelt mittels Sensoren erfassen und diese Informationen mit anderen Entitäten (z.B. Objekten oder Diensten) austauschen kann [MF10]. Im Rahmen des Lebenszyklus solcher Objekte, beginnend mit der Fertigung über den Transport, den Verkauf, die Nutzung und schließlich der Entsorgung, können Ereignisse den Zustand eines Objekts beeinflussen. Im Rahmen dieses Projekts wurde der Lebenszyklus von Objekten als Prozess modelliert, wobei insbesondere auch antizipierte Ereignisse und Reaktionen auf diese berücksichtigt wurden.

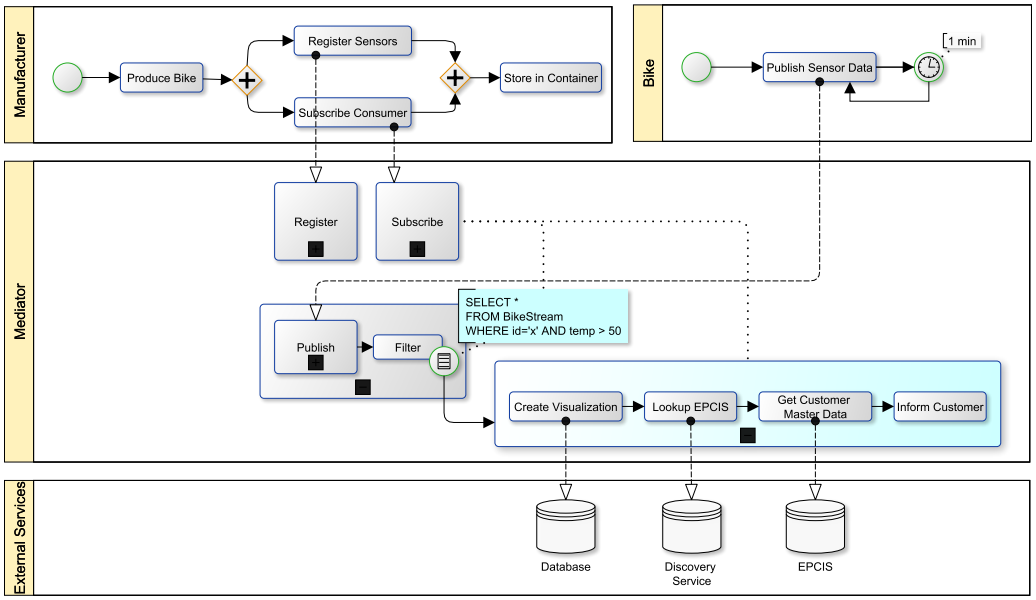


Abbildung 8.2.: Beispiel: Lebenszyklus eines intelligenten Objekts (im Original beschrieben in (KBFZ11, Kun11))

Abbildung 8.2 zeigt beispielhaft ein vereinfachtes BPMN-basiertes Modell eines Ausschnitts aus dem Lebenszyklus eines Elektrofahrrads (siehe Anwendungsbeispiel in Abschnitt 5.2.3). Ausgestattet mit einer Reihe von Sensoren (z.B. zur Überwachung der Akkukapazität und -temperatur) kann das Fahrrad Informationen über die Nutzung sowie über seinen aktuellen Zustand per Funk an andere Geräte (z.B. den zum Transport verwendeten Smart Container oder das Mobiltelefon eines LKW-Fahrers) melden. An bestimmten Punkten des Zyklus wurde der Prozess mit Bedingungsereignissen annotiert. Eine solches Ereignis beschreibt dabei die Reaktion auf eine antizipierte Ausnahmesituation, z.B. dem Auslösen eines Alarms bei Überschreiten eines Grenzwertes für die Akkutemperatur.

Wie in der Abbildung zu erkennen, wird ein Bedingungsereignis mit einer ereignisbasierten Abfrage ($SELECT * FROM...$) annotiert und mit einem Subprozess, d.h. der auszuführenden Eskalationsbehandlung, verknüpft. Die Sensoren des Fahrrads melden (engl. *publish*) jede Minute aktuelle Messwerte an einen Vermittler. Dieser verarbeitet die Messwerte und übergibt sie schließlich dem Filterdienst, bei dem zuvor die entsprechende ereignisbasierte Abfrage subskribiert wurde. Liefert diese ein Resultat, wird der damit verknüpfte Subprozess ausgeführt, in dessen Folge Zeitreihen-Visualisierungen erstellt und zusammen mit Informationen aus dem Stammblatt des Kunden an diesen übermittelt werden.

Im Rahmen des Projekts sollten die antizipierten Ereignisse innerhalb des Lebenszyklus von intelligenten Objekten überwacht und im Fall, dass ein solches Ereignis auftritt, eine entsprechende Eskalationsbehandlung zeitnah vorgenommen werden. Hierfür wurde das AEPIO/SOLA-System[1] entwickelt [KFZB11], welches in Abbildung 8.3 dargestellt ist. Das Gesamtsystem besteht aus einer Reihe von Diensten, die teils in den Standards für das Internet der Dinge durch EPCglobal/GS1 definiert wurden (vgl. Abschnitte 6.2.1 und 7.1.4.2). Hierzu gehören die *EPC Information Services* (EPCIS), welche beliebige Daten zu Objektinstanzen vorhalten können, sowie ein *Discovery Service* zum Auffinden relevanter EPCIS zu einer gegebenen Objektkennung. Des Weiteren finden sich eine Komponente zur Definition und Verwaltung oben erwähnter Lebenszyklusprozesse für einzelne Objektinstanzen (*Escalation Administration Point*), ein Proxy für den transparenten Zugriff auf den verteilten Discovery Service (*Discovery Service Proxy*) sowie die JESPA Middleware.

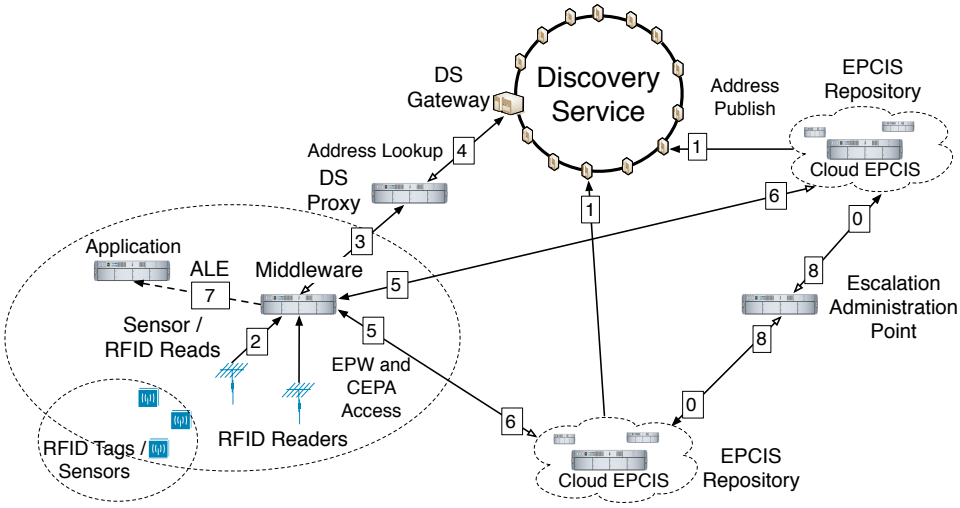


Abbildung 8.3.: AEPIO/SOLA-System zur Überwachung des Lebenszyklus intelligenter Objekte (KFZB11)

Ziel ist, eine zeitnahe Reaktion auf antizipierte Ereignisse im Lebenszyklus von Objekten zu ermöglichen. Hierzu müssen in einem ersten Schritt die Eska-

[1] *Adaptive Edge Processing for Intelligent Objects* (AEPIO) bzw. *Smart Objects Lifecycle Architecture* (SOLA)

lationsstrategien, bestehend aus komplexen Ereignismustern (*Complex Event Pattern*, CEPA) sowie sogenannten *Escalation Processing Workflows* (EPW), am Escalation Administration Point erstellt werden. Diese Strategien können individuell für einzelne Objektinstanzen angelegt und in entsprechenden EPCIS Repositories[2] hinterlegt werden (Schritt 0). Um ein späteres Auffinden passender EPCIS zu einer gegebenen Objektkennung zu ermöglichen, werden diese bei einem verteilten *EPC Discovery Service* registriert (Schritt 1), welcher im Gegensatz zum *Object Naming Service* (vgl. Abschnitt 7.1.4.2) die Auflösung von EPCIS-Adressen auch auf Objektinstanz-Ebene erlaubt.

Im Anschluss an diese Vorbereitungsphase kann die Überwachung des intelligenten Objektes beginnen. Eine zentrale Rolle übernimmt hierbei die JESPA Middleware. Sobald zum ersten Mal die digitale (RFID- bzw. EPC-basierte) Identität sowie Sensordaten des zu überwachenden Objektes über einen JESPA Gateway-Dienst zur weiteren Verarbeitung bei dem Vermittler eintreffen (Schritt 2), wird das Objekt automatisch als Sensordatenproduzent registriert. Hierfür wendet sich der Vermittler an einen *Discovery Service Proxy* (Schritt 3), um zu der gegebenen EPC ein passendes EPCIS Repository aufzufinden (Schritt 4). Aus dem Repository werden die Eskalationsbeschreibungen für die Objektinstanz abgerufen (Schritte 5 und 6) und einerseits die zu überwachenden Ereignisse (als ereignisbasierte Abfragen) beim Filterdienst von JESPA subskribiert und andererseits die Eskalationsbehandlungen in Form von BPMN Workflows in der Workflow-Verwaltung hinterlegt.

Von nun an werden alle eintreffenden Sensordaten des intelligenten Objektes gegen die subskribierten Abfragen ausgewertet. Liefert eine Abfrage ein Ergebnis, beginnt das Erzeugen einer kontextbasierten Sicht durch Ausführung des Eskalationsprozesses. Je nach Ereignis und Anwendungsfall könnte dies zum Beispiel das Benachrichtigen einer bestimmten Anwendung (Schritt 7) sein.

Die JESPA Middleware fungiert in diesem Projekt als ein Vermittler zwischen intelligenten Objekten als Produzenten von Sensordaten sowie etwaigen Objektadministratoren als Konsumenten. Hierbei koordiniert sie das Zusammenspiel aller beteiligten Instanzen (Discovery Service, Cloud EPCIS, Anwendung) und kann nicht nur Informationen über Objektzustände in Form kontextbasierter Sichten weiterleiten, sondern auch direkt die Eskalationsbehandlung durch benutzerdefinierte Eskalationsprozesse ausführen.

8.2.3. Ergebnisse

Eine Übersicht der Evaluationsergebnisse aus den beiden oben beschriebenen Evaluationsprojekten findet sich in Abbildung 8.4. Wie bei dem Vergleich verwandter Arbeiten in Abschnitt 6.3 teilen sich die Ergebnisse in unterschiedliche Aufgabenbereiche des Vermittlungsprozesses auf (vgl. Abschnitt 4.4). Im Folgenden sollen diese detailliert erläutert werden.

[2] Im Projekt selbst wurde ein Cloud EPCIS entwickelt und verwendet, was jedoch im Kontext dieser Arbeit unerheblich ist.

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	JESPA v2
Funktionale Anforderungen	
1. Datenübermittlung	
1.1 Kommunikationsadapter	++
1.2 Logische Adressierung	++
1.3 Push- und Pull-basierte Kommunikation	++
1.4 Mobilitätsverwaltung	++
2. Datenverarbeitung	
2.1 Orchestrierung von Verarbeitungsfunktionen	++
2.2 Angebot typischer Verarbeitungsfunktionen	+
2.3 Benutzerdefinierte Verarbeitungsfunktionen	++
2.4 Auffinden von Verarbeitungsdiensten	++
2.5 Zustandslose/-behaftete Verarbeitungsdienste	++
3 Datenverwaltung	
3.1 Unterschiedliche Datenbanksysteme	++
3.2 Verteiles Persistieren von Zeitreihen	++
3.3 Semantische Abbildungen und Ableitungen	-
4 Prozessverwaltung	
4.1 Logging & Accounting	+
4.2 Testing & Debugging	+
4.3 Monitoring & Reporting	++
4.4 Ressourcenverwaltung	-
4.5 Zugriffskontrolle	+
5 Metadatenverwaltung	
5.1 Unterschiedliche Datenbanksysteme	++
5.2 Verteiles Persistieren von Metadaten	++
5.3 Semantische Abbildungen und Ableitungen	-
5.4 Metadatenermittlung	-
6 Datenabfrage	
6.1 Historische Abfragen	++
6.2 Ereignisbasierte Abfragen	++
6.3 Ad-hoc Abfragen	-
6.4 Verschiedene ad-hoc Abfragestrategien	-
6.5 Multimediale Abfragen	-
6.6 Test-Unterstützung für ereignisbasierte Abfragen	++
7 Auftragsverwaltung	
7.1 Delegation von Aufträgen	-
7.2 Durchführbarkeitsanalyse von Aufträgen	-
7.3 Statusüberwachung von Aufträgen	-
Nicht-funktionale Anforderungen	
Offenheit	+
Heterogenität	++
Flexibilität	++
Erweiterbarkeit	++
Verteilung	++
Skalierbarkeit	++
Robustheit	++
Transparenz	+
Sicherheit	+
Lose Kopplung	++
Starke Kohäsion	++
Standards	++

Abbildung 8.4.: Übersicht der Evaluationsergebnisse des eigenen Ansatzes

Datenübermittlung Um die Heterogenität der Produzenten und Konsumenten und den von diesen verwendeten Kommunikationstechnologien zu verbergen, lassen sich unterschiedliche Kommunikationsadapter in JESPA integrieren. Die Gateways operieren ausschließlich auf dem TCP-Stack und unterstützen die Kommunikation über TCP- und UDP-Sockets, FIPA-konforme Agentennachrichten, Active Components-Dienstaufrufe[3] sowie SOAP- und RESTful Web Service-Schnittstellen. Für die Anbindung weiterer Klienten auf unteren Protokollebenen wurde ein leichtgewichtiger Proxy entwickelt, welcher nicht auf dem Jadex-Rahmenwerk aufbaut und somit prinzipiell auch auf ressourcenschwachen Geräten lauffähig ist. Dieser lässt sich über entsprechende Programmierschnittstellen einfach um weitere Protokolladapter erweitern.

Um eine Kommunikation auch über die unterschiedlichen Technologien und Protokolle hinweg zu gewährleisten, realisiert JESPA eine logische Adressierung auf Basis von URIs. Alle Dienstinstanzen des Vermittlungsprozesses bekommen während der Instanziierung eine eindeutige UUID zugewiesen, über die sie direkt adressiert werden können. Dasselbe gilt für registrierte Produzenten sowie subskribierte Konsumenten und deren Stellvertreter. Zusätzlich lassen sich Dienste auch auf Typebene durch ein festgelegtes Namensschema adressieren. Da einzelne Dienste zur besseren Skalierung beliebig oft im System repliziert werden können, erreicht man so eine lose Kopplung. Das Binden konkreter Instanzen als Endpunkt einer typbasierten Adressierung führt JESPA zur Laufzeit automatisch durch.

Für den Umgang mit mobilen Geräten realisiert JESPA eine einfache Mobilitätsverwaltung. Bei eintreffenden Nachrichten von Klienten wird deren aktuelle Endpunktadresse, falls verfügbar, beim Identitätsdienst hinterlegt. Sofern Nachrichten an Klienten, z.B. das Ergebnis einer ereignisbasierten Abfrage, push-basiert zugestellt werden sollen, kann die jeweils zuletzt bekannte Adresse des Klienten von dort wieder abgerufen werden. Falls eine Nachricht an einen Klienten nicht zugestellt werden konnte, wird sie in einem Nachrichtendienst zwischengespeichert, welcher periodisch erneute Zustellungsversuche unternimmt. Sofern Klienten die pull-basierte Kommunikation bevorzugen, können sie über das Gateway ihre Nachrichten vom Nachrichtendienst selbst abrufen.

Datenverarbeitung Produzenten und Konsumenten können einen Vermittler anweisen, Sensordaten bzw. kontextbasierte Sichten nach ihrer Maßgabe zu verarbeiten bzw. aufzubereiten. Hierfür werden vermittlerseitig eine Reihe von Verarbeitungsfunktionen für typische Anwendungsfälle angeboten. Diese können von Klienten durch Angabe BPMN-basierter Workflows orchestriert werden, was das Erstellen komplexer Ausführungssequenzen ermöglicht, welche schließlich durch die Jadex-Laufzeitumgebung ausgeführt

[3] Das Jadex Active Components-Rahmenwerk bietet zudem auch verschlüsselte Kommunikation sowie in der mobilen Version für die Android-Plattform auch die Kommunikation über Bluetooth.

werden. Zusätzlich ist es den Klienten möglich, Aufrufe benutzerdefinierter Verarbeitungsfunktionen mittels externer Dienstreferenzen in den Workflow zu integrieren.

Für die Erstellung von Workflows ist es unerlässlich, eine Übersicht verfügbarer Verarbeitungsfunktionen zu haben bzw. nach bestimmten Funktionen suchen zu können. Alle verfügbaren Funktionen werden mit ihren Metadaten, d.h. einer Funktionsbeschreibung sowie einer informellen Beschreibung der Ein- und Ausgabeparameter, in der Metadatenbereitstellung veröffentlicht, welche schließlich von Klienten für die Dienstauswahl abgefragt werden kann.

Die Anforderung zustandsbehafteter Dienste wird durch die Möglichkeiten i) der Angabe externer Dienstreferenzen, ii) dem Persistieren und Wiederherstellen eines Zustandes in der verteilten Datenhaltung sowie iii) einer speziellen Markierung zustandsbehafteter Dienste und ihrer Veröffentlichung in der Metadatenbereitstellung realisiert.

Datenverwaltung Im Rahmen der Datenverwaltung wurde die Unterstützung unterschiedlicher Datenbanksysteme gefordert, um den mannigfaltigen Anforderungen seitens der Anwendungen gerecht zu werden. JESPA kapselt das konkret verwendete Datenhaltungssystem hinter einer generischen, technologieneutralen Schnittstelle, welche die typischen CRUD-Operationen anbietet. Dies ermöglicht es verschiedenen Komponenten unterschiedliche Systeme zu nutzen. Anhand zweier Implementierungsvarianten wurde die Realisierbarkeit dieses Konzepts nachgewiesen: i) Eine verteilte, nicht-relationale Datenbank erlaubt das Speichern von Daten in einem *Cassandra*-Cluster und ii) eine leichtgewichtige *In Memory*-Datenbank erlaubt das hochperformante Zwischenspeichern von Daten, was sich insbesondere für Mobilgeräte anbietet. Semantische Abbildungen und Ableitungen wurden in dem aktuellen Prototypen nicht realisiert, aber das prinzipielle Vorgehen lehnt sich an die in Abschnitt 7.1.5.3 vorgestellten Lösungsmöglichkeiten des konzeptionellen Ansatzes an.

Prozessverwaltung Zur Verwaltung des Vermittlungsprozesses gehören Funktionen zum Überwachen, Kontrollieren und Abrechnen von Dienstaufrufen. Das verteilte Logging wird durch ein eigenes Rahmenwerk zum Festschreiben, Kommunizieren und Filtern von Logeinträgen realisiert. Ein Dienst zur Abrechnung (Accounting) auf Basis dieser Protokolle wurde nicht realisiert, da in den Evaluationsprojekten monetäre Interessen unberücksichtigt blieben. Werkzeuge zum Testen und Debuggen werden durch das Jadex-Rahmenwerk geboten, welche z.B. eine feingranulare Ausführungssteuerung einzelner Programmschritte erlauben, die Kommunikationsbeziehungen visualisieren, Unit-Tests unterstützen etc. Spezielle Unit-Tests für einzelne Aufgaben der JESPA Middleware wurden zugunsten von Systemtests nicht realisiert. Der *JESPA Deployment Manager* erlaubt das Überwachen der Vermittlungsdienste zur Laufzeit sowie das (automatisierte, z.B. im Fehlerfall) ent-

fernte Starten von Dienstinstanzen. Mechanismen zur Ressourcenverwaltung, z.B. das Einbinden neuer oder das Umwidmen existierender Ressourcen bei zu hoher Last, wird noch nicht unterstützt. Eine einfache Zugriffskontrolle auf Basis von *Token* und *Token Templates* kann die nicht-autorisierte Nutzung von Diensten verhindern. Eine Zugriffskontrolle auf Daten wurde nicht realisiert, kann im Prototypen aber ansatzweise über Ver-/Entschlüsselungsfunktionen im Rahmen der Verarbeitungs-Workflows nachgeahmt werden.

Metadatenverwaltung Die Metadatenbereitstellung baut auf der Datenverwaltung auf und unterstützt somit unterschiedliche Datenhaltungssysteme sowie das verteilte Persistieren von Metadaten. Möglichkeiten zur semantischen Abbildung und Ableitung sowie zum automatischen Ermitteln von Metadaten durch aktives Suchen wurden nicht realisiert.

Datenabfrage Die Abfrage historischer Datensätze wird durch direktes Abfragen der Datenverwaltung realisiert, deren generische Schnittstelle das konkret verwendete Datenhaltungssystem vor dem Klienten verbirgt und somit einen transparenten Zugriff erlaubt. Ergebnisse können als Java-Objekte sowie im JSON-, XML- oder CSV-Format zurückgeliefert werden. Ereignisbasierte Abfragen können von Konsumenten beim Subskriptionsdienst hinterlegt werden, welcher den Lebenszyklus der Abfragen überwacht. Die Abfrageverarbeitung selbst übernimmt eine Complex Event Processing-Laufzeitumgebung. Das Testen ereignisbasierter Abfragen wird durch eine Reihe von Werkzeugen zum Generieren „künstlicher“ Ereignisse unterstützt. Ad-hoc Abfragen, die über die Auftragsverwaltung (siehe unten) laufen, sowie multimediale Abfragen sind in JESPA noch nicht realisiert.

Auftragsverwaltung Die Verwaltung von Aufträgen, was unter anderem das Weiterleiten von Aufträgen an die Produzenten sowie Statusanalysen und die Integration des OGC Sensor Planning Service (siehe unten) beinhaltet, wird von dem aktuellen Prototypen noch nicht unterstützt.

Nicht-funktionale Anforderungen Die JESPA Middleware erfüllt die nichtfunktionale Anforderung der Offenheit dahingehend, dass die Schnittstellen und verwendeten Datenaustauschformate dokumentiert und etablierte Standards zur Beschreibung von Schnittstellen und Datenaustauschformaten weitestgehend nicht nur im Rahmen des internen Vermittlungsprozesses, sondern auch in der Kommunikation mit Klienten berücksichtigt werden. Während die EPCglobal/GS1-Standards für das Internet der Dinge teils realisiert wurden, fehlt in der prototypischen Realisierung jedoch noch die Umsetzung der OGC-Standards für das Sensor Web, deren konzeptionelle Integration jedoch in Abschnitt 7.1.4 erläutert wurde.

Hinsichtlich der Unterstützung heterogener Geräte wurde, wie oben beschrieben, mittels einer Reihe von Kommunikationsadaptern sowie einem leichtgewichtigen Proxy die Anbindung auch ressourcenschwacher Geräte

als Produzenten und Konsumenten ermöglicht. Die prototypische Umsetzung der Middleware für die Android-Plattform zeigt zudem, dass mobile Geräte, z.B. Telefone, auch einzelne Aufgaben im Rahmen des Vermittlungsprozesses übernehmen und die Rolle eines Vermittlers sogar vollständig ausfüllen können. Informations- bzw. Datenheterogenität wird dadurch adressiert, dass einfache Datenmodelle für den Austausch von Daten zwischen Beteiligten verwendet werden und durch Unterstützung semi- und (un-)strukturierter Datenhaltungssysteme prinzipiell beliebige Kontext- und Metadaten persistiert werden können.

Flexibel ist die Middleware dahingehend, dass keine Einschränkungen hinsichtlich der Anwendungsdomänen und der zu verarbeitenden Kontextdaten gemacht werden. Der Vermittlungsprozess beruht zudem auf einer minimalen Menge obligatorischer Dienste, sodass ein flexibles Deployment, sowohl lokal auf einem Gerät als auch global über das Internet verteilt, möglich ist. Offenheit und Erweiterbarkeit sind weitere nicht-funktionale Eigenschaften der Middleware, welche die Flexibilität unterstützen.

Die Erweiterbarkeit wird insbesondere durch die lose Kopplung der am Vermittlungsprozess beteiligten Komponenten erreicht. So können konkrete Instanzen zu einer gegebenen logischen Adresse zur Laufzeit durch die Jadex-Laufzeitumgebung automatisch aufgefunden und müssen nicht (ggf. bereits zur Entwurfszeit) manuell konfiguriert werden. Wohldefinierte Schnittstellen verbergen konkrete Implementierungsdetails von Komponenten, wodurch Änderungen in einer Komponente nicht zwangsläufig auch Anpassungen anderer Komponenten nach sich ziehen. Auch der asynchrone Nachrichtenaustausch sowie die Verwendung der Metadatenbereitstellung zum Auffinden angebotener Verarbeitungsdienste fördern die lose Kopplung.

Das Paradigma der aktiven Komponenten sowie die Jadex-Ausführungsumgebung erlauben das einfache Verteilen der Vermittlungsaufgaben in einem Netzwerk. Da alle Dienste grundsätzlich zustandslos sind, können Dienste einfach repliziert und auf mehreren Knoten verteilt werden, wodurch das System prinzipiell gut horizontal skalieren kann. Einzig der Filterdienst kann durch Verwendung der *Esper Complex Event Processing*-Laufzeitumgebung nicht verteilt angeboten werden und stellt somit einen Flaschenhals und Single Point of Failure im Gesamtsystem dar. Zwar existieren (kommerzielle) Lösungen zur Verteilung auch dieser Komponente, in der vorliegenden Arbeit wurden sie jedoch nicht eingesetzt.

Die Fähigkeit von Jadex, Komponenten mittels verschiedener Mechanismen zur Laufzeit auffinden und binden zu können, bildet nicht nur eine wichtige Grundlage zur einfachen Verteilung der Dienste, sondern auch für die Robustheit des Systems. Der Ausfall einer oder mehrerer Komponenten kann so durch ein Neubinden oder ggf. das (entfernte) Neustarten von Komponenten kompensiert werden. Die Kommunikation zwischen Komponenten erfolgt sowohl im lokalen als auch im entfernten Fall über dieselben Schnittstellen, wodurch eine vollständige Netzwerktransparenz erreicht wird, da unabhängig vom Ort der Komponenten identische Operationen verwendet werden können. Auch der

Zugriff auf historische Daten und Metadaten erfolgt transparent bezüglich des verwendeten Datenhaltungssystems über eine generische *CRUD*-Schnittstelle. Lediglich für die Formulierung ereignisbasierter Abfragen muss den Konsumenten die verwendete Complex Event Processing-Laufzeitumgebung bekannt sein, da die Abfragen technologieabhängig sind.

Sicherheitsaspekte konnten in der prototypischen Realisierung nur peripher adressiert werden. Zwar kann der Zugriff auf Verarbeitungsfunktionen seitens der Produzenten und Konsumenten eingeschränkt und durch einfache Token und Token Templates für einzelne Klienten und Rollen geregelt werden, nicht jedoch der Zugriff auf Daten. Durch Verwendung SSL-basierter Kommunikationsprotokolle seitens der Jadex-Plattform kann die Kommunikation verschlüsselt werden, allerdings werden alle Daten unverschlüsselt an die Verarbeitungsdienste, das Datenhaltungssystem sowie die Complex Event Processing-Laufzeitumgebung weitergereicht. Hier bedarf es auch konzeptionell weiterer Überlegungen zur Sicherstellung der Integrität und Vertraulichkeit der Daten.

Die Verwendung etablierter Standards zieht sich durch den gesamten Vermittlungsprozess. Die Kommunikation baut in erster Linie auf dem TCP-Stack auf. Der Proxy kann über einfache TCP- und UDP Socket-Verbindungen angesprochen werden, Gateways unterstützen neben FIPA-konformen Agentennachrichten auch SOAP- und RESTful Web Service-Schnittstellen, die mittels HTTP angesprochen werden. Die Kodierung von Nachrichten erfolgt je nach Anforderung der Klienten über Java-Objekte oder XML-, JSON- oder CSV-kodierte Nachrichten. Historische Abfragen können außer über die CRUD-Schnittstelle auch mittels der Abfragesprache des jeweils verwendeten Datenhaltungssystems (z.B. in SQL) formuliert werden, wodurch auch komplexere Abfragen möglich sind. Ähnliches gilt auch für ereignisbasierte Abfragen, welche beim exemplarischen Prototypen in einer an SQL angelehnten, erweiterten Sprachvariante unterstützt werden. Der Zugriff auf die von EPCglobal bzw. GS1 standardisierten Schnittstellen der Discovery- und EPCIS-Dienste im Internet der Dinge wird durch entsprechende Verarbeitungsfunktionen ermöglicht. Eine weitergehende Integration der Standards, insbesondere der OGC für das Sensor Web, erfolgte im Prototypen nicht (siehe oben).

8.3. Quantitative Evaluation

Bei der quantitativen Evaluation geht es um die Fragen, „wie gut oder schlecht ein System bestimmte Anforderungen erfüllt“ bzw. „wie sich ein System in bestimmten Situationen wirklich verhält“. Zur Beantwortung dieser Fragen wird kein konkretes Anwendungsbeispiel umgesetzt. Stattdessen werden unter „Laborbedingungen“ bestimmte Versuchsreihen durchgeführt, damit eine Wiederholbarkeit der Versuche gegeben ist. Hierbei stehen in erster Linie quantitativ messbare Größen im Vordergrund, welche durch Kennzahlen beschrieben werden können. Entsprechend ist das Ergebnis eine Darstellung verschiedener Versuchsreihen, während derer die Kennzahlen erhoben wurden. Für dieses

als *Benchmarking* bezeichnete Vorgehen wurde ein entsprechendes Werkzeug entwickelt, welches zusammen mit den Versuchsreihen sowie den jeweiligen Fragestellungen im Folgenden kurz skizziert wird.

8.3.1. Evaluationsprojekt JESPA Benchmarker

Zur Erhebung und späteren Analyse von Kennzahlen muss ein System in unterschiedlichen Versuchsreihen wiederholt unter eine bestimmte Last gesetzt werden. Für einen Vermittler bedeutet dies, dass Produzenten und Konsumenten Anfragen stellen, während deren Verarbeitung laufend Messungen bestimmter Kennzahlen durch den Vermittler durchgeführt werden, die dessen Verhalten unter einer bestimmten Last dokumentieren. Dabei helfen unterschiedliche Lastmuster, verschiedene Verhaltensweisen zu analysieren.

Für diese quantitative Evaluation wurde mit dem *JESPA Benchmarker* ein spezielles Werkzeug entwickelt, welches Produzenten und Konsumenten simuliert. Aufgabe der Produzenten ist das Versenden von (künstlich generierten) Sensordaten an den Vermittler, während die Konsumenten einfache ereignisbasierte Abfragen subskribieren und das Eintreffen von Ergebnissen dokumentieren. Der Vermittler wurde dahingehend angepasst, dass er an relevanten Punkten des Vermittlungsprozesses den zu verarbeitenden Daten Zeitstempel hinzufügt, die für die spätere Auswertung verwendet werden.

Zur Verwaltung von Produzenten und Versuchsreihen wurde eine Verwaltungskomponente entwickelt, welche Produzenten erzeugen, mit unterschiedlichen Verarbeitungs-Workflows registrieren und verschiedene Lastmuster generieren kann. Hierbei können sowohl die zeitliche Abfolge als auch der Umfang der Nachrichten variiert werden. Auch für die Verwaltung von Konsumenten wurde eine entsprechende Komponente entwickelt, mit Hilfe derer ereignisbasierte Abfragen formuliert und mit unterschiedlichen Verarbeitungs-Workflows subskribiert sowie Benchmark-Ergebnisse eingesammelt, persistiert und visualisiert werden können (siehe Abbildung 8.5).

8.3.2. Versuchsaufbau

Für den Versuchsaufbau standen zeitgemäße Linux-Arbeitsplatzcomputer (i5-3570, 3.4 GHz Quad-Core, 8 Gigabyte Arbeitsspeicher), die über ein 100mBit-Netzwerk miteinander verbunden waren, zur Verfügung. Auf diesen Computern wurde JESPA installiert und es wurden je nach Bedarf die benötigten Komponenten der Middleware gestartet. Etwaige Verfahren zum Zwischenspeichern (engl. *Caching*) von Teil-/Ergebnissen wurden sowohl in JESPA als auch in verwendeten Programmbibliotheken (z.B. dem Jadex Active Components-Rahmenwerk und der Cassandra-Datenbank) weitestgehend ausgeschaltet, um die Ergebnisse nicht zu verfälschen.

Für jeden Versuch wurden mittels des JESPA Benchmarkers Testreihen mit künstlich erzeugten Ereignissen generiert und in festgelegten zeitlichen Abständen über den Benchmark-Produzenten an die JESPA Middleware geschickt. Ein subskribierter Benchmark-Konsument empfängt die von JESPA

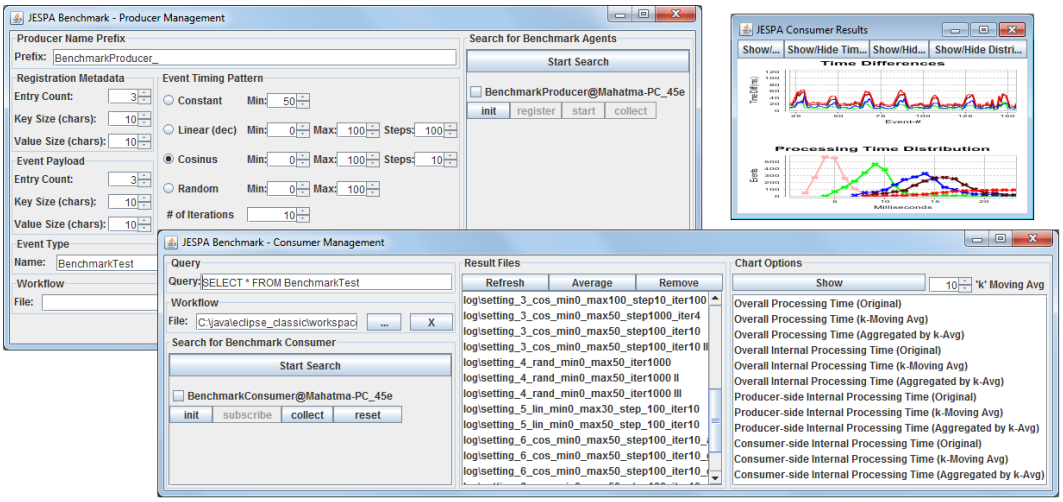


Abbildung 8.5.: Screenshot der Verwaltungskomponenten zur Steuerung von Benchmark-Produzenten und -Konsumenten

erzeugten kontextbasierten Sichten schließlich wieder. Dabei werden an insgesamt sechs Stellen des Gesamtprozesses für jede eintreffende Nachricht Zeitstempel erhoben und dieser angefügt. Hieraus lassen sich für die nachfolgende Analyse folgende Kennzahlen ableiten (vgl. auch Abbildung 8.6):

Gesamtdurchlaufzeit Die Gesamtdurchlaufzeit (engl. *Overall Processing Time*) beschreibt die Zeitspanne vom Versand einer Nachricht durch den Produzenten bis zum Empfang der kontextbasierten Sicht seitens des Konsumenten.

Interne Durchlaufzeit Die interne Durchlaufzeit (engl. *Internal Processing Time*) beschreibt die Verweildauer von Nachrichten im Vermittlungsprozess. Der initiale Zeitstempel wird beim Empfang einer Nachricht von einem Produzenten durch ein Gateway gesetzt und der finale Zeitstempel beim Versand einer kontextbasierten Sicht an den Konsumenten. Die Differenz zur Gesamtdurchlaufzeit beschreibt die Zeit für das De-/Serialisieren sowie Versenden von Nachrichten über ein Netzwerk.

Produzentenseitige Durchlaufzeit Die produzentenseitige Durchlaufzeit (engl. *Producer-side Processing Time*) beschreibt das Zeitintervall, das ausschließlich die Verarbeitung von Sensordaten nach Maßgabe der Produzenten umfasst, d.h. vom Eintreffen der Nachricht beim Gateway bis zur Übergabe der Nachricht an die Complex Event Processing-Laufzeitumgebung des Filterdienstes. Mit Hilfe dieser Kennzahl lässt sich das Systemverhalten bei einem Überhang an Produzenten im Vergleich zu Konsumenten abschätzen.

Konsumentenseitige Durchlaufzeit Die konsumentenseitige Durchlaufzeit (engl. *Consumer-side Processing Time*) ist analog zur produzentenseitigen Durchlaufzeit das Zeitintervall, das die Erstellung kontextbasierter

Sichten nach Maßgabe der Konsumenten umfasst. Das Intervall beginnt sobald die Complex Event Processing-Laufzeitumgebung ein Ergebnis generiert und endet beim Versand einer kontextbasierten Sicht durch das Gateway. Mit Hilfe dieser Kennzahl lässt sich das Systemverhalten bei einem Überhang von Konsumenten im Vergleich zu Produzenten abschätzen. Die Differenz von Gesamtdurchlaufzeit und der Summe von produzentenseitiger und konsumentenseitiger Durchlaufzeit ergibt die Verarbeitungszeit der Complex Event Processing-Laufzeitumgebung[4].

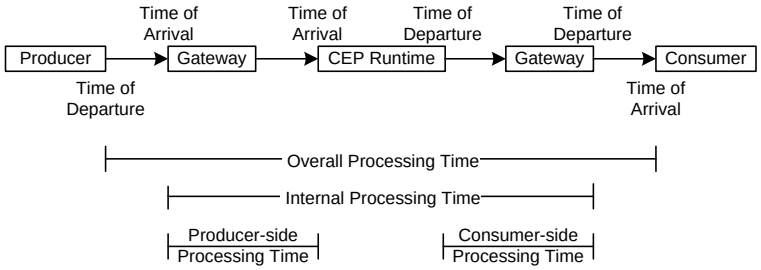


Abbildung 8.6.: Kennzahlen der Evaluationsergebnisse

Bei verteilter Ausführung des Vermittlers wurde ausschließlich die Gesamtdurchlaufzeit für die Evaluation verwendet und Produzenten und Konsumenten wurden auf demselben dedizierten Computer ausgeführt. Nur so ließ sich das Problem der zeitlich nicht synchronen Computer umgehen und aussagekräftige Ergebnisse ermitteln.

8.3.3. Ergebnisse

Nachdem nun der Versuchsaufbau beschrieben wurde, werden im Folgenden die einzelnen Versuchsreihen mit ihren jeweiligen Fragestellungen präsentiert sowie die Ergebnisse diskutiert.

Wie verhält sich das System bei rein lokaler Ausführung der Komponenten unter konstanter Last ? In dieser ersten Versuchsreihe wurden alle Komponenten des Vermittlers sowie jeweils ein Produzent und Konsument des JESPA Benchmarkers auf einem einzigen Computer installiert. Die Vermittlung von Kontextdaten erfolgte also rein lokal. Insgesamt wurden 1.000 Ereignisse im Abstand von jeweils 50ms erzeugt und vom Produzenten an das Gateway des Vermittlers übermittelt. Für jedes dieser Ereignisse hat der Vermittler den simplen Workflow des zuvor registrierten Produzenten abgerufen und ausgeführt und die Ereignisse schließlich an den Filterdienst übergeben. Der Benchmark-Konsument hat eine einfache Abfrage, die zu jedem Ereignis ein Ergebnis liefert, beim Filterdienst subskribiert. Für jedes Ergebnis wurde der Workflow des Konsumenten abgerufen und ausgeführt, bevor die resultierende

[4]Eine quantitative Evaluation der eingesetzten Complex Event Processing-Laufzeitumgebung wurde nicht durchgeführt. Hierfür sei auf [Esp10] verwiesen.

kontextbasierte Sicht schließlich an den Konsumenten übermittelt wurde. Auf diese Weise werden alle Kernkomponenten des Vermittlers angesprochen und der gesamte Prozess wird für jedes Ereignis einmal durchlaufen.

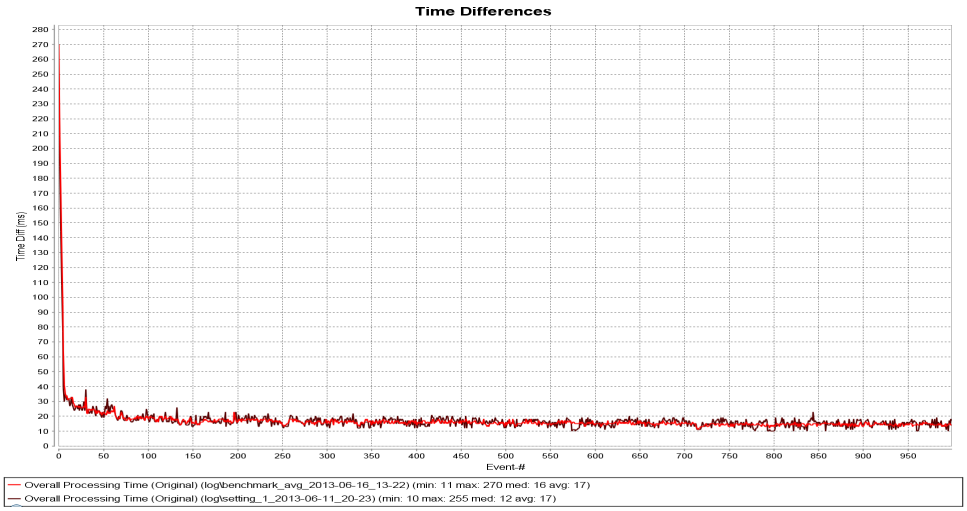


Abbildung 8.7.: Evaluation: Gesamtdurchlaufzeit bei lokaler Ausführung aller Vermittlungskomponenten

Abbildung 8.7 zeigt anhand zweier Kennlinien die Gesamtdurchlaufzeit aller 1.000 Ereignisse. Hierbei stellt die braune, etwas gezacktere Linie einen exemplarischen Versuchsdurchgang dar, während die rote, etwas glattere Linie, die Durchschnittswerte aus zehn aufeinanderfolgenden Durchgängen repräsentiert. Deutlich zu sehen ist eine kurze Initialisierungsphase, in welcher benötigte Programmklassen geladen sowie Vermittlungsdienste initial gesucht und gebunden werden. Die Durchlaufzeit stabilisiert sich fortfolgend auf einen Durchschnittswert von 17ms (Median 16ms bei dem exemplarischen Versuch und 12ms beim Durchschnitt aus 10 Durchläufen). Dies ist die Zeitspanne vom Versenden des Ereignisses durch den Produzenten bis zum Empfang des Ergebnisses beim Konsumenten.

In Abbildung 8.8 ist ein Ausschnitt der durchschnittlichen Durchlaufzeiten aus zehn aufeinanderfolgenden Durchgängen dargestellt. Die obere, rote Linie zeigt die Gesamtdurchlaufzeit, wie bereits oben erklärt. Die darunterliegende braune Linie repräsentiert die interne Verarbeitungszeit des Vermittlers (vgl. Abbildung 8.6). Die Differenz zur Gesamtdurchlaufzeit stellt die Zeit zur Übermittlung der Ereignisse vom Produzenten zum Vermittler und vom Vermittler zum Konsumenten dar, welche bei lokaler Ausführung lediglich 2ms im Durchschnitt beträgt, da die Ereignisdaten nicht de-/serialisiert werden müssen, sondern als Referenzen übergeben werden können. Die dritte, blaue Linie repräsentiert die interne Verarbeitungszeitspanne vom Eintreffen der Nachricht beim Gateway bis zur Übergabe an die Complex Event Processing-Laufzeitumgebung, d.h. die Zeit, die der Vermittler zur Verarbeitung des Ereignisses nach Maßgabe des Produzenten benötigt. Die untere, grüne Linie zeigt schließlich die Zeitspanne von der Ausgabe der Complex Event Processing-

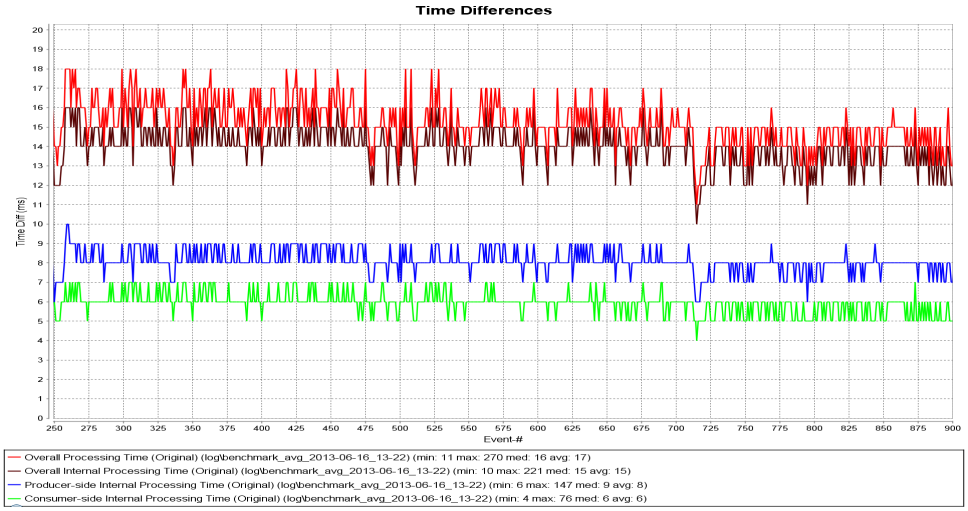


Abbildung 8.8.: Evaluation: Ausschnitt der Gesamtdurchlaufzeit aufgeteilt nach den vier Evaluationsphasen aus Abbildung 8.6 bei lokaler Ausführung aller Komponenten

Laufzeitumgebung bis zum Versand der Nachricht an den Konsumenten und repräsentiert somit die Verarbeitungszeit des Vermittlers nach Maßgabe des Konsumenten.

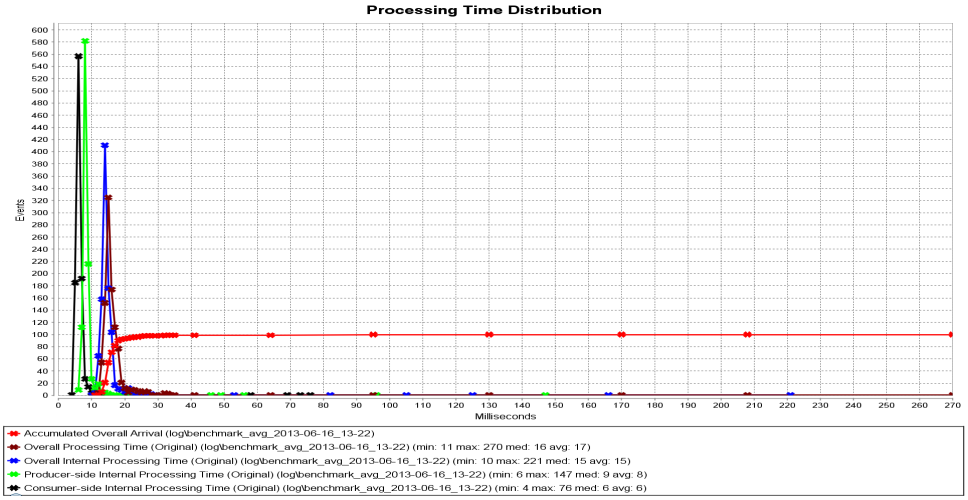


Abbildung 8.9.: Evaluation: Verteilung der Durchlaufzeiten aufgeteilt nach den vier Evaluationsphasen aus Abbildung 8.6 bei lokaler Ausführung aller Vermittlungskomponenten

Eine andere Darstellung der Durchlaufzeiten, aufgeteilt in obige vier Phasen, findet sich in dem Verteilungsgraphen in Abbildung 8.9 zu sehen. Die vier Linien repräsentieren die Anzahl an Ereignissen (Y-Achse), welche nach den auf der X-Achse aufgetragenen Millisekunden die einzelnen Phasen des Vermittlungsprozesses durchlaufen haben. Der Spitzenwert der grünen Linie beispielsweise bedeutet, dass von den 1.000 Ereignissen knapp 580 Ereignisse 9ms Millisekunden zur produzentenseitigen Verarbeitung bedurften. Der braune Punkt ganz rechts unten zeigt an, dass 1 Ereignis 270ms für den Ge-

samtdurchlauf benötigte. Auch hier ist wieder zu sehen, dass die vermittlerseitige Verarbeitungszeit für Konsumenten (schwarze Linie) geringer ist als die für Produzenten (grüne Linie) und dass die interne Verarbeitungszeit knapp geringer ist als die Gesamtdurchlaufzeit. Auch lassen sich hier die Anzahl an Ereignissen ablesen, die während der Initialisierungsphase eine längere Durchlaufzeit hatten. Die rote Linie, die sich dem Wert 100 auf der Y-Achse annähert, stellt die akkumulierte Anzahl an Ereignissen, die den Gesamtprozess innerhalb gegebener Millisekunden durchlaufen haben, in Prozent dar. Hier ist deutlich zu sehen, dass über 90% der Ereignisse weniger als 20ms für den gesamten Durchlauf benötigten. Versuchsdurchgänge mit über 300.000 Ereignissen bestätigten, dass sich die Gesamtdurchlaufzeit auch langfristig auf diesem Niveau hält.

Wie verhält sich das System bei rein lokaler Ausführung der Komponenten unter variabler Last ? Im vorigen Versuchslauf wurden die Ereignis in konstantem Abstand von 50ms generiert. Da die durchschnittliche Gesamtdurchlaufzeit lediglich 17ms beträgt, wird der Vermittler nicht kontinuierlich ausgelastet und hat ausreichend Zeit zur Verarbeitung einzelner Ereignisse. Im Folgenden werden daher unterschiedliche Lastmuster betrachtet, welche den Vermittler kurzzeitig überlasten. Hierdurch kann das Verhalten des Vermittlers bei zu hoher Last und die jeweiligen Auswirkungen auf die Gesamtdurchlaufzeit von Ereignissen veranschaulicht werden.

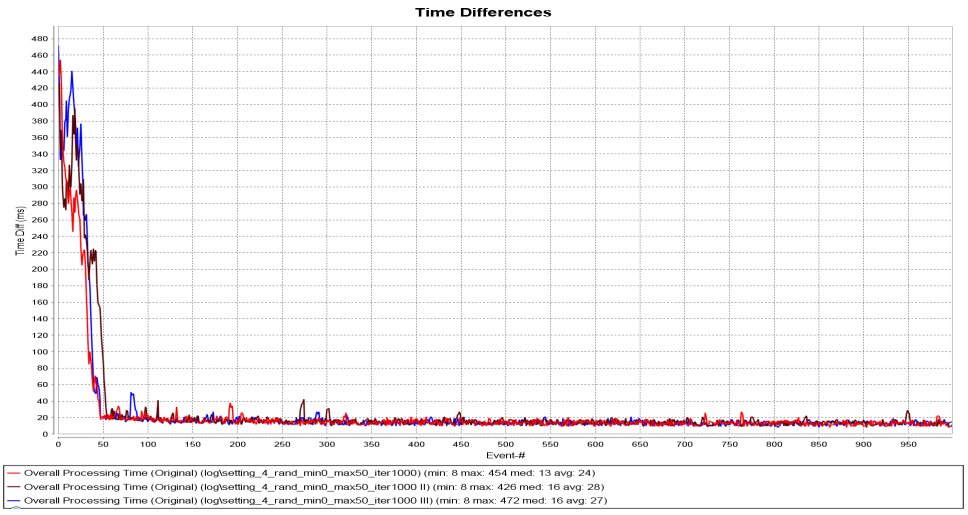


Abbildung 8.10.: Evaluation: Gesamtdurchlaufzeit bei lokaler Ausführung aller Vermittlungskomponenten und zufällig generierter Last

Abbildung 8.10 zeigt in drei Versuchsreihen die Gesamtdurchlaufzeiten von 1.000 Ereignissen, die in einem zufälligen Abstand zwischen 0 und 50 ms generiert wurden. Im Vergleich mit den Versuchen unter konstanter Last ist hier lediglich die längere Initialisierungsphase auffällig, welche für eine Gesamtdurchlaufzeit von bis zu 472ms für die ersten 40-50 Nachrichten verantwortlich sind.

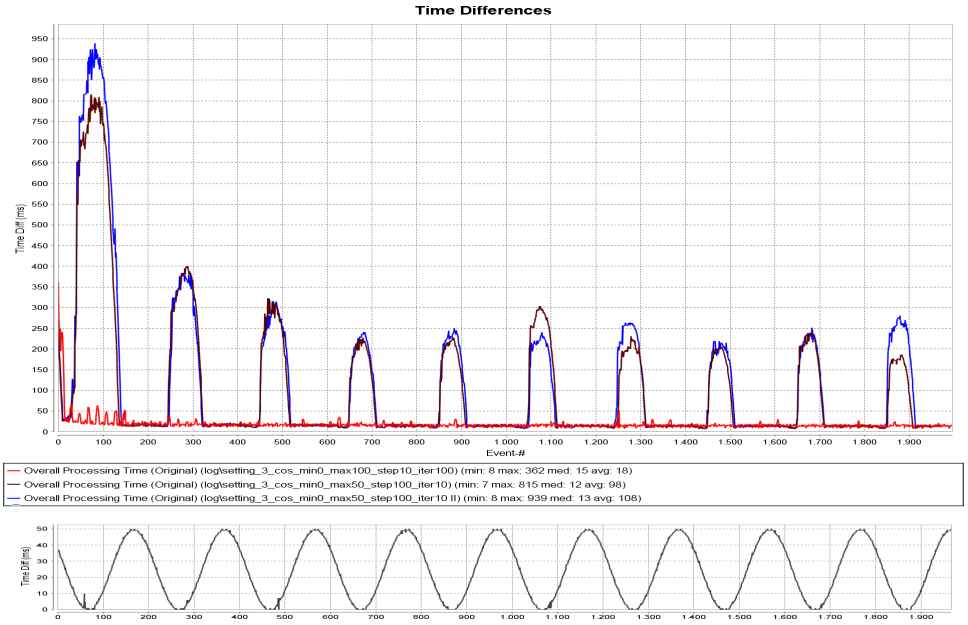


Abbildung 8.11.: Evaluation: Gesamtdurchlaufzeit bei lokaler Ausführung aller Vermittlungskomponenten und Kosinus-förmig generierter Last

In Abbildung 8.11 ist das Ergebnis (oberer Teil der Abbildung) eines ursprünglich Kosinus-förmigen Last-Eingangsmusters (unterer Teil der Abbildung) dargestellt. Die ersten zwei Ereignisse werden mit einer initial hohen Latenz generiert, dann verringert sich der Abstand sukzessive auf 0ms, um schließlich wieder auf das Ursprungsniveau anzusteigen. Die rote Linie repräsentiert 100 solcher Kosinus-Folgen mit jeweils ca. 20 Ereignissen und einem initialen Abstand von 100ms zwischen den Ereignissen. Die blaue und braune Linie zeigen jeweils 10 Folgen mit je ca. 200 Ereignissen und einem initialen Abstand von nur 50ms.

Letztere sind besonders interessant, da hierbei deutlich die Überlastung des Vermittlers abzulesen ist, der kurz aufeinanderfolgende Nachrichten zunächst in Warteschlangen einreihen muss, was die Gesamtdurchlaufzeit entsprechend erhöht. Zu berücksichtigen ist hierbei allerdings, dass der erste Zeitstempel bereits vor der Serialisierung und dem Versand eines Ereignisses gesetzt wird, sodass bei eng aufeinanderfolgenden Nachrichten auch der Produzent überlastet wird und somit die Überlastspitzen nicht allein dem Vermittler zuzurechnen sind. Die rein interne Durchlaufzeit liegt in den Lastspitzen wenige Millisekunden unter der Gesamtdurchlaufzeit.

Welchen Einfluss haben unterschiedlich große Nutzdatenvolumina ? In den bisherigen Versuchen wurden Ereignisse mit einer relativ geringen Menge an Nutzdaten (engl. *Payload*)[5] erzeugt. Unter der vereinfachten Annah-

[5]Als Nutzdaten werden in diesem Fall ausschließlich die Schlüssel/Wert-Paare mit Sensormessungen sowie die Kennung des Sensors (ca. 30 Zeichen), die Kennung des Datenstroms (ca. 15 Zeichen) sowie ein Zeitstempel (4 Byte) angesehen.

me, dass jedes Zeichen mit 16 Bit kodiert wird, betrug die Netto-Datenmenge ungefähr 150 Bytes[6]. In diesem Versuch wurde die Nettodatenmenge in 4 Durchgängen sukzessive erhöht.

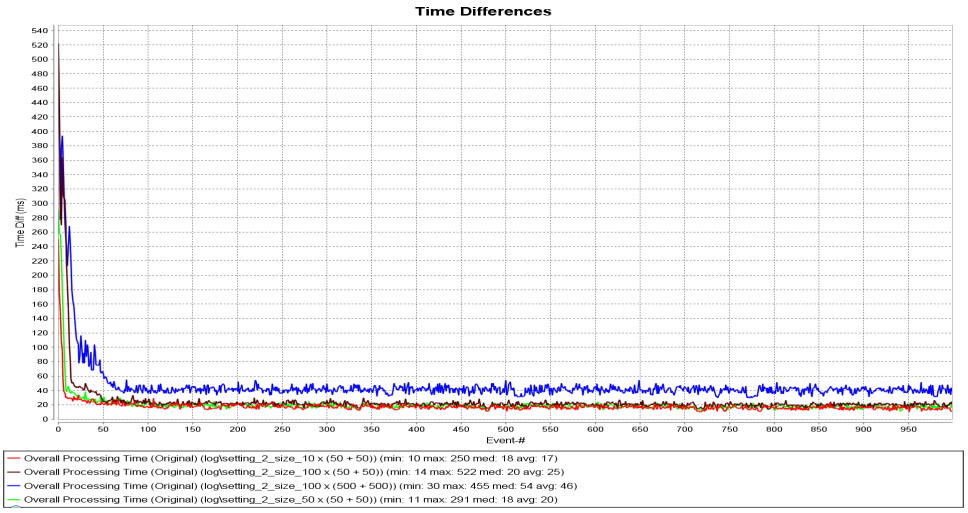


Abbildung 8.12.: Evaluation: Gesamtdurchlaufzeit bei lokaler Ausführung aller Vermittlungskomponenten mit unterschiedlichen Nutzdatenvolumina

Abbildung 8.12 zeigt den Einfluss der Menge an Nutzdaten auf die Gesamtdurchlaufzeit bei lokaler Ausführung aller Komponenten. Das bedeutet, Latenzen durch De-/Serialisierung der Datenobjekte sowie die Netzwerkkommunikation können außen vor gelassen werden. Auch wurden die Sensordaten nicht persistiert, entsprechend ist das Ergebnis auch nicht von der Wahl eines Datenhaltungssystems abhängig. Die rote Linie repräsentiert einen exemplarischen Durchgang mit 10 Schlüssel/Wert-Paaren a 100 Zeichen (insgesamt ca. 2.000 Byte). Die grüne Linie zeigt einen Durchlauf mit 50 Schlüssel/Wert-Paaren a 100 Zeichen (insgesamt ca. 10.000 Byte) und die braune Linie einen Durchlauf mit 100 Schlüssel/Wert-Paaren a 100 Zeichen (insgesamt ca. 20.000 Byte). Die obere, blaue Linie stellt einen Durchlauf mit 100 Schlüssel/Wert-Paaren a 1.000 Zeichen (insgesamt ca. 200.000 Byte) dar.

Wie in der Abbildung zu sehen, steigt die Gesamtdurchlaufzeit bei Nachrichten mit mehr als mindestens 20.000 Byte Nutzdaten langsam an. Bei Versuchen mit noch größeren Datenmengen (bis zu 2.000.000 Byte) tritt eine deutliche Überlastung des Vermittlers auf, welcher Datenpakete, die in konstantem 50ms-Abstand eintreffen, zunächst noch zeitnah (ca. 5.000ms) ausliefern kann, aber linear zunehmend länger braucht, bis nach 500 Ereignissen bereits eine Gesamtdurchlaufzeit von 120 Sekunden und mehr erreicht wird.

Welchen Einfluss haben unterschiedliche Datenhaltungssysteme ? Da die Schnittstellen der Daten- und Metadatenverwaltung des Vermittlers un-

[6]Je nach verwendeter Nachrichtenkodierung, z.B. Binärformat oder XML, sowie unter Berücksichtigung weiterer Metainformationen kann die Bruttodatenmenge, die wirklich versendet wird, teils erheblich abweichen.

abhängig vom konkret verwendeten Datenhaltungssystem sind, lassen sich unterschiedliche Systeme einsetzen. Für die exemplarische Implementierung wurden zwei verschiedene Varianten umgesetzt: eine schnelle In-Memory-Datenbank basierend auf Hash-Tabellen und *Apache Cassandra* als Beispiel einer schwergewichtigen verteilten Datenbank. Erstere dient für den rein lokalen Einsatz, während Letztere die Daten beliebig im Netz verteilen kann. Sofern einzelne Komponenten (z.B. der Identitätsdienst oder die Metadatenbereitstellung) nicht mehrfach im Netz instanziiert werden, hängt die Wahl einer geeigneten Datenhaltung lediglich von den Anforderungen der Anwendungen, den zur Verfügung stehenden Ressourcen und der Frage ab, ob die Daten in einem Persistenzspeicher festgeschrieben werden müssen. In diesem Versuchsaufbau wurde der Einfluss eines Datenhaltungssystems auf die Gesamtdurchlaufzeit bei rein lokaler Ausführung aller Vermittlungskomponenten getestet.

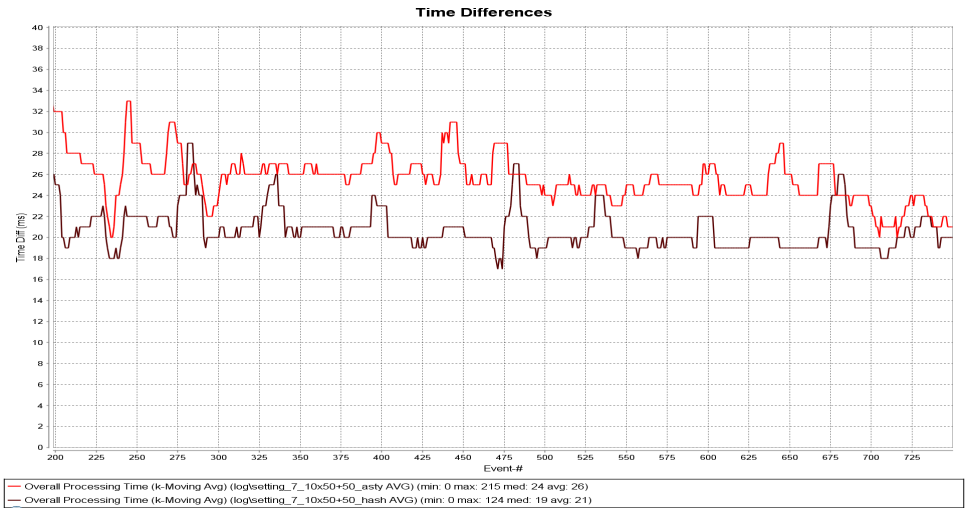


Abbildung 8.13.: Evaluation: Gesamtdurchlaufzeit bei lokaler Ausführung aller Vermittlungskomponenten und verschiedenen Datenhaltungssystemen

Die obere, rote Linie in Abbildung 8.13 repräsentiert einen typischen Ausschnitt der durchschnittlichen Gesamtdurchlaufzeit aus zwei Einzelversuchen unter Verwendung der Apache Cassandra-Datenbank. Die untere, braune Linie repräsentiert den Durchschnitt aus zwei Durchläufen mit der In-Memory-Datenbank. Um die Unterschiede deutlicher hervorzuheben, wurden beide Linie mit einem Tiefpassfilter (gleitender Mittelwert mit einem Fenster von 10 Werten) nachbearbeitet. Deutlich zu sehen ist, dass die Wahl eines Datenhaltungssystems deutlichen Einfluss auf die Durchlaufzeit hat, wobei die Gesamtdurchlaufzeit mit der In-Memory-Datenbank mit durchschnittlich 21ms ca. 20-25% Prozent schneller ist, als Apache Cassandra mit durchschnittlich 26ms, dafür natürlich aber nur einen minimalen (für den Vermittler jedoch in vielen Fällen ausreichenden) Funktionsumfang bietet.

Kann die Leistung des Gesamtsystems durch Bereitstellen mehrerer Gateways positiv beeinflusst werden ? Für alle bisher beschriebenen Versuche wurden Produzent, Konsument und alle Vermittlungskomponenten auf demselben Computer ausgeführt. In diesem und allen folgenden Versuchen werden die einzelnen Komponenten nun in einem lokalen Netzwerk über mehrere Computer verteilt. Dabei werden Produzent und Konsument stets auf demselben Computer ausgeführt, um das Problem der Zeitsynchronisierung bei der Erhebung der Gesamtdurchlaufzeit zu umgehen.

Im folgenden Versuch soll zunächst die Lastverteilung über die Gateways untersucht werden. Da diese keinen eigenen Zustand besitzen und daher nur ganz geringe Anforderungen an die Ausführungsumgebung und ihre Ressourcen stellen, bieten sie sich zur Lastverteilung am Rand des Gesamtsystems an. In einer ersten Versuchsreihe wurden hierfür zunächst alle Komponenten des Vermittlungsprozesses, d.h. Filter-, Subskriptions-, Identitäts- und High-level-Verarbeitungsdienst, Metadatenbereitstellung, Registrierung und die Workflow-Verwaltung, auf einem zweiten Computer ausgeführt. In den nachfolgenden Versuchsreihen wurde dann das Gateway sowie von diesem erzeugte Low-level-Verarbeitungsdienste zur Aufbereitung der Sensordaten nach Maßgabe des Produzenten auf 1-3 weitere Computer ausgelagert. Abbildung 8.14 zeigt den finalen Versuchsaufbau mit 3 Gateways.

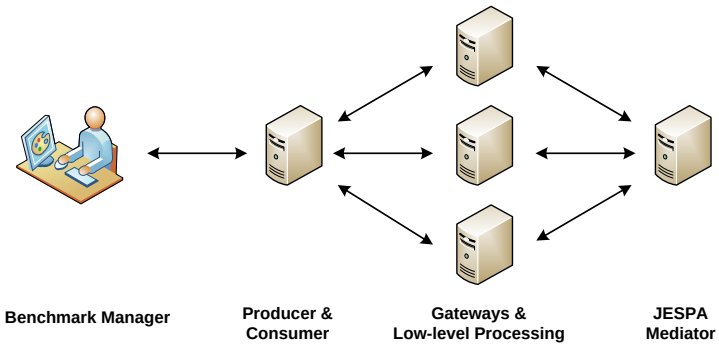


Abbildung 8.14.: Evaluation: Versuchsaufbau mit 3 Gateways

In Abbildung 8.15 sind die Ergebnisse der Versuchsreihe dargestellt: Die obere, rote Linie repräsentiert die Gesamtdurchlaufzeit bei Kosinus-förmigem Lastmuster und bei Verwendung lediglich eines Computers, auf dem sowohl das Gateway als auch alle anderen Vermittlungskomponenten ausgeführt wurden. Die darunterliegende braune Linie zeigt den Verlauf unter Hinzunahme eines dedizierten Gateway-Computers. Bei den fast identisch verlaufenen blauen und grünen Linien wurden 2 bzw. 3 Gateway-Computer im Versuchsaufbau verwendet. Abbildung 8.16 verdeutlicht die Ergebnisse nochmals mittels eines normierten Verteilungsgraphen. Allen Linien liegen je vier exemplarische Testläufe zugrunde, aus denen jeweils Durchschnittswerte gebildet wurden.

Das Ergebnis zeigt, dass das Hinzufügen weiterer Gateways ein probates Mittel darstellt, um prinzipiell Lastspitzen zu entschärfen. Je nach Systemaufbau, das heißt der Verteilung anderer Vermittlungskomponenten, gibt es

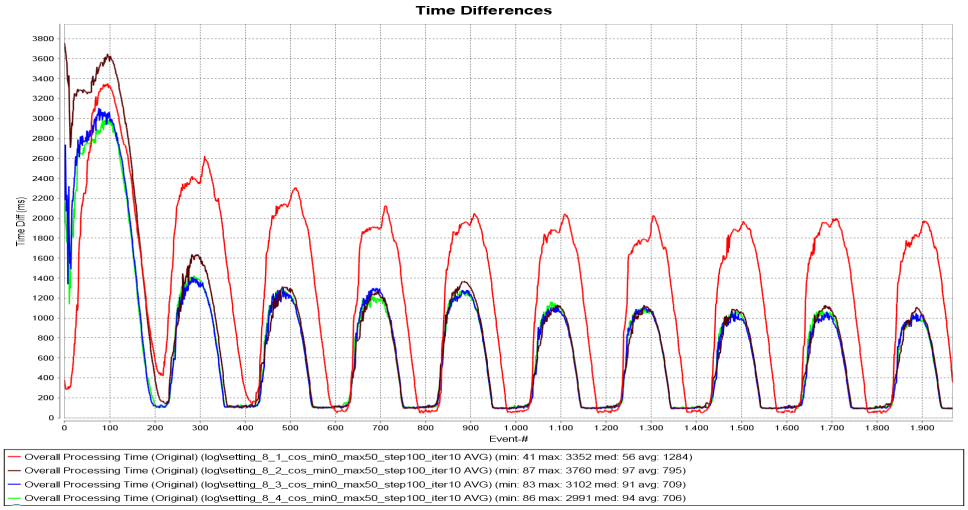


Abbildung 8.15.: Evaluation: Einfluss der Anzahl an Gateways auf die Gesamtdurchlaufzeit

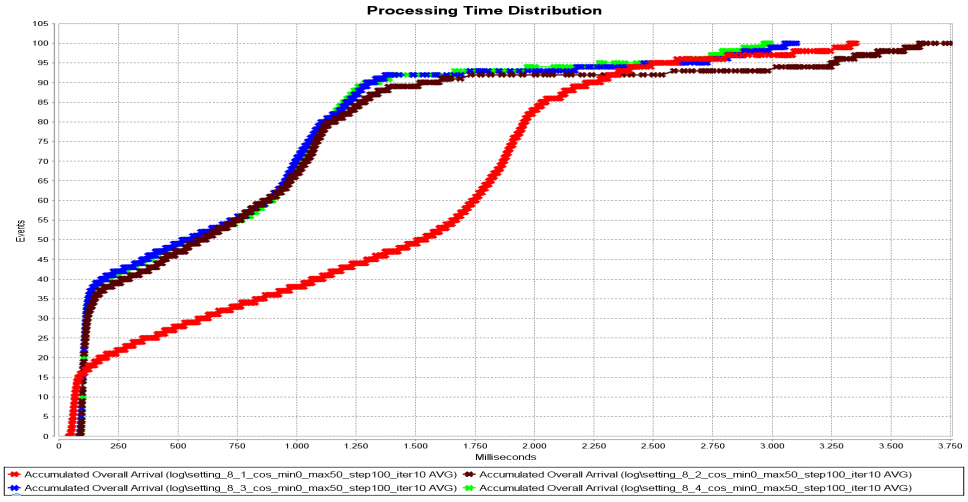


Abbildung 8.16.: Evaluation: Einfluss der Anzahl an Gateways auf die Gesamtdurchlaufzeit (Verteilungsgraph)

jedoch eine Grenze, in diesem Versuch 2 dedizierte Gateways, ab der keine wesentliche Verbesserung der Gesamtdurchlaufzeit zu erzielen ist, da die Hauptlast der Vermittlung weiterhin auf einem Computer liegt.

Kann die Leistung des Gesamtsystems durch Replikation aller Dienste positiv beeinflusst werden ? Während in vorigem Versuch nur die Gateways mehrfach repliziert wurden und hierbei eine maximal sinnvolle, obere Grenze festgestellt wurde, ab der zusätzliche Gateways die Überlastung anderer Kernkomponenten nicht weiter kompensieren konnten, soll in diesem Versuch gezeigt werden, dass durch Replikation aller Komponenten die Gesamtdurchlaufzeit weiter verringert werden kann.

Der Versuchsaufbau orientiert sich an dem Aufbau aus vorigem Versuch, doch anstatt nur das Gateway auf einzelnen dedizierten Computern auszuführen, werden in diesem Versuch alle Vermittlungskomponenten (außer dem Filterdienst, welcher nur einmal instanziiert werden darf) auf jedem beteiligten Computer ausgeführt. Auch hierbei wurden bis zu 5 Computer verwendet: auf einem Computer liefen nur Produzent und Konsument und auf allen anderen komplette JESPA-Installationen. Bei jedem Dienstaufruf wurde eine zufällige Instanz ausgewählt und alle Instanzen beruhten auf einer verteilten Cassandra-Datenbank mit einem Replikationsfaktor von 1 und abgeschaltetem Caching, um die maximale Auslastung zu erreichen.

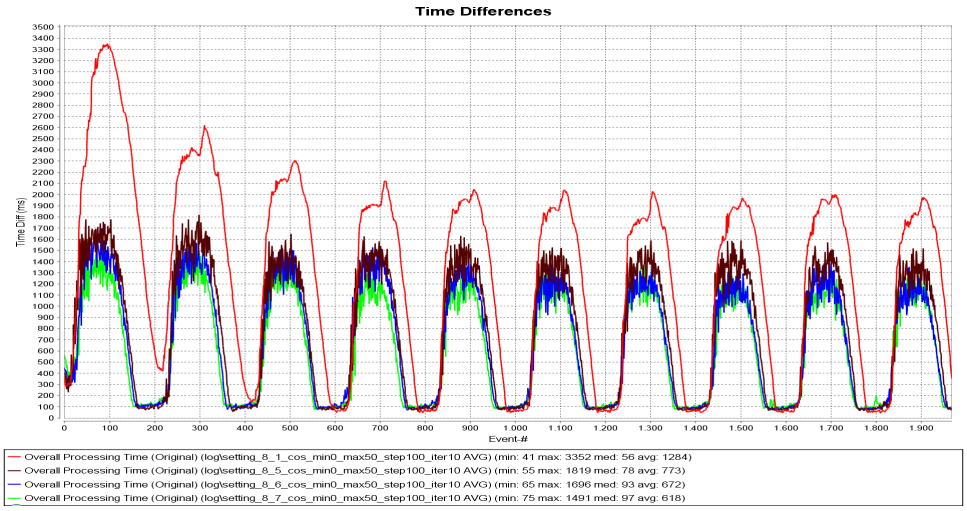


Abbildung 8.17.: Evaluation: Einfluss der Anzahl kompletter Installationen auf die Gesamtdurchlaufzeit

Abbildungen 8.17 und 8.18 zeigen, wie im vorigen Versuch auch, die einzelnen Durchläufe mit unterschiedlicher Anzahl an beteiligten Computern sowie die Verteilung der akkumulierten Gesamtdurchlaufzeiten auf einer Skala bis 100 %. Hier ist noch deutlicher zu erkennen, dass durch Hinzunahme weiterer Computer die Lastspitzen entschärft werden können. Hieraus kann geschlussfolgert werden, dass das System prinzipiell gut skaliert.

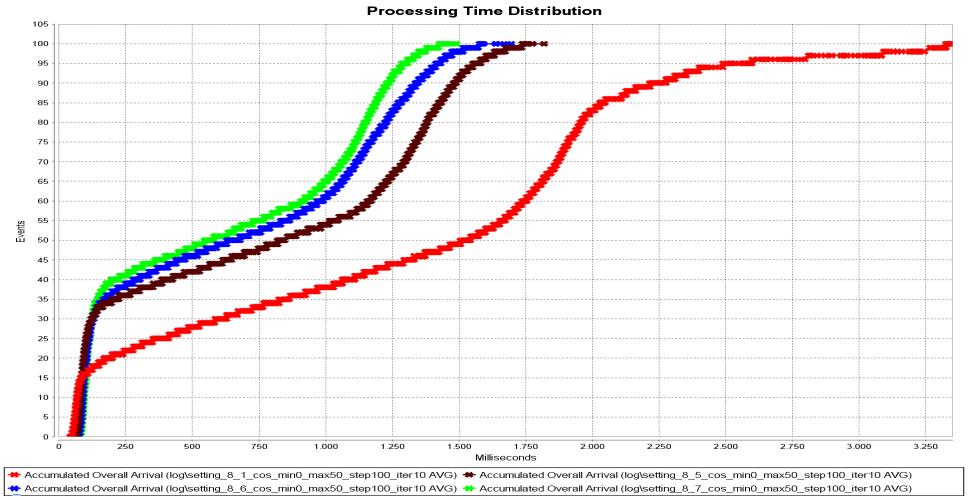


Abbildung 8.18.: Evaluation: Einfluss der Anzahl kompletter Installationen auf die Gesamtdurchlaufzeit (Verteilungsgraph)

Bei ohnehin geringer Last erhöht sich die durchschnittliche Gesamtdurchlaufzeit jedoch im Vergleich zur Nutzung eines einzelnen Computers, da der Austausch von Nachrichten durch De-/Serialisierung und Netzwerkkommunikation einen Mehraufwand mit sich bringt. Hier ist allerdings eine obere Schranke hinsichtlich der Durchlaufzeit zu erwarten, die erreicht wird, sofern eine Nachricht für jeden Verarbeitungsschritt zu einem anderen Computer geschickt werden muss.

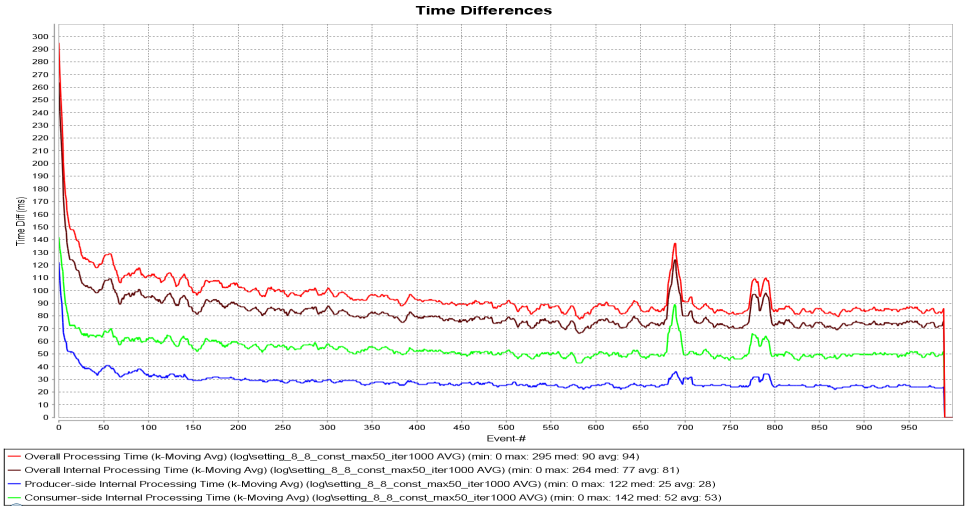


Abbildung 8.19.: Evaluation: Einfluss der Anzahl kompletter Installationen auf die Gesamtdurchlaufzeit bei geringer Last

Abbildung 8.19 zeigt das Verhalten des obigen Versuchsaufbaus bei einer geringen, konstanten Last (1.000 Ereignisse mit jeweils 50ms Abstand) aufgeteilt in die vier Phasen der Verarbeitung (die Linien zeigen Durchschnittswerte aus je 4 exemplarischen Durchgängen und wurden mittels eines Tief-

passfilter geglättet). Mit einer durchschnittlichen Gesamtdurchlaufzeit von 94ms ist die Verarbeitung etwas langsamer. Die Differenz zur internen Durchlaufzeit von durchschnittlich 13ms deutet den Einfluss der De-/Serialisierung und Netzwerkkommunikation an, wobei 13ms zwei Kommunikationsakte repräsentieren (vom Produzenten zum Gateway und vom Gateway zum Konsumenten). Auch ist hier eine höhere konsumentenseitige Verarbeitungszeit im Vergleich zur rein lokalen Ausführung (vgl. Abbildung 8.8), bei der die produzentenseitige Verarbeitungszeit überwiegt, zu sehen, was durch die größere Anzahl an Dienstaufrufen bei konsumentenseitiger Verarbeitung bedingt ist.

Wie gut kompensiert das System den Ausfall notwendiger Komponenten ?
Im vorigen Versuch konnte gezeigt werden, dass das System prinzipiell gut skaliert. In diesem letzten Versuch soll die Reaktion des Systems auf den Ausfall von Komponenten untersucht werden, um Aussagen über die Robustheit treffen zu können.

Der Versuchsaufbau besteht aus einem Computer, auf dem Produzent und Konsument laufen, sowie einem Computer auf dem alle Vermittlungsdienste laufen. Insgesamt wurden 120 Ereignisse im Abstand von 1 Sekunde an den Vermittler geschickt. Im Laufe dieser 2 Minuten wurde der Identitätsdienst, welcher sowohl bei der produzenten- als auch bei der konsumentenseitigen Verarbeitung unverzichtbar ist, mehrere Male gestoppt und neu gestartet.

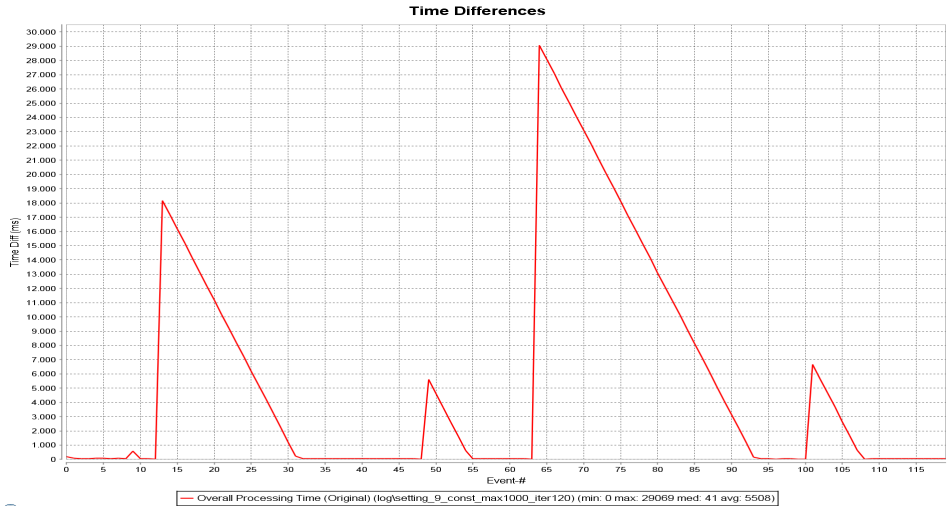


Abbildung 8.20.: Evaluation: Verhalten bei temporärem Ausfall einer obligatorischen Komponente

Abbildung 8.20 zeigt einen exemplarischen Durchlauf, in dem der Identitätsdienst insgesamt 4 mal gestoppt und nach unterschiedlich langen Zeiträumen neu instanziiert wurde. Der erste Ausfall dauerte ca. 20 Sekunden und beginnt bei Sekunde 12. Hier ist zu sehen, dass das entsprechende Ereignis trotz Ausfall zugestellt werden konnte, allerdings erst knapp 20 Sekunden später als der Dienst wieder verfügbar war. Das Ereignis bei Sekunde 13 wurde 19 Sekunden später zugestellt etc. Hier wird ersichtlich, dass der

Vermittler bei einer fehlgeschlagenen Zustellung die Nachricht in eine Warteschlange einreiht und nach kurzer Wartezeit (im Beispiel 1 Sekunde) einen erneuten Versuch unternimmt.

In diesem Versuch wurde der Identitätsdienst manuell gestoppt und neu instanziiert. Es ist allerdings ohne Weiteres möglich dies unter Verwendung des *JESPA Deployment Monitors* (vgl. Abschnitt 7.2.2.6) zu automatisieren.

8.4. Zusammenfassung & Bewertung

Die in diesem Kapitel durchgeführte Evaluation des Konzepts zur Vermittlung kontextbasierter Sichten und der exemplarischen, prototypischen Implementierung eines Vermittlers sollte deren Stärken und Schwächen aufdecken und zeigen, ob die identifizierten funktionalen und nicht-funktionalen Anforderungen erfüllt werden konnten. Durch eine qualitative Evaluation mit den beiden Evaluationsprojekten *MagicTracker* und *AEPIO/SOLA* wurde anhand der Fragen „was das System leistet“ und „welches Verhalten zu erwarten ist“ eine deskriptive Analyse des Systems hinsichtlich des Anforderungskatalogs durchgeführt.

Das Ergebnis der qualitativen Analyse zeigt, dass konzeptionell alle Anforderungen prinzipiell erfüllt werden können, da dass Konzept auf Basis des Anforderungskatalogs entwickelt wurde. Die exemplarische, prototypische Implementierung hingegen erfüllt zwar alle obligatorischen funktionalen Anforderungen, eine Reihe optionaler Anforderungen wurden jedoch nicht umgesetzt, da sie für die grundlegende Funktion eines Vermittlers, die es zu evaluieren galt, unerheblich sind. Weiterer Entwicklungs- und auch Forschungsbedarf konnten insbesondere bei der Integration semantischer Abbildungen und Ableitungen, der ad-hoc und multimedialen Abfragen, der Prozess- und Auftragsverwaltung sowie der Integration existierender Standards aufgedeckt werden. Alle anderen funktionalen und nicht-funktionalen Anforderungen konnten adäquat umgesetzt werden, wobei insbesondere auch die spezifischen Charakteristika mobiler, ubiquitärer Systeme Berücksichtigung fanden.

Die im Anschluss vorgestellte quantitative Evaluation sollte zeigen, wie gut oder schlecht einzelne Anforderungen im praktischen Betrieb wirklich erfüllt werden. Im Gegensatz zur deskriptiven Analyse der qualitativen Evaluation wurden hier anhand verschiedener Fragestellungen konkrete Versuchsreihen mit der prototypischen Implementierung und einem speziell entwickelten Benchmark-Werkzeug unter Laborbedingungen durchgeführt, um belastbare Aussagen treffen zu können. Die Fragestellungen betrafen dabei das Verhalten der Implementierung auf unterschiedliche Lastmuster, Auswirkungen unterschiedlicher Nutzdatenvolumina und Datenhaltungssysteme, Skalierbarkeitseffekte durch Verteilung und Replikation von Komponenten im Netzwerk sowie das Systemverhalten bei Ausfall einzelner Dienste.

Das Ergebnis dieser quantitativen Analyse zeigt, dass die Leistung der exemplarischen, prototypischen Implementierung wesentlich vom Systemaufbau abhängt. Bei rein lokaler Ausführung können Ereignisse ausgesprochen effek-

tiv verarbeitet werden. Sobald die Last zu groß für eine rein lokale Ausführung ist und einzelne Vermittlungskomponenten im Netz verteilt werden müssen, nimmt die Verarbeitungszeit durch die Nachrichtenkommunikation erwartungsgemäß zu. Hierbei konnte gezeigt werden, dass die Leistung durch Replikation von Komponenten gesteigert werden kann, wobei je nach angelegter Last durchaus nicht alle Lastspitzen gänzlich aufgefangen werden können. Für die Evaluation wurden jedoch etwaige Optimierungsmaßnahmen wie das Zwischenspeichern von Ergebnissen oder das Replizieren von Einträgen der verteilten Datenhaltung weitestgehend inaktiv gesetzt, um die Ergebnisse nicht zu beeinträchtigen, sodass im praktischen Einsatz durchaus Optimierungspotential durch Substitution von Rechenzeit gegen Speicherbedarf besteht. Zudem sind die gemessenen Reaktionszeiten eines Vermittlers für ein Groß der Anwendungsbereiche bereits gänzlich ausreichend.

Auch konnte gezeigt werden, dass der prototypische Vermittler robust gegenüber einem Ausfall notwendiger Systemkomponenten ist, da er Ausfälle erkennen und mit Hilfe entsprechender Werkzeuge auch automatisch kompensieren kann.

9. Schlussbetrachtung

In der folgenden Schlussbetrachtung sollen die Ergebnisse dieser Arbeit zunächst zusammengefasst und anschließend diskutiert werden. Hierbei soll insbesondere auch der wissenschaftliche Beitrag dieser Arbeit eingeordnet werden. Das Kapitel schließt mit einer Zusammenfassung der gesamten Arbeit sowie einem Ausblick auf mögliche Erweiterungen des Konzepts der Vermittlung kontextbasierter Sichten sowie der prototypischen Implementierung eines Vermittlers.

9.1. Diskussion der Ergebnisse

Motiviert durch die Feststellung, dass der Entwurf und die Entwicklung kontextbasierter Anwendungen auf einer Reihe von Annahmen hinsichtlich der zu nutzenden Informationen, der Geräte und der Infrastruktur sowie der involvierten Teilnehmer aufbauen und entsprechende Anforderungen stellen, widmet sich diese Arbeit mit ihrer Forschungsfrage der Aufgabe, wie sich individuell aufbereitete, kontextbasierte Sichten zwischen heterogenen Produzenten und Konsumenten in mobilen, ubiquitären Systemen vermitteln lassen. Ziele dabei sind, die Entwicklung kontextbasierter Anwendungen insgesamt zu vereinfachen, die Zeit zur Umsetzung zu verkürzen, Mashup-ähnliche Anwendungen zu fördern, die dynamische Adaption von Anwendungen an sich ändernde Umstände zur Laufzeit zu ermöglichen, den Nutzungsskopus von Kontextinformationen zu vergrößern und letztlich mobile Geräte zu entlasten.

Das Ergebnis und der wesentliche Beitrag dieser Arbeit ist die Konzeption eines Vermittlungsprozesses, welcher Konsumenten (z.B. Menschen oder kontextbasierte Anwendungen) eine individuell aufbereitete Sicht auf den Kontext beliebiger Entitäten zur Verfügung stellt, sofern dieser durch Produzenten (z.B. Sensoren) beobachtbar ist. Eine solche Sicht entlastet die Konsumenten dahingehend, dass sie keinen Aufwand für die Erzeugung dieser Sichten erbringen müssen, sondern die Informationen direkt für ihren eigentlich intendierten Anwendungszweck nutzen können. Auf diese Weise müssen sich Konsumenten nicht an die heterogenen Produzenten und die von diesen verwendeten Datenformate, Kommunikationsprotokolle und Technologien anpassen. Hierdurch wird nicht nur auch ressourcenbeschränkten Geräten die Nutzung von Kontextinformationen ermöglicht, sondern es werden auch die Benutzer und Entwickler kontextbasierter Anwendungen entlastet, da die Verarbeitungslogik und -last in die Infrastruktur verschoben werden kann und somit die Wiederverwendbarkeit von Verarbeitungsfunktionen ermöglicht wird.

Aufbauend auf einem Anforderungskatalog, der einerseits aus der Sicht konkreter Anwendungsbeispiele, andererseits aus den technischen Möglichkeiten

und der Vision des Ubiquitous Computing resultierte, wurde ein Vermittlungsprozess konzipiert und in eine Reihe von Aufgabenbereichen, mit jeweils spezifischen Teilaufgaben untergliedert. Dieser Prozess, bzw. ein den Prozess ausführender Vermittler, hebt sich von verwandten Arbeiten dahingehend ab, dass er auf die Verwaltung, Verarbeitung und Aufbereitung von Kontextdaten fokussiert. Dazu stellt er die Daten etwaigen Konsumenten nicht nur in Rohform zur Verfügung, sondern erzeugt darüber hinaus auch den individuellen Erfordernissen der Konsumenten entsprechend geeignete Repräsentationen und kann hierfür beliebige Verarbeitungsschritte ausführen. Als Beweis der Umsetzbarkeit des Konzepts wurden Kernfunktionen aller identifizierten Aufgabenbereiche des Prozesses sowie eine Reihe optionaler Funktionalitäten im Rahmen einer exemplarischen, prototypischen Implementierung umgesetzt. Hier gehören unter anderem das Registrieren von Produzenten und Subskribieren von Konsumenten, die Verwaltung und das Abfragen von Meta- und Kontextdaten, die Detektion komplexer Kontextereignisse sowie die Aufbereitung von Kontextdaten nach Maßgabe von Produzenten und Konsumenten. Eine qualitative Evaluation des Prototypen in zwei Forschungsprojekten sowie eine quantitative Evaluation im Rahmen konstruierter Versuchsreihen zeigten, dass das Konzept in verschiedenen Anwendungsdomänen mit ihren jeweils eigenen Anforderungen prinzipiell umsetzbar ist und auch dahingehend einen Mehrwert bringt, dass nur noch geringe Anforderungen an Produzenten und Konsumenten gestellt werden müssen, Kontextdaten von beliebigen, unter Umständen heterogenen und global verteilten Produzenten bezogen werden können, eine lose Kopplung sowie eine n:m-Beziehung zwischen Produzenten und Konsumenten erreicht und insgesamt die Entwicklung kontextbasierter Anwendungen vereinfacht und effizienter werden kann sowie neue Arten Mashup-ähnlicher Anwendungen ermöglicht werden.

Jedoch zeigte die Evaluation auch noch bestehende Schwächen des Ansatzes auf. Hierzu gehört insbesondere die Beobachtung, dass die Nutzung des Systems durch Entwickler und Anwender trotz unterstützender Werkzeuge noch immer nicht trivial und intuitiv ist. Gerade das Formulieren von Abfragen stellt ein Benutzungsproblem dar und erfordert in Teilen noch die Kenntnis der verwendeten Technologien, z.B. der Complex Event Processing-Laufzeitumgebung, welche ereignisbasierte Abfragen verarbeitet. Auch das Erstellen BPMN-basierter Verarbeitungs-Workflows erfordert zumindest die Kenntnis der entsprechenden Modellierungssprache. Beide Probleme treten jedoch auch in anderen Bereichen der Informatik auf und sind nicht spezifisch für den Vermittlungsprozess. Lösungsansätze existieren bereits in Form (meist) kommerzieller Werkzeuge, die jedoch im Rahmen dieser Arbeit nicht weiter untersucht wurden. Auch die Nutzung semantischer Technologien sowie multimedialer Abfragen könnten hier einen Mehrwert bringen, wurden jedoch nicht in den Prototypen integriert und somit nicht evaluiert.

Weiterer Optimierungsbedarf besteht auch bei der Nutzung von Verarbeitungsressourcen. Der Vermittlungsprozess sieht zwar die (dynamische) Auswahl von Ressourcen (z.B. Verarbeitungsdiensten) basierend auf Metadaten-

verzeichnissen prinzipiell vor, nutzt diese in der prototypischen Realisierung jedoch eher suboptimal, weshalb die bei den verteilten Versuchsaufbauten der quantitativen Evaluation aufgezeigten Skalierbarkeitseigenschaften weniger deutlich hervortraten als erwartet. Hier müssen effizientere Maßnahmen zum Umgang mit Ressourcen und zur bedarfsgerechten Auswahl getroffen werden.

Eine weitere, generelle Herausforderung stellt das Thema Sicherheit dar. Kontextdaten können sensible Informationen repräsentieren, deren Zugriff auf autorisierte Nutzer eingeschränkt werden können muss. Ähnliches gilt auch für Verarbeitungsdienste. Im Rahmen dieser Arbeit wurden zwar einfache Möglichkeiten zum Schutz vor unberechtigten Dienstaufrufen vorgestellt, allerdings fehlt es an einem umfassenden Konzept zum Schutz der Daten. Für einen praktischen Einsatz ist dies unumgänglich.

Eine weitere Herausforderung für den Einsatz eines solchen Vermittlers in der Praxis stellt die kritische Menge initial verfügbarer Produzenten dar. Konsumenten können ihre Abfragen nur für bereits existierende Produzenten (bzw. Datenströme) subskribieren. Es ist also nicht möglich, sich über eine bevorstehende Flut in Hamburg informieren zu lassen, ohne dass zuvor ein entsprechender Wasserstandsensor registriert wurde. Hier könnte das Konzept dahingehend erweitert werden, dass auch Ereignisse der Prozessverwaltung, z.B. das Registrieren neuer Sensoren, dem Filterdienst als Metadatenstrom zugeführt wird, und Konsumenten „schwebende“ Abfragen subskribieren können, die erst bei Eintreten bestimmter Prozessereignisse beim Filterdienst angemeldet werden.

Für eine Reihe weiterer Herausforderungen im praktischen Einsatz erwiesen sich die vielfältigen Erweiterungsmöglichkeiten des Vermittlungsprozesses als überaus sinnvoll. So konnte beispielsweise der Umgang mit Ausreißern in den inhärent unzuverlässigen Sensordaten, die fälschlicherweise ereignisbasierte Abfragen erfüllen (z.B. $Temperatur > 30°$), mittels einfacher Tiefpassfilter, die als Aktivität in den Verarbeitungs-Workflow von Produzenten eingefügt wurden, verbessert werden. Auf ähnliche Weise konnten auch vertrauliche Verarbeitungsfunktionen, z.B. die Authentifizierung bei externen FTP-Servern oder Datenbanken, unterstützt werden, indem hierfür externe Dienstreferenzen in den Workflows verwendet wurden. Fehlende Unterstützung des Filterdienstes für spezielle Verarbeitungsfunktionen in ereignisbasierten Abfragen, zum Beispiel zur Distanzberechnung für Geokoordinaten, konnten einfach durch Bereitstellung entsprechender Hilfsklassen für die Complex Event Processing-Laufzeitumgebung kompensiert werden, wodurch sich die Abfrageformulierung teils wesentlich vereinfacht.

Auch wenn in der exemplarischen, prototypischen Realisierung nicht alle optionalen Anforderungen umgesetzt wurden, sind aufgrund der vielfältigen Erweiterungsmöglichkeiten keine prinzipiellen Integrationshindernisse der beschriebenen Funktionalitäten zu erwarten.

9.2. Zusammenfassung der Arbeit

Die stetige Verbreitung von Sensoren aller Art sowie die voranschreitende Integration dieser Sensoren in (mobile) Alltagsgegenstände erlauben eine zunehmend umfassendere Beobachtung der Entitäten und Geschehnisse in der Umwelt. Hierdurch eröffnen sich eine Vielzahl neuer Möglichkeiten sowohl im öffentlichen (z.B. Katastrophenwarnung) und im wirtschaftlichen (z.B. Überwachung von Lieferketten) als auch im privaten Bereich (z.B. intelligente Wohnumgebungen).

In vielen Bereichen zeichnet sich darüber hinaus ein Trend zur weitergehenden Automatisierung ab, was dazu führt, dass die erhobenen Sensordaten nicht mehr von Menschen, sondern zunehmend direkt von Software verarbeitet und analysiert werden sowie als Basis für Entscheidungen und die automatisierte Reaktion dienen. Temperaturwerte müssen somit nicht mehr dem Bewohner eines Hauses auf einem Thermometer angezeigt werden, damit dieser manuell die Heizung regelt; Temperatursensor und Heizungsregelung könnten in Zukunft direkt miteinander interagieren. Bei dieser Form des Austauschs von Kontextinformationen und ggf. Steuerbefehlen zwischen „Maschinen“ müssen insbesondere die Heterogenität der Geräte und die von diesen unterstützten Datenformate, Protokolle, Kommunikationstechnologien etc. verborgen und somit eine transparente Interaktion ermöglicht werden. Dies erfordert zwangsläufig die Unterstützung durch eine vermittelnde Instanz, da eine Übereinkunft hinsichtlich verwendeter Standards nicht realistisch ist und die Übersetzung zwischen den verwendeten Formaten, Protokollen und Technologien vieler Ressourcen bedarf.

Verwandte Arbeiten aus den Anwendungsbereichen des Sensor Webs und des Internet der Dinge haben diesen Punkt bereits aufgegriffen, jedoch liegt deren Fokus auf der reinen Datenvermittlung zwischen heterogenen Technologien in globalen Netzen. An dieser Stelle setzt das Konzept eines Vermittlers für kontextbasierte Sichten an, dessen Ziel es ist, Kontextdaten beliebiger, auch verteilter, Produzenten (z.B. Sensoren) so aufzubereiten, dass Konsumenten (z.B. Menschen oder Anwendungen) diese höherwertigen Informationen ohne weiteren Aufwand für ihren eigentlichen Zweck nutzen können. Dabei wird die Verarbeitungslogik und -last hin zu einem Vermittler verschoben, welcher je nach Bedarf lokal auf einem Gerät oder beliebig in der Infrastruktur verteilt laufen kann. Da verschiedene Anwendungen und Nutzer unterschiedliche Anforderungen haben können, muss die Verarbeitung individuell nach Maßgabe der Konsumenten erfolgen. Zu diesem Zweck können Konsumenten einen Vermittler mittels Verarbeitungs-Workflows anweisen, wie anwendungsadäquate kontextbasierte Sichten, welche lediglich relevante Informationen in einer geeigneten Repräsentation umfassen, zu erstellen sind. Das Gleiche gilt auch für Produzenten, die ebenfalls Verarbeitungsanweisungen vorgeben können, um ihre Daten aufbereiten oder anreichern zu lassen.

Darüber hinaus bietet ein Vermittler auch weitergehende Funktionen, welche den Umgang und die Verwaltung von Kontext- und Metadaten betref-

fen: So können die Daten in verschiedenartigen Datenhaltungssystemen persistiert und über eine technologieneutrale Schnittstelle abgefragt werden. Ein Filterdienst erlaubt das Subskribieren ereignisbasierter Abfragen auf Datenströmen, wobei auch kausale und temporale Bedingungen und Beziehungen zwischen Ereignissen sowie komplexe Inhaltefilter definiert werden können. Eine Metadatenverwaltung erlaubt nicht nur das Auffinden geeigneter Produzenten, sondern unterstützt die Entwickler und Anwender bei der Formulierung von Abfragen und Workflows. Außerdem stellt sie die Grundlage der automatisierten Ressourcenverwaltung dar, da sie u.a. auch aktuelle Lastdaten einzelner Verarbeitungsknoten bereitstellt und somit die dynamische Adaption des Vermittlungsprozesses an schwankende Lasten ermöglicht. Durch Berücksichtigung etablierter Standards des Sensor Webs und des Internet der Dinge kann ein Vermittler zudem mit existierenden Systemen interoperieren und höherwertige Aufgaben, wie das Erstellen und Vermitteln kontextbasierter Sichten, übernehmen.

Die exemplarische, prototypische Implementierung eines Vermittlers für Standardcomputer sowie Android-basierte Geräte beweist die prinzipielle Umsetzbarkeit des Konzepts. Auf Basis des Active Component-Paradigmas in Verbindung mit der Jadex-Laufzeitumgebung wurden alle wesentlichen Kernkomponenten eines Vermittlers sowie eine Reihe zusätzlicher Funktionalitäten implementiert. Eine qualitative Evaluation im Rahmen zweier Forschungsprojekte sowie eine quantitative Evaluation unter Laborbedingungen zeigen, dass die Implementierung die wichtigsten funktionalen und nicht-funktionalen Anforderungen an den Vermittlungsprozess umsetzen kann. Eine Reihe von Werkzeugen, die im Kontext dieser Arbeit entwickelt wurden, unterstützen dabei den Einsatz und erleichtern die Verwendung für Benutzer, Entwickler und Administratoren.

Zusammenfassend leistet diese Arbeit durch eine umfassende Anforderungsanalyse, der Konzeption eines Vermittlungsprozesses für kontextbasierte Sichten unter Berücksichtigung der besonderen Charakteristika mobiler und ubiquitärer Systeme sowie der exemplarischen Realisierung eines prototypischen Vermittlers einen Beitrag zur Evolution der Konzepte, Mechanismen, Verfahren und Best Practices für den nächsten Entwicklungsschritt hin zur Erfüllung der Vision des Ubiquitous Computing.

9.3. Ausblick

Das in dieser Arbeit entwickelte Konzept der Vermittlung kontextbasierter Sichten in mobilen, ubiquitären Systemen bietet noch eine Reihe von Anknüpfungspunkten für weitere Forschungs- und Entwicklungsaktivitäten. Anhand des Leitbilds des „Internet des Wissens“ (engl. *Internet of Knowledge*) sollen diese im Folgenden kurz dargelegt werden.

Der Metapher „Vermittlung von Wissen“ bzw. „Vermittlung von Eindrücken“ folgend, ist das übergeordnete Ziel auch des in dieser Arbeit beschriebenen Konzepts, das Vermitteln von Sachverhalten, Wahrnehmungen oder Vorstel-

lungen in verständlicher Art und Weise. Das bedeutet, Informationen müssen auf dem Weg zum Empfänger so aufbereitet und präsentiert werden, dass dieser sie ohne zusätzliches Wissen verstehen kann. Das Internet heutzutage bietet bereits einen riesigen Fundus an Informationen, welche jedoch in den seltensten Fällen semantisch annotiert sind und somit schwerlich automatisch interpretiert werden können. In gewissem Maße schaffen hier spezialisierte Erweiterungen Abhilfe, das Internet der Dinge oder das Sensor Web beispielsweise, welche auf den etablierten Internet-Standards aufsetzen, aber einen engen Fokus hinsichtlich der zu verwaltenden und auszutauschenden Informationen haben und daher semantisch leichter zugänglich sind. Auch finden sich hier Informationen, die über die klassischen Suchmaschinen des Internets nicht aufgefunden werden können. Die Vision des Internet des Wissens verknüpft diese Welten dahingehend, dass die eher statischen Informationen des Internets mit den dynamischen Kontextinformationen über Dinge, Personen und Orte in Beziehung gesetzt werden können. Wesentliche Herausforderungen bestehen hier in der Suche nach gewünschten Informationen, der Verknüpfung unterschiedlicher Informationsquellen sowie der Aufbereitung und angemessenen Präsentation von Ergebnissen für die Nutzer (Menschen und Maschinen).

In einfacher Form stellt ein Vermittler, wie er in dieser Arbeit dargestellt wurde, bereits eine Art Rahmenwerk für entsprechende „Suchmaschinen“ dar. So sieht das Konzept des Vermittlungsprozesses das Auffinden von Daten in verteilten, heterogenen Informationsquellen (z.B. unterschiedlichen Datenhaltungssystemen, Sensor Observation Services, EPC Information Services etc.) vor, erlaubt die individuelle Aufbereitung nach Maßgabe von Klienten, bietet konzeptionelle Ansätze für den Umgang mit semantisch annotierten Informationen sowie deren semantische Abbildung und Ableitung und integriert Teile der Standards des Sensor Webs und des Internet der Dinge. Doch wie bereits in obiger Diskussion der Ergebnisse dargestellt, bedarf es noch einer Reihe (konzeptioneller) Überlegungen, Verbesserungen und Erweiterungen.

Allen voran muss die Benutzbarkeit eiens Vermittlers weitergehend vereinfacht werden. Bisher bedarf es immer noch der Kenntnis von Abfragesprachen (z.B. Esper EQL/EPL) und Modellierungsnotationen (z.B. BPMN) bzw. der Unterstützung durch Werkzeuge zur Erstellung von Abfragen und Verarbeitungs-Workflows. Das Ziel sollte jedoch eine intuitive Abfrage sein, zum Beispiel natürlichsprachlich oder mittels anderer Medien (z.B. Bildern). Hierfür wurden im Rahmen der Arbeit bereits erste, einfache Lösungsansätze aufgezeigt, jedoch sind diese nicht gänzlich ausgereift. Semantische Interpretation von Informationen könnte die Benutzbarkeit in vielen Bereichen verbessern. Doch auch wenn es kleine, meist eher pragmatische Fortschritte in dieser Richtung gibt, stellt dies ein bisher noch nicht zufriedenstellend gelöstes Problem der Informatik dar. Dennoch bedarf es der Integration auch nicht ausgereifter Lösungen, um aus Erfahrungen neue Anforderungen an zukünftige Lösungen abzuleiten.

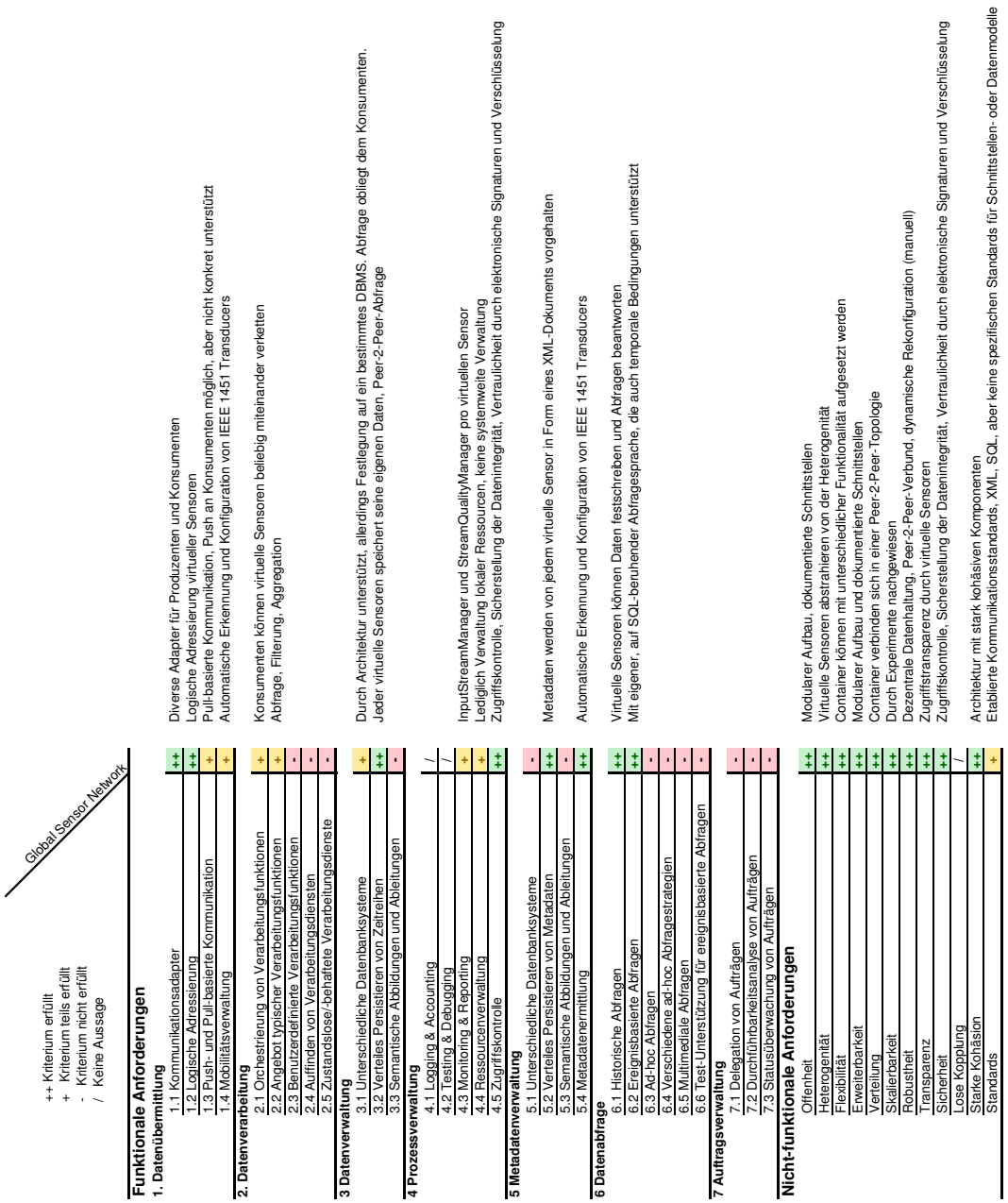
Erfahrung spielt auch weitergehend für einen Vermittler selbst eine Rolle. So könnte dieser aus den Abfragen von Benutzern lernen, relevante von weniger relevanten Informationen zu unterscheiden und somit die Benutzbarkeit von Werkzeugen, bspw. durch Priorisieren von Eingabevorschlägen, zu erhöhen. Einen Schritt weiter geht das Individualisieren der Vermittlungsdienstleistung dahingehend, dass ein Vermittler Interaktionshistorien individueller Benutzer führt und sich auf Basis derer den einzelnen Benutzern anpasst. Hierbei können auch Kontextinformationen über einen Nutzer oder mit diesem assoziierte Objekte oder Orte helfen, sofern die Kontextinformationen entsprechend eindeutig zuzuordnen sind. In einer einfachen Form könnten so beispielsweise kontextabhängige Vorschläge zur Vervollständigung von Abfragen unterbreitet werden, wenn einem Vermittler zum Beispiel der aktuelle Aufenthaltsort des Nutzers, dessen Einkaufshistorie oder Freundeskreis bekannt sind. Etwas weitergehend könnten so auch Abfragen über den persönlichen Kontext ermöglicht werden.

Spätestens an dieser Stelle bedarf es auch der Integration von Maßnahmen zur Sicherstellung der Vertraulichkeit, Integrität und Authentizität von Informationen. Im Rahmen dieser Arbeit wurden mit Maßnahmen wie Verschlüsselung und Signaturen lediglich einfache Lösungsansätze für den Schutz vertraulicher Daten und Dienste präsentiert, welche den Anforderungen bei einem praktischen Einsatz nicht gerecht werden können. Hier bedarf es zunächst eines Konzepts zur feingranularen Rechteverwaltung sowie entsprechender Maßnahmen zur Kontrolle und Durchsetzung der Befugnisse. Auch Ideen, bei denen Nutzer selbst die Kontrolle über ihre Daten und von ihnen angebotene Dienste haben und Befugnisse dezentral koordiniert werden, bieten interessante Ansätze und sind durch die Offenheit, Flexibilität, Erweiterbarkeit und die vielfältigen Möglichkeiten eines Vermittlers, Daten und Dienste auch verteilt zuzugreifen, prinzipiell realisierbar.

Weitergehend bedarf auch das Problem der suboptimalen Nutzung von Ressourcen in der exemplarischen, prototypischen Implementierung eines Vermittlers weiterer Entwicklungs- und ggf. auch Forschungsaktivitäten. Bei Letzterem steht insbesondere die Durchsetzung und Überwachung nichtfunktionaler Aspekte, z.B. Verfügbarkeit oder Leistungsanforderungen, im Vordergrund und bietet weiteres Forschungspotential hinsichtlich selbstkonfigurierender, -optimierender und -heilender Systeme.

Der letzte Punkt des Ausblicks betrifft die Generalisierbarkeit des vorgestellten Konzepts. Im Rahmen dieser Arbeit geht es um die Vermittlung kontextbasierter Sichten. Die Vermittlung und Aufbereitung von Kontextdaten ist eine, wenn auch recht weitgefasste, Einschränkung auf einen bestimmten Anwendungsbereich. Doch auch in anderen Bereichen, in denen es um die Aufbereitung und Vermittlung von Daten geht, könnte der Einsatz eines Vermittlers, wie er hier präsentiert wurde, einen Mehrwert bringen. Die Identifikation weiterer Anwendungsdomänen mit ihren speziellen Anforderungen stellt einen weiteren Anknüpfungspunkt für nachfolgende Arbeiten dar.

A. Kommentare zu den Bewertungsergebnissen verwandter Arbeiten



++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

Global Sensor Network		
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Diverse Adapter für Produzenten und Konsumenten
1.2 Logische Adressierung	++	Logische Adressierung virtueller Sensoren
1.3 Push- und Pull-basierte Kommunikation	+	Pull-basierte Kommunikation, Push an Konsumenten möglich, aber nicht konkret unterstützt
1.4 Mobilitätsverwaltung	+	Automatische Erkennung und Konfiguration von IEEE 1451 Transducers
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	+	Konsumenten können virtuelle Sensoren beliebig miteinander verketten
2.2 Angebot typischer Verarbeitungsfunktionen	+	Abfrage, Filterung, Aggregation
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Durch Architektur unterstützt, allerdings Festlegung auf ein bestimmtes DBMS. Abfrage obliegt dem Konsumenten.
3.2 Verteiles Persistieren von Zeitreihen	++	Jeder virtuelle Sensoren speichert seine eigenen Daten, Peer-2-Peer-Abfrage
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	InputStreamManager und StreamQualityManager pro virtuellen Sensor
4.3 Monitoring & Reporting	+	Lediglich Verwaltung lokaler Ressourcen, keine systemweite Verwaltung
4.4 Ressourcenverwaltung	+	Zugriffskontrolle, Sicherstellung der Datenintegrität, Vertraulichkeit durch elektronische Signaturen und Verschlüsselung
4.5 Zugriffskontrolle	++	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	Metadaten werden von jedem virtuelle Sensor in Form eines XML-Dokuments vorgehalten
5.2 Verteiles Persistieren von Metadaten	++	
5.3 Semantische Abbildungen und Ableitungen	-	Automatische Erkennung und Konfiguration von IEEE 1451 Transducers
5.4 Metadatenermittlung	++	
6 Datenabfrage		
6.1 Historische Abfragen	++	Virtuelle Sensoren können Daten festschreiben und Abfragen beantworten
6.2 Ereignisbasierte Abfragen	++	Mit eigener, auf SQL-beruhender Abfragesprache, die auch temporale Bedingungen unterstützt
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	++	Modularer Aufbau, dokumentierte Schnittstellen
Heterogenität	++	Virtuelle Sensoren abstrahieren von der Heterogenität
Flexibilität	++	Container können mit unterschiedlicher Funktionalität aufgesetzt werden
Erweiterbarkeit	++	Modularer Aufbau und dokumentierte Schnittstellen
Verteilung	++	Container verbinden sich in einer Peer-2-Peer-Topologie
Skalierbarkeit	++	Durch Experimente nachgewiesen
Robustheit	++	Dezentrale Datenhaltung, Peer-2-Peer-Verbund, dynamische Rekonfiguration (manuell)
Transparenz	++	Zugriffstransparenz durch virtuelle Sensoren
Sicherheit	++	Zugriffskontrolle, Sicherstellung der Datenintegrität, Vertraulichkeit durch elektronische Signaturen und Verschlüsselung
Lose Kopplung	/	
Starke Kohäsion	++	Architektur mit stark kohäsiven Komponenten
Standards	+	Etablierte Kommunikationsstandards, XML, SQL, aber keine spezifischen Standards für Schnittstellen- oder Datenmodelle

Abbildung A.1.: Kommentiertes Bewertungsergebnis: Global Sensor Network

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

Sensor Web Enablement		
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Kann durch die SensorBus-Erweiterung realisiert werden
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	++	Pull wird vom Sensor Observation Service unterstützt, Push vom Web Notification Service
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	Nicht direkt Teil des Sensor Web Enablements (siehe OpenGIS Web Processing Service)
2.2 Angebot typischer Verarbeitungsfunktionen	-	
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung einzelner Dienstimplementierungen prinzipiell möglich
3.2 Verteiles Persistieren von Zeitreihen	+	Die Standards sehen nicht vor, dass einzelne Dienstinstanzen auf einer verteilten Datenbasis arbeiten, obwohl es prinzipiell möglich wäre
3.3 Semantische Abbildungen und Ableitungen	++	Kann z.B. durch Semantic Enablement Layer oder SemSOS realisiert werden
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	-	
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung einzelner Dienstimplementierungen prinzipiell möglich
5.2 Verteiles Persistieren von Metadaten	+	Die Standards sehen nicht vor, dass einzelne Dienstinstanzen auf einer verteilten Datenbasis arbeiten, obwohl es prinzipiell möglich wäre
5.3 Semantische Abbildungen und Ableitungen	++	Kann z.B. durch Semantic Enablement Layer oder SemSOS realisiert werden
5.4 Metadatenermittlung	++	Kann durch die Sensor Instance Registry durchgeführt werden
6 Datenabfrage		
6.1 Historische Abfragen	++	Wird durch Sensor Observation Service realisiert
6.2 Ereignisbasierte Abfragen	++	Wird durch Sensor Event Service realisiert
6.3 Ad-hoc Abfragen	++	Wird durch Sensor Planning Service realisiert werden
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimedidale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	++	Wird durch Sensor Planning Service realisiert
7.2 Durchführbarkeitsanalyse von Aufträgen	++	Wird durch Sensor Planning Service realisiert
7.3 Statusüberwachung von Aufträgen	++	Wird durch Sensor Planning Service realisiert
Nicht-funktionale Anforderungen		
Offenheit	++	Gut dokumentierter Satz an Daten- und Schnittstellenmodellen
Heterogenität	+	Für alle Arten von Produzenten und Konsumenten geeignet, erwartet jedoch die Fähigkeit XML zu verarbeiten und erfordert einen IP-Stack
Flexibilität	++	Dienstprofile und die Möglichkeit der leichtgewichtigen Realisierung erlauben auch die Ausführung auf ressourcenbeschränkten Geräten
Erweiterbarkeit	++	Durch gut dokumentierte Schnittstellen und unabhängige Komponenten leicht erweiterbar
Verteilung	+	Die Dienste an sich können leicht verteilt werden, jedoch nicht unter einheitlichem, gemeinsamem Zugriff
Skalierbarkeit	+	Durch dezentrale Datenhaltung und Verarbeitung skaliert das Gesamtsystem, jedoch nicht unbedingt einzelne Dienste
Robustheit	+	Unterstützung von Verzeichnisdiensten erlaubt das erneute, dynamische Binden im Fehlerfall. Daten werden nicht zwangsläufig repliziert.
Transparenz	+	Zugriff auf Sensoren und historische Daten erfolgt transparent über Dienstaufrufe, für viele andere Funktionen ist Wissen über technische Details erforderlich
Sicherheit	-	
Lose Kopplung	++	Dienste sind lose gekoppelt, dynamisches Binden nach Suche über Verzeichnisdienste zur Laufzeit.
Starke Kohäsion	++	Dienste sowie ihre einzelnen Bestandteile haben jeweils fest umrissene Aufgaben
Standards	++	SWE selbst setzt die Standards.Diese beruhen u.a. auf HTTP, XML und Webservices

Abbildung A.2.: Kommentiertes Bewertungsergebnis: Sensor Web Enablement

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	Sensor Web Agent Platform	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Können sowohl produzenten- als auch konsumentenseitig durch Sensor Agents bzw. Application Agents realisiert werden.
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	++	Durch Verwendung der SWE-Standards Sensor Observation Service und Web Notification Service unterstützt
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	++	Unterstützt durch Workflow Agents
2.2 Angebot typischer Verarbeitungsfunktionen	-	
2.3 Benutzerdefinierte Verarbeitungsfunktionen	++	Nutzer können eigene Workflow Agents definieren und dem System zur Laufzeit hinzufügen
2.4 Auffinden von Verarbeitungsdiensten	++	Durch Verzeichnisdienste realisiert
2.5 Zustandslose/-behaftete Verarbeitungsdienste	++	Unterstützt einerseits durch Agenten-Dienste, andererseits durch Webservice-Dienste
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung einzelner Dienstimplementierungen prinzipiell möglich
3.2 Verteiles Persistieren von Zeitreihen	+	Nicht vorgesehen, aber aufgrund der freien Wahl des Persisteenzmechnismus prinzipiell möglich
3.3 Semantische Abbildungen und Ableitungen	++	Verwendet Ontologien für Sensoren, Sensordaten und Verarbeitungsprozesse
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	+	QoS-Messungen erwähnt, aber nicht weiter beschrieben
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung einzelner Dienstimplementierungen prinzipiell möglich
5.2 Verteiles Persistieren von Metadaten	+	Nicht vorgesehen, aber aufgrund der freien Wahl des Persisteenzmechnismus prinzipiell möglich
5.3 Semantische Abbildungen und Ableitungen	++	Verwendet Ontologien für Sensoren, Sensordaten und Verarbeitungsprozesse
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Durch SWE-Sensor Observation Service realisiert
6.2 Ereignisbasierte Abfragen	++	Durch SWE-Sensor Alert Service realisiert
6.3 Ad-hoc Abfragen	-	Erwähnt, aber nicht weiter beschrieben
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	+	Erwähnt, aber nicht weiter beschrieben
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	++	Verwendung offener, standardisierter Schnittstellen
Heterogenität	++	Sensor Agents kapseln Zugriff auf Sensoren, Application Agents den Zugriff von Seiten des Nutzers.
Flexibilität	++	Neue Agenten und Workflows können einfach hinzugefügt und entfernt werden. Durch Lose Kopplung lassen sich alle Teile beliebig über Knoten hinweg verteilen
Erweiterbarkeit	++	Neue Agenten und Workflows können einfach hinzugefügt und entfernt werden.
Verteilung	++	Einfache Verteilung unterstützt durch Agenten- und Dienstparadigma. Unklar, ob der Application Catalogue zentral realisiert ist.
Skalierbarkeit	++	Durch Möglichkeiten der Verteilung sollte das System gut skalieren. Der Application Catalogue, falls nicht verteilt, wird nur selten zugegriffen.
Robustheit	+	Dienste und Daten nicht repliziert, könnte aber durch die Architektur unterstützt werden.
Transparenz	++	Transparenter Zugriff über Application Agents
Sicherheit	-	
Lose Kopplung	++	Agenten und Dienste werden über Dienstverzeichnis gefunden und gebunden.
Starke Kohäsion	+	Agenten und Dienste haben relativ fest umrissene Aufgabenbereiche
Standards	++	Verwendung existierender Standards: SWE-Standards, OWL, FIPA

Abbildung A.3.: Kommentiertes Bewertungsergebnis: Sensor Web Agent Platform

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	GeoSwift	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Sensing Server abstrahiert von Kommunikationsprotokollen und Datenformaten
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	-	
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	
2.2 Angebot typischer Verarbeitungsfunktionen	-	
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	-	
3.2 Verteiltes Persistieren von Zeitreihen	++	Verwendet eine verteilte Hashtabelle
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	-	
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	
5.2 Verteiltes Persistieren von Metadaten	++	Verwendet eine verteilte Hashtabelle
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Über HTTP-GET und Schnittstelle des SWE-Sensor Observation Service
6.2 Ereignisbasierte Abfragen	-	
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	+	Es basiert auf den SWE-Standards, hat aber selbst keine weiteren dokumentierten Schnittstellen oder Datenmodelle
Heterogenität	+	Sensing Server kann mittels Adaptern auf verschiedene Sensoren zugreifen
Flexibilität	-	Die Infrastrukturdienste sind darauf ausgelegt über viele Ressourcen zu verfügen und ständig erreichbar zu sein
Erweiterbarkeit	+	Es können neue Sensoren hinzugefügt werden, weitere Dienste sind nicht vorgesehen
Verteilung	++	Datenhaltung und das Abfragen von Daten erfolgt über ein Peer-to-Peer-Netz
Skalierbarkeit	++	Aufgrund der Peer-to-Peer-Toopologie und der Verwendung verteilter Hashtabellen skaliert das System gut. Verhalten bei hoher Dynamik ist offen
Robustheit	+	Durch verteilte Datenhaltung und Peer-to-Peer-Ansatz ist das System robust. Daten werden allerdings nicht repliziert.
Transparenz	++	Peer-to-Peer-Ansatz bietet ortstransparenzen Zugriff auf Sensordaten
Sicherheit	-	
Lose Kopplung	++	Dienste sind lose über eine Registrierung gekoppelt
Starke Kohäsion	/	
Standards	++	Nutzt eine Teilmenge der SWE-Standards

Abbildung A.4.: Kommentiertes Bewertungsergebnis: GeoSwift

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	Hourglass	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Proxies erlauben das Anbinden heterogener Produzenten und Konsumenten
1.2 Logische Adressierung	-	Lediglich Circuits können über eine eindeutige Kennung identifiziert werden
1.3 Push- und Pull-basierte Kommunikation	-	Die Daten werden von Produzenten auf dem Weg zu Konsumenten nur per Push übertragen
1.4 Mobilitätsverwaltung	++	Unterstützung mobiler Klienten durch BufferServices und ein Verlässlichkeitsmodell
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	+	Verkettung von Funktionen über proprietäre Circuit-Beschreibungen, lediglich sequentielle und keine konditionale Ausführung
2.2 Angebot typischer Verarbeitungsfunktionen	+	Filterung, Aggregation, Kompression und Zwischenspeichern werden als Standardfunktionen geboten
2.3 Benutzerdefinierte Verarbeitungsfunktionen	+	Können von Dritten durch Implementierung vorgegebener Schnittstellen hinzugefügt werden
2.4 Auffinden von Verarbeitungsdiensten	++	Über ein dezentrales Dienstverzeichnis möglich
2.5 Zustandslose/-behaftete Verarbeitungsdienste	++	Allgemeine und anwendungsspezifische Dienste möglich
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Durch Architektur grundsätzlich unterstützt, allerdings ohne weitere Angaben
3.2 Verteiles Persistieren von Zeitreihen	+	Daten werden vom PersistentStorageService auf einzelnen Nodes gespeichert, jedoch ohne Verknüpfung
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	+	CircuitManager beobachten Leistungskennzahlen
4.4 Ressourcenverwaltung	+	CircuitManager können einfache Ressourcenumwidmung vornehmen
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	Gleichzeitige Verwendung unterschiedlicher DBMS von den Registries nicht sinnvoll unterstützt
5.2 Verteiles Persistieren von Metadaten	++	Verteilte Registries übernehmen das Speichern von Metadaten
5.3 Semantische Abbildungen und Ableitungen	+	Mittels Topics und Predicates ganz rudimentäre Semantik realisiert
5.4 Metadatenermittlung	-	Dienste und Produzenten müssen durch ServiceAnnouncements auf sich aufmerksam machen
6 Datenabfrage		
6.1 Historische Abfragen	-	
6.2 Ereignisbasierte Abfragen	+	Lediglich durch eigene Filter realisierbar
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	-	Verwendet XML, TCP/IP, deklariert einige proprietäre Schnittstellen
Heterogenität	++	Abstrahiert über Circuits von eigentlichen Datenquellen
Flexibilität	++	ServiceProvider auch auf ressourcenbeschränkten Geräten installierbar, ggf. Geräte über Proxy anbinden
Erweiterbarkeit	+	Benutzer können eigene Dienste hinzufügen
Verteilung	++	ServiceProvider und Circuits lassen sich beliebig über Knoten verteilen
Skalierbarkeit	++	Aufgrund der Verteilung ohne Single Point of Failure gut skalierbar
Robustheit	++	Heartbeat-Infrastruktur, verteilte Verarbeitung innerhalb von Circuits ggf. auf unterschiedlichen Knoten
Transparenz	++	Circuits abstrahieren von konkreten Produzenten und Kommunikationsverbindungen
Sicherheit	-	
Lose Kopplung	+	Circuits und Dienste weisen lose Kopplung auf, keine Aussagen über konkrete Realisierung
Starke Kohäsion	++	Wohldefinierte Aufgaben für einzelne Komponenten
Standards	-	Nur XML und TCP/IP, aber keine Sensor Web- oder Internet der Dinge-Standards

Abbildung A.5.: Kommentiertes Bewertungsergebnis: Hourglass

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	HiFi	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	+	Wird nicht explizit addressiert, jedoch werden produzentenseitig im Prototypen zwei unterschiedliche Adapter verwendet
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	-	Kann durch ad-hoc-Verbindungen zwischen Knoten mit eigenem Verbindungsmanagement realisiert werden.
1.4 Mobilitätsverwaltung	++	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	Der Nutzer hat keinen Einfluss auf die Verarbeitung in verschiedenen Verarbeitungsstufen
2.2 Angebot typischer Verarbeitungsfunktionen	+	Gemäß der einzelnen Verarbeitungsstufen werden einige Verarbeitungsfunktionen (z.B. Filterung, Aggregation etc.) bereitgestellt
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung des Archive Managers prinzipiell möglich
3.2 Verteiles Persistieren von Zeitreihen	++	Zeitreihen werden verteilt auf einzuelnen Knoten persistiert. Durch einen Abfrageplaner können diese Daten auch verteilt abgefragt werden.
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	++	Lokal durch einen Resource Manager, global durch einen Control Manager
4.4 Ressourcenverwaltung	++	Lastverteilung kann durch Neubindung von Verbindungen erzielt werden
4.5 Zugriffskontrolle	++	Zugriffskontrolle wird durch Abfrageplaner durchgesetzt
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	Global zugreifbares Metadata Repository würde zwar unterschiedliche Datenbanksysteme unterstützen, allerdings nur einheitlich auf allen Knoten dasselbe.
5.2 Verteiles Persistieren von Metadaten	++	Metadaten können vom Metadata Repository verteilt gespeichert und zugegriffen werden. Weitere Metadaten auch lokal auf einzelnen Knoten vorgehalten.
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Soll unterstützt sein, keine genaueren Angaben gegeben
6.2 Ereignisbasierte Abfragen	++	Durch eigene SQL-basierte Abfragesprache, die von einem Data Stream Processor verarbeitet wird
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	-	Keine Aussagen über offene Schnittstellen, keine besondere Verwendung existierender Standards
Heterogenität	++	Unterstützt unterschiedliche Arten von Geräten in der Verarbeitungshierarchie
Flexibilität	++	Unterstützt unterschiedlich leistungsstarke Knoten auf verschiedenen Hierarchieebenen. Dienste teils optional und können leicht ausgetauscht werden.
Erweiterbarkeit	+	Kann um weitere Teilnehmer erweitert werden, jedoch ist die gebotene Funktionalität nicht einfach erweiterbar
Verteilung	++	Hierarchisches Verteilungskonzept
Skalierbarkeit	++	Hierarchisches Verteilungskonzept
Robustheit	+	Fehlerhafte Knoten können entdeckt und durch Neubindung von Knoten übersprungen werden. Daten werden nicht repliziert
Transparenz	++	Ortstransparenter Zugriff auf Daten ohne technische Details der Produzenten kennen zu müssen
Sicherheit	+	Zugriffskontrolle wird durch Zugriffsberechtigungen durch den Abfrageplaner durchgesetzt
Lose Kopplung	++	Einzelne Komponenten lose gekoppelt, Metadata Repository zur Suche nach Diensten und Knoten, dynamisches Binden zur Laufzeit
Starke Kohäsion	++	Komponenten des HiFi-Systems haben fest umrissene Aufgaben, die auch auf unterschiedlichen Ebenen der Hierarchie anders ausgeprägt sind (CSAVA).
Standards	-	Außer einer proprietären Erweiterung von SQL sowie etablierten Internetstandards für die Kommunikation keine weiteren Standards berücksichtigt

Abbildung A.6.: Kommentiertes Bewertungsergebnis: HiFi

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	IrisNet	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Sensing Agents erlauben das Anbinden unterschiedlicher Sensortypen
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	/	
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	
2.2 Angebot typischer Verarbeitungsfunktionen	-	
2.3 Benutzerdefinierte Verarbeitungsfunktionen	++	ServiceAgents können benutzerdefinierte Senselets ausführen, die untereinander Daten austausch können
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	++	Da Senselets vom Benutzer selbst erstellt werden können, sind beide Varianten möglich
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	-	Eigene XML-basierte Datenverwaltung
3.2 Verteiles Persistieren von Zeitreihen	++	Verteilte, hierarchische Datenhaltung durch Organizing Agents realisiert
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	++	Durch IrisLog realisiert
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	+	Zugriffskontrolle durch Unterscheidung von (un)trusted Senselets und Privacy Filter
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	
5.2 Verteiles Persistieren von Metadaten	-	
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Historische Abfragen über XPath
6.2 Ereignisbasierte Abfragen	+	"Triggered Queries" erwähnt, aber nicht weiter ausgeführt.
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	+	Verwendung von XML und XPath, DNS, teils offene Schnittstellen
Heterogenität	+	Unterstützt, aber nicht explizit adressiert
Flexibilität	-	IrisNet ist als globaler Verbund angelegt, daher keine wesentliche Unterstützung verschhiedenartiger Deployments
Erweiterbarkeit	+	Funktionsumfang kann mittels Senselets erweitert werden, die Architektur sieht keine besonderen Erweiterungspunkte vor.
Verteilung	+	Daten- und Abfrageverarbeitung geschieht verteilt, die Verarbeitung pro Datenstrom nicht
Skalierbarkeit	++	Keine zentralen Elemente, Abfrage-Routing basiert auf DNS, Datenhaltung hierarchisch verteilt
Robustheit	++	Unterstützung von primären und sekundären Replikas
Transparenz	++	Zugriffstransparenz durch Organizing Agents
Sicherheit	++	(Un)Trusted Senselets, Privacy Filter und "Sandbox"-Ansatz für Senselets
Lose Kopplung	+	Suche nach Organizing Agents über das DNS, weiter keine Aussagen möglich
Starke Kohäsion	/	
Standards	+	Nur XML, XPath und DNS, aber keine Sensor Web- oder Internet der Dinge-Standards

Abbildung A.7.: Kommentiertes Bewertungsergebnis: IrisNet

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	SStreaMWare	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Adapter Services zur Anbindung unterschiedlichster Arten von Sensoren
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	-	
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	Sensordaten werden nicht verarbeitet
2.2 Angebot typischer Verarbeitungsfunktionen	-	
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	-	Sensordaten werden nicht persistiert
3.2 Verteiles Persistieren von Zeitreihen	-	
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	-	
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	Properties werden verteilt gespeichert
5.2 Verteiles Persistieren von Metadaten	++	
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	-	Formulierung von Abfragen durch proprietäre SQL-Erweiterung
6.2 Ereignisbasierte Abfragen	++	
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	-	
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	+	Dokumentierte Schnittstellen, Service-orientierte Architektur, teils Verwendung von Standards
Heterogenität	+	Am Rand des Netzes können beliebige Sensoren mittels Adapter angebunden werden, im Kern werden leistungsstärkere Computer vorausgesetzt
Flexibilität	-	Die Infrastrukturdienste sind darauf ausgelegt über viele Ressourcen zu verfügen und ständig erreichbar zu sein
Erweiterbarkeit	++	Service-orientierte Architektur, Verwendung von OSGi, dynamisches Binden von Diensten zur Laufzeit
Verteilung	++	Hierarchisches Verteilungskonzept
Skalierbarkeit	+	Hierarchisches Verteilungskonzept, zentraler Top-Level-Knoten über den alle Abfragen und Ergebnisse laufen
Robustheit	-	Ausfall eines Knotens betrifft den gesamten Teilbaum bis zu den Blättern und ggf. alle subskribierten Abfragen bis zum Top-Level-Knoten
Transparenz	++	Orts- und Zugriffstransparenz durch Abstraktion von Sensoren und Verwendung von OSGi-Diensten
Sicherheit	-	
Lose Kopplung	++	Dienste werden zur Laufzeit über einen Lookup Service gefunden und gebunden
Starke Kohäsion	++	Einzelne Dienste haben fest umrissene Aufgabenbereiche
Standards	+	Proprietären Erweiterung von SQL, etablierte Internetstandards für die Kommunikation, OSGi

Abbildung A.8.: Kommentiertes Bewertungsergebnis: SStreaMWare

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	EPCglobal / Fosstrak	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Beliebige Produzenten durch Hardware Abstraction Layer unterstützt, Konsumenten müssen per SOAP und HTTP kommunizieren.
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	+	Beide Varianten unterstützt, Push jedoch nur per HTTP
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	-	
2.2 Angebot typischer Verarbeitungsfunktionen	+	Einige Funktionen spezifisch für den Umgang mit RFID werden geboten, z.B. Filtern und Aggregieren
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	Nur durch eigene Implementierung einer EPCIS Accessing Application
2.4 Auffinden von Verarbeitungsdiensten	-	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	-	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	+	Nicht vorgesehen, aber bei Anpassung der EPCIS-Implementierung prinzipiell möglich
3.2 Verteiles Persistieren von Zeitreihen	+	Die Standards sehen nicht vor, dass EPCIS-Dienste auf einer verteilten Datenbasis arbeiten, obwohl es prinzipiell möglich wäre
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	/	
4.3 Monitoring & Reporting	-	
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	
5.2 Verteiles Persistieren von Metadaten	-	
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Über das EPCIS
6.2 Ereignisbasierte Abfragen	++	Über die Filtering & Collection Middleware
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	++	Reader Simulator erlaubt das Simulieren von Lesegeräten und Tags
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	++	Tags können beschrieben und Reader konfiguriert werden
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	++	Dokumentierte, standardisierte Schnittstellen
Heterogenität	+	Unterstützt heterogene Produzenten, verlangt jedoch Server für die Infrastrukturdienste
Flexibilität	+	EPCIS und Filtering & Collection sind optional.
Erweiterbarkeit	++	Unterschiedliche Erweiterungspunkte sind gegeben. Wenig Details verfügbar.
Verteilung	+	Die Dienste an sich können leicht verteilt werden, jedoch nicht unter einheitlichem, gemeinsamem Zugriff
Skalierbarkeit	+	Durch dezentrale Datenhaltung und Verarbeitung skaliert das Gesamtsystem, jedoch nicht unbedingt einzelne Dienste
Robustheit	+	Zustandsbehaftete Dienstinstanzen können verteilt und in mehreren Instanzen ausgeführt werden. Daten werden nicht repliziert.
Transparenz	++	Business Events abstrahieren von allen technischen Details darunterliegender Schichten
Sicherheit	/	
Lose Kopplung	/	
Starke Kohäsion	+	Komponenten haben mehr oder weniger eng umrissende Aufgaben. Manche Aufgaben in mehreren Komponenten zu finden
Standards	++	Baut auf den Standards von EPCglobal auf, unterstützt HTTP, Webbservices, XML

Abbildung A.9.: Kommentiertes Bewertungsergebnis: EPCglobal / Fosstrak

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	Savant	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Unterschiedliche Arten von RFID-Readern können über Adapter angesprochen werden
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	+	Push nicht explizit unterstützt, kann aber durch Logger einfach abgebildet werden
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	++	Benutzer sollen Processing Modules selbst kombinieren können, keine weiteren Details gegeben
2.2 Angebot typischer Verarbeitungsfunktionen	++	Angebot von Standard Processing Modules z.B. zum Zugriff auf Auto-Id Dienste
2.3 Benutzerdefinierte Verarbeitungsfunktionen	++	Benutzer können eigene Processing Modules definieren, keine weiteren Details gegeben
2.4 Auffinden von Verarbeitungsdiensten	+	Ein Class Server unterstützt das Verteilen von Code, keine weiteren Details gegeben
2.5 Zustandslose/-behaftete Verarbeitungsdienste	++	One-Shot und Continuous Tasks
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	++	Jeder Savant kann unterschiedliche DBMS realisieren, Einschränkung auf JDBC
3.2 Verteiles Persistieren von Zeitreihen	++	Daten werden verteilt in einer Savant-Hierarchie gespeichert
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	-	
4.2 Testing & Debugging	-	
4.3 Monitoring & Reporting	+	Scheduler des Task Management Systems unterstützt Monitoring
4.4 Ressourcenverwaltung	-	
4.5 Zugriffskontrolle	-	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	
5.2 Verteiles Persistieren von Metadaten	-	
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Werden über JDBC-Schnittstelle unterstützt.
6.2 Ereignisbasierte Abfragen	+	Lassen sich über eigene Filter und Logger nachbilden
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	-	
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	++	Soll unterstützt sein, keine weiteren Details gegeben
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	++	Dokumentierte Schnittstellen und Erweiterungpunkte, Verwendung von Standards
Heterogenität	++	Unterstützt unterschiedliche Arten von Geräten in der Verarbeitungskette
Flexibilität	++	Edge und Internal Savants können in unterschiedlichen Konfigurationen laufen
Erweiterbarkeit	++	An diversen Stellen im Gesamtsystem erweiterbar
Verteilung	++	Hierarchisches Verteilungskonzept
Skalierbarkeit	+	Hierarchisches Verteilungskonzept, zentraler Top-Level-Knoten über den alle Abfragen und Ergebnisse laufen
Robustheit	+	Robuster Task Scheduler, bei Ausfall eines Savant-Knotens kann jedoch der gesamte Ast bis zum Blatt nicht mehr operieren. Daten werden (aggregiert) repliziert.
Transparenz	++	Transparenter Zugriff abstrahiert von technischen Details
Sicherheit	-	
Lose Kopplung	+	Einzelne Module durch Schnittstellen entkoppelt, allerdings kein dynamisches Binden zur Laufzeit
Starke Kohäsion	++	Einzelne Dienste haben fest umrissene Aufgabenbereiche
Standards	++	Savant ist eine Referenzimplementierung der Auto-Id-Standards. Verwendet außerdem SOAP, SQL/JDBC und XML.

Abbildung A.10.: Kommentiertes Bewertungsergebnis: Savant

++ Kriterium erfüllt
+ Kriterium teils erfüllt
- Kriterium nicht erfüllt
/ Keine Aussage

	SAP Auto-ID Infrastructure	
Funktionale Anforderungen		
1. Datenübermittlung		
1.1 Kommunikationsadapter	++	Hardware-unabhängige Schnittstellen zur Unterstützung heterogener RFID Reader und Sensoren
1.2 Logische Adressierung	-	
1.3 Push- und Pull-basierte Kommunikation	++	Push und Pull Unterstützung auf Seiten der Device Controller
1.4 Mobilitätsverwaltung	-	
2. Datenverarbeitung		
2.1 Orchestrierung von Verarbeitungsfunktionen	++	Verkettung verschiedener Data Processor möglich
2.2 Angebot typischer Verarbeitungsfunktionen	++	Data Processor bieten eine Reihe typischer Funktionen wie Filtern, Anreichern, Aggregieren etc.
2.3 Benutzerdefinierte Verarbeitungsfunktionen	-	
2.4 Auffinden von Verarbeitungsdiensten	/	
2.5 Zustandslose/-behaftete Verarbeitungsdienste	/	
3 Datenverwaltung		
3.1 Unterschiedliche Datenbanksysteme	-	
3.2 Verteiles Persistieren von Zeitreihen	++	Daten werden auf Auto-ID Nodes verteilt vorgehalten (und ggf. mit Backend synchronisiert)
3.3 Semantische Abbildungen und Ableitungen	-	
4 Prozessverwaltung		
4.1 Logging & Accounting	/	
4.2 Testing & Debugging	+	Test- und Workload Generatoren, skriptbarer Simulator
4.3 Monitoring & Reporting	/	
4.4 Ressourcenverwaltung	/	
4.5 Zugriffskontrolle	/	
5 Metadatenverwaltung		
5.1 Unterschiedliche Datenbanksysteme	-	
5.2 Verteiles Persistieren von Metadaten	++	Daten werden auf Auto-ID Nodes verteilt vorgehalten (und ggf. mit Backend synchronisiert)
5.3 Semantische Abbildungen und Ableitungen	-	
5.4 Metadatenermittlung	-	
6 Datenabfrage		
6.1 Historische Abfragen	++	Über das Auto-ID Repository unterstützt
6.2 Ereignisbasierte Abfragen	++	Rule Engine zur Auswertung benutzerdefinierter Regeln
6.3 Ad-hoc Abfragen	-	
6.4 Verschiedene ad-hoc Abfragestrategien	-	
6.5 Multimediale Abfragen	-	
6.6 Test-Unterstützung für ereignisbasierte Abfragen	++	Test- und Workload Generatoren, skriptbarer Simulator
7 Auftragsverwaltung		
7.1 Delegation von Aufträgen	++	Data Processor zum Schreiben auf RFID Tags und zum Ansteuern von Aktuatoren
7.2 Durchführbarkeitsanalyse von Aufträgen	-	
7.3 Statusüberwachung von Aufträgen	-	
Nicht-funktionale Anforderungen		
Offenheit	-	Trotz Verwendung einzelner Standards sehr auf das Zusammenspiel mit SAP-Produkten ausgelegt
Heterogenität	+	Beliebige Produzenten und Konsumenten, in der Infrastruktur jedoch leistungsstarke Server vorausgesetzt
Flexibilität	-	Eingeschränkte Möglichkeiten hinsichtlich des Deployments und der Konfiguration
Erweiterbarkeit	-	Die wenigen konstituierenden Komponenten können nicht nennenswert erweitert werden
Verteilung	++	Device Controller und Auto-ID Nodes können verteilt werden
Skalierbarkeit	+	Trotz weitreichender Möglichkeit der Verteilung scheint die Rule Engine ein zentrales Element zu sein
Robustheit	++	Da Device Controller und Auto-ID Nodes jeweils gleiche Funktionalitäten besitzen, dürfte ein Ausfall durch eine gleichwertige Instanz kompensiert werden können
Transparenz	++	Transparenter Zugriff auf Daten erfolgt über Auto-ID Nodes bzw. ein Administrations-Werkzeug
Sicherheit	/	
Lose Kopplung	/	
Starke Kohäsion	/	
Standards	+	Verwendung von HTTP, XML, PML und low-level EPCglobal Standards, sonst abhängig von SAP Produkten

Abbildung A.11.: Kommentiertes Bewertungsergebnis: SAP Auto-ID Infrastructure

Eigene Veröffentlichungen

Folgende Veröffentlichungen sind aus den Ergebnissen im Umfeld dieses Dissertationsprojekts hervorgegangen:

D. BADE, K. KREMPELS, S. LILIENTHAL und S. WIDYADHARMA, Agent-Society Configuration Manager and Launcher In *F. Bellifemine and G. Caire and D. Greenwood (Hrsg.): „Developing Multi-Agent Systems with JADE". Wiley & Sons, 2007, pp. 207-223*.

D. BADE, Context-Dependent and Self-Responsible Migration of Software Agents in Heterogeneous Environments In *GI Gesellschaft für Informatik e.V. (Hrsg.): „Informatiktage 2008". GI-Edition, Lecture Notes in Informatics, Bonn, 2008, pp. 57-60*.

S. ZAPLATA, D. BADE und A. VILENICA, Service-based Interactive Workflows for Mobile Environments. In *Hans Robert Hansen, Dimitris Karagiannis, Hans-Georg Fill (Hrsg.): „Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)". Österreichische Computer Gesellschaft, Wien, 2009, pp. 631-640*

S. ZAPLATA, A. VILENICA, D. BADE und C. KUNZE, Abstract User Interfaces for Mobile Processes In *Klaus David, Gurt Geihs (Hrsg.): „16. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2009)". Springer, Berlin, 2009, pp. 129-140*

D. BADE Towards an Extensible Agent-based Middleware for Sensor Networks and RFID Systems In *„Third International Workshop on Agent Technology for Sensor Networks (ATSN-09), Budapest, Hungary". 2009, Technical Report, http://web.mac.com/lteacy/ATSN-09/proceedings.html*

D. BADE, L. BRAUBACH, L. POKAHR und W. LAMERSDORF An Awareness Model for Agents in Heterogeneous Environments In *Hindriks, Pokahr, Sardina (Hrsg.): „Programming Multi-Agent Systems 6". Springer, Berlin, 2009, pp. 152-167*

P. IBACH, D. BADE und S. KUNZ Smart Items in Ereignisgesteuerten Prozessketten In *S. Fischer, E. Maehle, R. Reischuk (Hrsg.): „Verwaltung, Analyse und Bereitstellung kontextbasierter Informationen, Workshop, 39. GI-Jahrestagung, 2009, Lübeck, Germany". GI-Edition, Lecture Notes in Informatics, Bonn, 2009, pp. 2015 - 2028*

D. BADE Verteilte Abfragebearbeitung von Sensordaten In *Fachgespräch Sensornetzwerke, TU Hamburg-Harbung, GI Gesellschaft*

für Informatik e.V., 8.2009, Technical Report, http://www.ti5.tu-harburg.de/events/fgsn09/proceedings/fgsn_013.pdf

D. BADE und W. LAMERSDORF An Agent-based Event Processing Middleware for Sensor Networks and RFID Systems In *Computer Journal, Special Issue on „Agent Technologies for Sensor Networks", The British Computer Society (2010)*

S. ZAPLATA, D. STRASSENBURG, B. WUNDERLICH, D. BADE, K. HAMANN und W. LAMERSDORF Ad-hoc Management Capabilities for Distributed Business Processes In *3rd International Conference on Business Process and Services Computing (BPSC 2010), GI-Edition, Lecture Notes in Informatics (2010)*

D. BADE und R. PUSZIES A Webservice-based Context-Data Collection Service for the Android Platform In *Tagungsband zum 7. GI/KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste, Logos Verlag (2010)*

S. KUNZ, B. FABIAN, H. ZIEKOW und D. BADE From Smart Objects to Smarter Workflows ? An Architectural Approach In *15th IEEE International Enterprise Distributed Object Computing Conference, (EDOCW 2011), Helsinki, Finland, 2011, pp. 194-203*

D. BADE und W. LAMERSDORF Integration von Kontextinformationen in Smart Applications und Smart Workflows In *Claudia Linnhoff-Popien und Stephan Verclas (Hrsgb.): „Smart Mobile Apps". Springer-Verlag, Berlin, 2011, pp. 301-317*

J. WISCHWEH und D. BADE Activity-oriented Context Adaptation in Mobile Applications In *8th International ICST Conference on Mobile and Ubiquitous Systems (MobiQuitous), Copenhagen, Denmark, 2011, Springer LNICST, 2012, pp. 298-313*

D. BADE und D. GLEIM Towards Sensor-supported Indoor Localization Using Cloud-based Machine Learning Techniques In *Tagungsband zum 9. GI/KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste, 2012*

D. BADE Harmlose Sensoren ? Gefahr für die Privatsphäre abseits herkömmlicher Sensordaten-Interpretationen In *hakin9 - Hard Core IT Security Magazin, Nr. 77, 1, 2013*

Abbildungsverzeichnis

Literaturverzeichnis

[52N12a] 52NORTH: *Initiative for Geospatial Open Source Software*. http://52north.org, April 2012. aufgerufen am 26.04.2012.

[52N12b] 52NORTH: *Sensor Web Community*. http://52north.org/SensorWeb, Mai 2012. aufgerufen am 08.05.2012.

[52N13] 52NORTH: *Sensor Event Service Developer Information*. https://wiki.52north.org/bin/view/SensorWeb/SensorEventServiceDeveloperInformation, January 2013. aufgerufen am 05.01.2013.

[ABW06] ARASU, ARVIND, SHIVNATH BABU und JENNIFER WIDOM: *The CQL continuous query language: semantic foundations and query execution*. The VLDB Journal, 15(2):121–142, 2006.

[AGRS05] ADELSTEIN, FRANK., SANDEEP K.S. GUPTA, GOLDEN G.. RICHARDIII und LOREN. SCHWIEBERT: *Fundamentals of mobile and pervasive computing*. McGraw-Hill professional engineering. McGraw-Hill, New York, 2005.

[AHS06a] ABERER, K, M HAUSWIRTH und A SALEHI: *Middleware support for the Internet of Things*. In: *5th GI/ITG KuVS Fachgespraech Drahtlose Sensornetze*, Germany, 2006.

[AHS06b] ABERER, KARL, MANFRED HAUSWIRTH und ALI SALEHI: *Global Sensor Networks*. Technischer Bericht, École Polytechnique Fédérale de Lausanne, 2006.

[AHS06c] ABERER, KARL, MANFRED HAUSWIRTH und ALI SALEHI: *A Middleware for Fast and Flexible Sensor Network Deployment*. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB 2006)*, Seoul, South Korea, 2006.

[AHS07] ABERER, KARL, MANFRED HAUSWIRTH und ALI SALEHI: *Infrastructure for Data Processing in Large-scale Interconnected Sensor Networks*. In: *Proceedings of the Mobile Data Management (MDM 2007)*, Mannheim, Germany, 2007. IEEE Computer Society.

[AIM10] ATZORI, LUIGI, ANTONIO IERA und GIACOMO MORABITO: *The Internet of Things: A survey*. Computer Networks, 54(15):2787 – 2805, 2010.

[Ama12] AMAZON INC.: *Amazon Web Services*. http://aws.amazon.com, September 2012. aufgerufen am 26.09.2012.

[Apa12a] APACHE SOFTWARE FOUNDATION: *Apache River*. http://river.apache.org/, July 2012. aufgerufen am 06.07.2012.

[Apa12b] APACHE SOFTWARE FOUNDATION: *Apache Thrift*. http://thrift.apache.org/, August 2012. aufgerufen am 07.08.2012.

[Apa13a] APACHE SOFTWARE FOUNDATION: *Apache Cassandra Database*. http://cassandra.apache.org, January 2013. aufgerufen am 29.01.2013.

[Apa13b] APACHE SOFTWARE FOUNDATION: *Cassandra Query Language (CQL) v3.0.1*. http://cassandra.apache.org/doc/cql3/CQL.html, February 2013. aufgerufen am 24.02.2013.

[Apa13c] APACHE SOFTWARE FOUNDATION: *Log4j 2*. http://logging.apache.org/log4j/2.x/, January 2013. aufgerufen am 15.01.2013.

[APM05] AKYILDIZ, IAN F., DARIO POMPILI und TOMMASO MELODIA: *Underwater Acoustic Sensor Networks: Research Challenges*. Ad-Hoc Networks (Elsevier), 3:257–279, 2005.

[App12] APPLE INC.: *Apple Push Notification Service*. http://en.wikipedia.org/wiki/Apple_Push_Notification_Service, August 2012. aufgerufen am 10.06.2013.

[AS09] ABBASI, ABU ZAFAR und ZUBAIR A. SHAIKH: *A Conceptual Framework for Smart Workflow Management*. International Conference on Information Management and Engineering, 0:574–578, 2009.

[Aut02] AUTO-ID CENTER, MIT: *The Savant - Technical Manual 0.1 (Alpha)*. http://www.autoidlabs.org/uploads/media/MIT-AUTOID-TM-003.pdf, February 2002.

[Aut12] AUTO-ID LABS: *Homepage*. http://www.autoidlabs.org, November 2012. aufgerufen am 08.11.2012.

[AV10] AKYILDIZ, I.F. und M.C. VURAN: *Wireless Sensor Networks*. Advanced Texts in Communications and Networking. John Wiley & Sons, 2010.

[BAAIA09] BIN AHMAD, M., M. ASIF, M.H. ISLAM und S. AZIZ: *A short survey on distributed in-network query processing in wireless sensor networks*. In: *First International Conference on Networked Digital Technologies*, Seiten 541 –543, July 2009.

[Bac00] BACE, R.: *Intrusion detection*. Macmillan technology series. MacMillan Technical Publishing, 2000.

[Bad07] BADE, DIRK: *Kontextabhaengige und eigenverantwortliche Migration von Software-Agenten in heterogenen Umgebungen*. Diplomarbeit, Universitaet Hamburg - Fakultaet fuer Mathematik, Informatik und Naturwissenschaften - Fachbereich Informatik - Verteilte Systeme und Informationssysteme, May 2007.

[Bad09a] BADE, DIRK: *Towards an Extensible Agent-based Middleware for Sensor Networks and RFID Systems*. In: *Inproceedings of the 3rd Int. Workshop on Agent Technology for Sensor Networks (ATSN-09)*, 2009.

[Bad09b] BADE, DIRK: *Verteilte Abfragebearbeitung von Sensordaten*. Technischer Bericht, Ges. f. Informatik, August 2009.

[Bad10a] BADE, DIRK: *Esper-Android*. `http://vsis-www.informatik.uni-hamburg.de/projects/esper-android/`, October 2010. aufgerufen am 26.02.2013.

[Bad10b] BADE, DIRK: *Jadex-Android*. `http://vsis-www.informatik.uni-hamburg.de/projects/jadex-android/`, October 2010. aufgerufen am 26.02.2013.

[Bad10c] BADE, DIRK: *JESPA-Android*. `http://vsis-www.informatik.uni-hamburg.de/projects/jespa-android/`, October 2010. aufgerufen am 26.02.2013.

[Bad12] BADE, DIRK: *YADrone - Yet Another Drone Framework*. `http://vsis-www.informatik.uni-hamburg.de/projects/yadrone/`, December 2012. aufgerufen am 27.02.2013.

[Bar08] BARTRAM, LYN: *Moving off the Desktop: Pervasive Computing*. http://www.sfu.ca/iat351/Lectures/IAT351-Week8.pdf, Februar 2008.

[Bay12] BAYESSTORE: *Homepage*. http://www.cs.berkeley.edu/~daisyw/BayesStore.html, June 2012. aufgerufen am 01.06.2012.

[BBG10] BUYYA, R., J. BROBERG und A.M. GOSCINSKI: *Cloud Computing: Principles and Paradigms*. Wiley Series on Parallel and Distributed Computing. Wiley, 2010.

[BCL+04] BOHN, JÜRGEN, VLAD COROAMA, MARC LANGHEINRICH, FRIEDEMANN MATTERN und MICHAEL ROHS: *Living in a World of Smart Everyday Objects – Social, Economic, and Ethical Implications*. Journal of Human and Ecological Risk Assessment, 10(5):763–786, Oktober 2004.

[BCTR10] BELLIFEMINE, F., G. CAIRE, TIZIANA TRUCCO und G. RIMASSA: *JADE Programmer's Guide*. http://jade.tilab.com/doc/programmersguide.pdf, April 2010. aufgerufen am 10.06.2013.

[BDF+07] BALAZINSKA, MAGDALENA, AMOL DESHPANDE, MICHAEL J. FRANKLIN, PHILLIP B. GIBBONS, JIM GRAY, MARK HANSEN, MICHAEL LIEBHOLD, SUMAN NATH, ALEXANDER SZALAY und VINCENT TAO: *Data Management in the Worldwide Sensor Web*. IEEE Pervasive Computing, 6(2):30–40, April 2007.

[BEJ+11] BROERING, ARNE, JOHANNES ECHTERHOFF, SIMON JIRKA, INGO SIMONIS, THOMAS EVERDING, CHRISTOPH STASCH, STEVE LIANG und ROB LEMMENS: *New Generation Sensor Web Enablement*. Sensors, 11(3):2652–2699, 2011.

[BFdM10] BERL, ANDREAS, ANDREAS FISCHER und HERMANN DE MEER: *Virtualisierung im Future Internet*. Informatik-Spektrum, 33:186–194, 2010.

[BFJ10] BROERING, A., T. FOERSTER und S. JIRKA: *Interaction patterns for bridging the gap between sensor networks and the Sensor Web*. In: *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Seiten 732 –737, April 2010.

[BFJP10] BROERING, ARNE, THEODOR FOERSTER, SIMON JIRKA und CARSTEN PRIESS: *Sensor bus: an intermediary layer for linking geosensors and the sensor web*. In: *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application*, COM.Geo '10, Seiten 12:1–12:8, New York, NY, USA, 2010. ACM.

[BGKR06] BEIER, STEVE, TYRONE GRANDISON, KARIN KAILING und RALF RANTZAU: *Discovery Services-Enabling RFID Traceability in EPCglobal Networks.* In: LAKSHMANAN, LAKS V. S., PRASAN ROY und ANTHONY K. H. TUNG (Herausgeber): *COMAD*, Seiten 214–217. Tata McGraw-Hill Publishing Company Limited, 2006.

[BGS01] BONNET, PHILIPPE, JOHANNES GEHRKE und PRAVEEN SESHADRI: *Towards Sensor Database Systems*. In: *Proceedings of the Second International Conference on Mobile Data Management*, MDM '01, Seiten 3–14, London, UK, UK, 2001. Springer-Verlag.

[BH13] BACHFELD, DANIEL und SVEN HANSEN: *Home, smart Home!* c't Computertechnik, 9:82–90, 2013.

[BIK09a] BADE, D., P. IBACH und S. KUNZ: *MagicTracker*. `http://wiki.informatik.hu-berlin.de/nomads/index.php/MagicTracker-Plugin`, 2009. aufgerufen am 26.02.2013.

[BIK09b] BADE, D., P. IBACH und S. KUNZ: *MagicTracker User Guide*. `http://wiki.informatik.hu-berlin.de/nomads/index.php/MagicTracker_User_Guide`, 2009. aufgerufen am 26.02.2013.

[BL09] BADE, DIRK und WINFRIED LAMERSDORF: *An Agent-based Event Processing Middleware for Sensor Networks and RFID Systems*. Computer Journal, Special Issue on Agent Technologies for Sensor Networks, 2009.

[BL12] BADE, DIRK und WINFRIED LAMERSDORF: *Integration von Kontextinformationen in Smart Applications und Smart Workflows*, Kapitel 20, Seiten 301–317. Springer, 2012.

[BLHS04] BORNHÖVD, CHRISTOF, TAO LIN, STEPHAN HALLER und JOACHIM SCHAPER: *Integrating automatic data acquisition with business processes experiences with SAP's auto-ID infrastructure*. In: *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, Seiten 1182–1188. VLDB Endowment, 2004.

[Blu10] BLUETOOTH SIG, INC.: *Bluetooth Technology Info Site*. `http://www.bluetooth.com`, August 2010.

[BP11] BRAUBACH, LARS und ALEXANDER POKAHR: *Addressing Challenges of Distributed Systems using Active Components*. In: BRAZIER, FRANCES M. T., KEES NIEUWENHUIS, GREGOR PAVLIN, MARTI-

JN WARNIER und COSTIN BADICA (Herausgeber): *In Proceedings of 5th International Symposium on Intelligent Distributed Computing (IDC-2011)*, Seiten 141–151. Springer, 10 2011.

[BP13] BRAUBACH, LARS und ALEXANDER POKAHR: *Jadex Active Components*. http://www.activecomponents.org, January 2013. aufgerufen am 21.01.2013.

[BPRD08] BOTTS, MIKE, GEORGE PERCIVALL, CARL REED und JOHN DAVIDSON: *OGC Sensor Web Enablement: Overview and High Level Architecture*, Band 4540/2008 der Reihe *Lecture Notes in Computer Science*, Seiten 175–190. Springer Berlin / Heidelberg, 2008.

[BR05] BRAUN, PETER und WILHELM ROSSAK: *Mobile Agents - Basic Concepts, Mobility Models and the Tracy Toolkit*. Morgan Kaufmann and dpunkt.verlag, 2005.

[BS08] BOONMA, P. und J. SUZUKI: *Middleware Support for Pluggable Non-Functional Properties in Wireless Sensor Networks*. In: *IEEE Congress on Services - Part I*, Seiten 360–367, 2008.

[BWC09] BALLARI, DANIELA, MONICA WACHOWICZ und MIGUEL ANGEL MANSO CALLEJO: *Metadata behind the Interoperability of Wireless Sensor Networks*. Sensors, 9(5):3635–3651, 2009.

[BZ10] BOLLMANN, T. und K. ZEPPENFELD: *Mobile Computing*. W3L GmbH, 2010.

[BZIS07] BRUNING, S., J. ZAPOTOCZKY, P. IBACH und V. STANTCHEV: *Cooperative Positioning with MagicMap*. In: *Workshop on Positioning, Navigation and Communication 2007 (WPNC'07), Hannover*, Seiten 17–22, March 2007.

[Car12] CARNEGIE MELLON UNIVERSITY: *CMUSphinx - Open Source Toolkit For Speech Recognition*. http://cmusphinx.sourceforge.net/, September 2012.

[CB07] CHU, X. und R. BUYYA: *Service Oriented Sensor Web*. In: MAHALIK, N. P. (Herausgeber): *Sensor Network and Configuration: Fundamentals, Standards, Platforms, and Applications*, Seiten 51–74. Springer-Verlag, Germany, 2007.

[CCBB00] CATTELL, R.G.G., R.G.G. CATTELL, D.K. BARRY und M. BERLER: *The Object Data Standard: ODMG 3.0*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2000.

[CDK12] COULOURIS, GEORGE, JEAN DOLLIMORE und TIM KINDBERG: *Distributed Systems: Concepts and Design, Edition 5*. Addison-Wesley, Boston, MA, USA, 5 Auflage, 2012.

[Che04] CHEN, HARRY: *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Doktorarbeit, University of Maryland, Baltimore County, December 2004.

[CHE10] CHESS HAARLEM: *MyriaNed - Self Organizing Sensor and Control Nodes*. `http://wsn.chess.nl`, August 2010. aufgerufen am 10.06.2013.

[Cit12] CITRIX SYSTEMS INC.: *CloudStack*. `http://cloudstack.org`, September 2012. aufgerufen am 26.09.2012.

[CJ09] CHAKRAVARTHY, S. und Q.C. JIANG: *Stream Data Processing: a Quality of Service Perspective: Modeling, Scheduling, Load Shedding, and Complex Event Processing*. Advances in Database Systems, 36. Springer London, Limited, 2009.

[CK00] CHEN, GUANLING und DAVID KOTZ: *A Survey of Context-Aware Mobile Computing Research*. Technischer Bericht TR2000-381, Dartmouth College, Hanover, NH, USA, 2000.

[CKDB06] CHU, X., T. KOBIALKA, B. DURNOTA und R. BUYYA: *Open Sensor Web Architecture: Core Services*. In: *International Conference on Intelligent Sensing and Information Processing (ICISIP 2006)*, Seiten 98–103, Bangalore, India, 2006. IEEE Press, Piscataway.

[Cle03] CLEMENTS, P.: *Documenting Software Architectures: Views and Beyond*. Sei Series in Software Engineering. Addison-Wesley, 2003.

[CM00] CASTRO, P. und R. MUNZ: *Managing context data for smart spaces*. Personal Communications, IEEE, 7(5):44–46, oct 2000.

[CN10] CAMPESATO, O. und K. NILSON: *Web 2.0 Fundamentals: With AJAX, Development Tools, and Mobile Platforms*. Jones & Bartlett Learning, 2010.

[Com06] COMER, D.: *Internetworking with TCP/IP: Principles, protocols, and architecture*. Pearson Prentice Hall, 2006.

[Com13a] COMMUNITY, JBOSS: *Drools Fusion*. `http://www.jboss.org/drools/drools-fusion.html`, January 2013. aufgerufen am 23.01.2013.

[Com13b] COMMUNITY, JBOSS: *Hibernate - Relational Persistence for Java and .NET*. http://www.hibernate.org/, April 2013. aufgerufen am 23.04.2013.

[Con12] CONSTELLATION SDI: *A Geospatial Data Services Infrastructure for Science and the Environment*. http://www.constellation-sdi.org, April 2012. aufgerufen am 26.04.2012.

[Cop04] COPELAND, L.: *A Practitioner's Guide to Software Test Design*. Artech House Computing Library. Artech House, 2004.

[Cro13] CROCKFORD, D.: *Javascript Object Notation*. http://www.json.org, June 2013. aufgerufen am 10.06.2013.

[CS05] CRADDOCK, R.J. und E.V. STANSFIELD: *Sensor fusion for smart containers*. In: *Signal Processing Solutions for Homeland Security, 2005. The IEE Seminar on (Ref. No. 2005/11108)*, Seite 12 pp., 2005.

[CTAO03] CLARK, SEAN, KEN TRAUB, DIPAN ANARKAT und TED OSINSKI: *Auto-ID Savant Specification 1.0*. Technischer Bericht, Auto-ID Center, MIT, 2003.

[CYD06] COOK, DIANEJ., MICHAEL YOUNGBLOOD und SAJALK. DAS: *A Multi-agent Approach to Controlling a Smart Environment*. In: AUGUSTO, JUANCARLOS und CHRISD. NUGENT (Herausgeber): *Designing Smart Homes*, Band 4008 der Reihe *Lecture Notes in Computer Science*, Seiten 165–182. Springer Berlin Heidelberg, 2006.

[DA99] DEY, ANIND K. und GREGORY D. ABOWD: *Towards a Better Understanding of Context and Context-Awareness*. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, Seiten 304–307, London, UK, 1999. Springer-Verlag.

[Dav93] DAVENPORT, T.H.: *Process innovation: reengineering work through information technology*. Harvard Business School Publishing India Pvt. Limited, 1993.

[DGD05] DUAN, ZHENHAI, KARTIK GOPALAN und YINGFEI DONG: *Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic*. In: *Proc. of USENIX SRUTI 2005 Workshop, MIT, Cambridge, MA*, July 2005.

[Dou04] DOURISH, PAUL: *What we talk about when we talk about context*. Personal Ubiquitous Computing, 8(1):19–30, Februar 2004.

[DPPR06] DELICATO, FLÁVIAC., LUCI PIRMEZ, PAULOF. PIRES und JOSÉFERREIRA REZENDE: *Exploiting Web Technologies to Build Autonomic Wireless Sensor Networks*. In: PUJOLLE, GUY (Herausgeber): *Mobile and Wireless Communication Networks*, Band 211 der Reihe *IFIP The International Federation for Information Processing*, Seiten 99–114. Springer US, 2006.

[DS07] DALVI, NILESH und DAN SUCIU: *Management of probabilistic data: foundations and challenges*. In: *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, Seiten 1–12, New York, NY, USA, 2007. ACM.

[DS08] DELIN, KEVIN A. und EDWARD SMALL: *The Sensor Web: Advanced Technology for Situational Awareness*, Kapitel 1, Seiten 1–15. John Wiley & Sons, Inc., 2008.

[DSSG+08] DE SOUZA, LUCIANA MOREIRA SÁ, PATRIK SPIESS, DOMINIQUE GUINARD, MORITZ KÖHLER, STAMATIS KARNOUSKOS und DOMNIC SAVIO: *SOCRADES: a web service based shop floor integration infrastructure*. In: *Proceedings of the 1st international conference on The internet of things*, IOT'08, Seiten 50–67, Berlin, Heidelberg, 2008. Springer-Verlag.

[EBF+10] EDLICH, S., C. BINDER, A. FRIEDLAND, J. HAMPE und B. BRAUER: *NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. Hanser Fachbuchverlag, 2010.

[EC08] EGGER, ANNE E. und ANTHONY CARPI: *Data: Analysis and Interpretation*. `http://www.visionlearning.com/library/module_viewer.php?mid=154`, 2008. aufgerufen am 23.03.2013.

[ED06] EREN, EVREN und KAI-OLIVER DETKEN: *Mobile Security - Risiken mobiler Kommunikation und Loesungen zur mobilen Sicherheit*. hanser Verlag, 2006.

[Edl13] EDLICH, STEFAN: *NoSQL-Database.org*. `http://nosql-database.org`, January 2013. aufgerufen am 29.01.2013.

[EM09] EVENSEN, PÅL und HEIN MELING: *SenseWrap: A Service Oriented Middleware with Sensor Virtualization and Self-Configuration*. In: *5th Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing*, December 2009.

[EnO13] ENOCEAN GMBH: *Energy Harvesting Wireless Sensor Solutions and Networks*. `http://www.enocean.com`, June 2013. aufgerufen am

10.06.2013.

[EPC07] EPCGLOBAL INC.: *EPC Information Services (EPCIS), Version 1.0.1*. Technischer Bericht, EPCglobal Inc., http://www.gs1.org/gsmp/kc/epcglobal/epcis, September 2007. aufgerufen am 08.01.2013.

[EPC08] EPCGLOBAL INC.: *Object Naming Service (ONS), Version 1.0.1*. Technischer Bericht, EPCglobal Inc., http://www.gs1.org/gsmp/kc/epcglobal/ons, May 2008. aufgerufen am 08.01.2013.

[EPC10] EPCGLOBAL INC.: *The EPCglobal Architecture Framework, Version 1.4*. Technischer Bericht, EPCglobal Inc., http://www.gs1.org/gsmp/kc/epcglobal/architecture, December 2010. aufgerufen am 08.01.2013.

[EPC11] EPCGLOBAL INC.: *The Application Level Events (ALE) Specification, Version 1.1.1*. Technischer Bericht, EPCglobal Inc., http://www.gs1.org/gsmp/kc/epcglobal/ale, April 2011. aufgerufen am 08.01.2013.

[EPC12] EPCGLOBAL INC.: *Homepage*. http://www.epcglobalinc.org, April 2012. aufgerufen am 08.01.2013.

[EPC13] EPCGLOBAL INC.: *EPC Discovery Services Standard*. Technischer Bericht, EPCglobal Inc., http://www.gs1.org/gsmp/kc/epcglobal/discovery, January 2013. aufgerufen am 08.01.2013.

[Esp10] ESPERTECH INC.: *Esper - Performance*. http://docs.codehaus.org/display/ESPER/Esper+performance, May 2010. aufgerufen am 04.06.2013.

[Esp13a] ESPERTECH INC.: *Esper - Event Stream Intelligence*. http://esper.codehaus.org, January 2013.

[Esp13b] ESPERTECH INC.: *Esper Reference Documentation*. http://esper.codehaus.org/esper/documentation/documentation.html, January 2013. aufgerufen am 21.01.2013.

[Euc12] EUCALYPTUS SYSTEMS INC.: *Eucalyptus Cloud*. http://www.eucalyptus.com, September 2012. aufgerufen am 26.09.2012.

[Eur11] EUROPEAN RESEARCH CLUSTER ON THE INTERNET OF THINGS: *Internet of Things - Strategic Research Agenda*. http://www.internet-of-things-research.eu/pdf/IoT_

Cluster_Strategic_Research_Agenda_2011.pdf, 2011. aufgerufen am 04.11.2012.

[FJK+05] FRANKLIN, MICHAEL J., SHAWN R. JEFFERY, SAILESH KRISHNAMURTHY, FREDERICK REISS, SHARIQ RIZVI, EUGENE WU 0002, OWEN COOPER, ANIL EDAKKUNNI und WEI HONG: *Design Considerations for High Fan-In Systems: The HiFi Approach*. In: *Seconds Biennal Conference on Innovative Data Systems Research, Asilomar, California*, Seiten 290–304, 2005.

[FLZ11] FAN, JU, GUOLIANG LI und LIZHU ZHOU: *Interactive SQL query suggestion: Making databases user-friendly*. In: *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, Seiten 351–362, 2011.

[FMF09] FAIRGRIEVE, SCOTT M., JOHN A. MAKUCH und STEFAN R. FALKE: *PULSENet - An implementation of Sensor Web standards*. In: *Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems*, CTS '09, Seiten 64–75, Washington, DC, USA, 2009. IEEE Computer Society.

[Fol11] FOLLEHER, PASCAL: *Verteilte Klassifizierung von Sensordaten: Ein agentenbasierter Ansatz*. Bachelorarbeit, Universität Hamburg, Fachbereich Informatik, Vogt-Koelln-Str. 30, 22527 Hamburg, Germany, June 2011.

[Fos12] FOSSTRAK: OPEN SOURCE RFID PLATFORM: *Homepage*. www.fosstrak.org, April 2012. aufgerufen am 24.04.2012.

[Fra10] FRADEN, J.: *Handbook of Modern Sensors: Physics, Designs, and Applications*. Springer, 2010.

[FRL06] FOK, CHIEN-LIANG, GRUIA-CATALIN ROMAN und CHENYANG LU: *Agilla: A Mobile Agent Middleware for Sensor Networks*. Technischer Bericht WUCSE-2006-16, Washington University in St. Louis Dept. of Comp.Sc.and Eng., 2006.

[FRL07] FLOERKEMEIER, CHRISTIAN, CHRISTOF RODUNER und MATTHIAS LAMPE: *RFID Application Development With the Accada Middleware Platform*. IEEE Systems Journal, 1(2):82–94, 2007.

[FS06] FERSTL, O.K. und E.J. SINZ: *Grundlagen der Wirtschaftsinformatik*. Oldenbourg, 2006.

[FSB06] FRISCHBIER, SEBASTIAN, KAI SACHS und ALEJANDRO BUCH-

MANN: *Evaluating RFID Infrastructures*. In: *2. Workshop RFID Intelligente Funketiketten - Chancen und Herausforderungen*, July 2006.

[FTR+10] FÜLÖP, LAJOS JENO, GABRIELLA TÓTH, RÓBERT RÁCZ, JÁNOS PÁNCZÉL, TAMÁS GERGELY, ARPÁD BESZÉDES und LÓRÁNT FARKAS: *Survey on complex event processing and predictive analytics*. Technischer Bericht, University of Szeged, 2010.

[Fuc09] FUCHSS, T.: *Mobile Computing: Grundlagen und Konzepte für mobile Anwendungen*. Hanser, Carl, 2009.

[GB00] GARG, VISHAL und N.K. BANSAL: *Smart occupancy sensors to reduce energy consumption*. Energy and Buildings, 32(1):81 – 87, 2000.

[GB08] GEORGOULAS, DIMITRIOS und KEITH BLOW: *Intelligent Mobile Agent Middleware for Wireless Sensor Networks: A Real Time Application Case Study*. The Fourth Advanced International Conference on Telecommunications, Athens, Greece, 0:95–100, 2008.

[GBMP12] GUBBI, JAYAVARDHANA, RAJKUMAR BUYYA, SLAVEN MARUSIC und MARIMUTHU PALANISWAMI: *Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions*. CoRR, abs/1207.0203, 2012.

[GEG09] GAMMA, E. und R.H.J.V.R.J. ERICH GAMMA: *Entwurfsmuster*. Programmer's choice. Addison Wesley Verlag, 2009.

[Gei11] GEIHS, KURT: *Die Anwendung denkt mit auf Schritt und Tritt*, Kapitel 21, Seiten 319–329. Smart Mobile Apps Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse, Verclas, Stephan; Linnhoff-Popien, Claudia (Hrsg.). Springer, 2011.

[GHJV11] GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Programmer's Choice. Addison Wesley Verlag, 2011.

[GHKW08] GRÖTKER, THORSTEN, ULRICH HOLTMANN, HOLGER KEDING und MARKUS WLOKA: *The Developer's Guide to Debugging*. Springer Publishing Company, Incorporated, 1 Auflage, 2008.

[GKK+03] GIBBONS, P.B., B. KARP, Y. KE, S. NATH und SRINIVASAN SESHAN: *IrisNet: An Architecture for a Worldwide Sensor Web*. Pervasive Computing, IEEE, 2(4):22 – 33, oct.-dec. 2003.

[Goo12a] GOOGLE INC.: *Google Cloud Messaging*. http://developer.android.com/guide/google/gcm/, August 2012. aufgerufen am 07.08.2012.

[Goo12b] GOOGLE INC.: *Google Cloud Platform*. http://cloud.google.com/, September 2012. aufgerufen am 26.09.2012.

[Goo12c] GOOGLE INC.: *Google Goggles*. http://www.google.com/mobile/goggles/, Mai 2012. aufgerufen am 09.05.2012.

[Goo12d] GOOGLE INC.: *Google Protocol Buffers*. http://code.google.com/p/protobuf/, August 2012. aufgerufen am 07.08.2012.

[Gra13] GRAYSTONE INDUSTRIES: *EnviroNet - Self Powered Wireless Sensor Network Products*. http://www.graystone-ind.com/products.html, June 2013. aufgerufen am 10.06.2013.

[GRHY09] GUALTIERI, MIKE, JOHN R. RYMER, RANDY HEFFNER und WALLIS YU: *The Forrester Wave: Complex Event Processing (CEP) Platforms, Q3 2009*. Research Document, August 2009.

[GRL+08] GURGEN, LEVENT, CLAUDIA RONCANCIO, CYRIL LABBÉ, ANDRÉ BOTTARO und VINCENT OLIVE: *SStreaMWare: a service oriented middleware for heterogeneous sensor data management*. In: *Proceedings of the 5th international conference on Pervasive services*, ICPS '08, Seiten 121–130, New York, NY, USA, 2008. ACM.

[GSS10] GROHE, STEFAN, CHRISTOPH SCHLAMEUSS und RALF SOMMER: *Performancevergleich von CEP-Engines*. Technischer Bericht, Universitaetsbibliothek der Universitaet Stuttgart, Stuttgart, 2010.

[Gue11] GUENTHER, RICHARD: *Integration ereignisbasierter Middleware-Systeme in kontextbasierte Geschaeftsprozesse*. Diplomarbeit, University of Hamburg, 2011.

[Gut01] GUTTMAN, ERIK: *Autoconfiguration for IP Networking: Enabling Local Communication*. IEEE Internet Computing, 5:81–86, 2001.

[GW10] GROFF, JAMES und PAUL WEINBERG: *SQL The Complete Reference, 3rd Edition*. McGraw-Hill, Inc., New York, NY, USA, 2010.

[HA07] HARTMANN, MELANIE und GERHARD AUSTALLER: *Context Models and Context Awareness*. In: MÜHLHAUSER, MAX und IRYNA GUREVYCH (Herausgeber): *Handbook of Research on Ubiquitous Com-*

puting Technology for Real Time Enterprises, Seiten 235–256. Idea Group Publishing, Dezember 2007.

[Hai13] HAIFA, IBM RESEARCH: *IBM Active Middleware Technology*. https://www.research.ibm.com/haifa/dept/services/soms_ebs_tech.html, January 2013. aufgerufen am 23.01.2013.

[HAR10] HART COMMUNICATION FOUNDATION: *WirelessHART Technology*. http://www.hartcomm.org/protocol/wihart/wireless_technology.html, August 2010. aufgerufen am 26.08.2010.

[HC09] HU, F. und X. CAO: *Wireless Sensor Networks: Principles and Practice*. Auerbach Publications, 2009.

[Hec13] HECTOR: *Hector - High-level Java Client for Cassandra*. https://github.com/hector-client/hector, May 2013. aufgerufen am 24.Mai 2013.

[Hen03] HENRICKSEN, KAREN: *A framework for context-aware pervasive computing applications*. Doktorarbeit, School of Information Technology and Electrical Engineering, University of Queensland, 2003.

[HHM10] HUMMEL, KARIN ANNA, ANDREA HESS und HARALD MEYER: *Mobilitaet im Future Internet*. Informatik-Spektrum, 33:143–159, 2010.

[HI05] HENRICKSEN, KAREN und JADWIGA INDULSKA: *Developing context-aware pervasive computing applications: models and approach*. Journal of Pervasive and Mobile Computing, 2(1):37–64, 2005.

[HKS08] HALLER, STEPHAN, STAMATIS KARNOUSKOS und CHRISTOPH SCHROTH: *The Internet of Things in an Enterprise Context*. In: *Future Internet Symposium 2008*, Seiten 14–28, 2008.

[HLFP99] HUANG, ANDREW C., BENJAMIN C. LING, ARMANDO FOX und SHANKAR PONNEKANTI: *Pervasive computing: What is it good for*. In: *In Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access*, Seiten 84–91. ACM Press, 1999.

[HMCP04] HEINZELMAN, WENDI BETH, AMY L. MURPHY, HERVALDO S. CARVALHO und MARK A. PERILLO: *Middleware to Support Sensor Network Applications*. IEEE Network, 18(1):6–14, 2004.

[HPST09] HENSON, CORY A., JOSH K. PSCHORR, AMIT P. SHETH und KRISHNAPRASAD THIRUNARAYAN: *SemSOS: Semantic sensor Observation Service*. In: *Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems*, CTS '09, Seiten 44–53, Washington, DC, USA, 2009. IEEE Computer Society.

[HV07] HENNING, P.A. und H. VOGELSANG: *Handbuch Programmiersprachen*. Hanser, 2007.

[IBK09] IBACH, PETER, DIRK BADE und STEFFEN KUNZ: *Smart Items in Ereignisgesteuerten Prozessketten*. In: *Verwaltung, Analyse und Bereitstellung kontextbasierter Informationen, Workshop, 39. GI-Jahrestagung, 2009, Luebeck, Germany*, GI Lecture Notes in Informatics. GI - Gesellschaft fuer Informatik, September 2009.

[IDC12] IDC: *The Digital Universe in 2020*. http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf, December 2012. aufgerufen am 16.05.2013.

[Ins12] INSTITUT, BIBLIOGRAPHISCHES: *Duden – Deutsches Universalwoerterbuch online*. `http://www.duden.de`, April 2012.

[Ins13] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE): *IEEE 802.15: Wireless Personal Area Neworks*. `http://standards.ieee.org/getieee802/802.15.html`, June 2013. aufgerufen am 10.06.2013.

[Int99] INTERNET ENGINEERING TASK FORCE (IETF): *Hypertext Transfer Protocol - HTTP/1.1*. `http://tools.ietf.org/html/rfc2616`, June 1999.

[Int03] INTERNET ENGINEERING TASK FORCE (IETF): *Internet Message Access Protocol - Version 4rev1*. `http://tools.ietf.org/html/rfc3501`, March 2003.

[Int05a] INTERNET ENGINEERING TASK FORCE (IETF): *Uniform Resource Identifier*. `http://tools.ietf.org/html/rfc3986`, January 2005.

[Int05b] INTERNET ENGINEERING TASK FORCE (IETF): *Universally Unique Identifier*. `http://www.ietf.org/rfc/rfc4122.txt`, July 2005.

[Int10a] INTEGRASOFT L.L.C.: *CEP Service Cloud Virtualized CEP Services Integrated with GigaSpaces XAP*. White Paper, April 2010. aufgerufen am 05.07.2013.

[Int10b] INTERNATIONAL SOCIETY FOR AUTOMATION (ISA): *ISA-100.11a: Wireless systems for industrial automation: Process control and related applications*. http://www.isa.org, August 2010.

[Int10c] INTERNET ENGINEERING TASK FORCE (IETF): *IPv6 over Low power WPAN*. http://datatracker.ietf.org/wg/6lowpan/charter, August 2010.

[Int12] INTERNET WORLD STATS: *World Internet Usage and Population Statistics*. http://www.internetworldstats.com/stats.htm, June 2012. aufgerufen am 04.11.2012.

[Int13] INTERNET ENGINEERING TASK FORCE (IETF): *IP Mobility Support for IPv4*. http://datatracker.ietf.org/wg/mobileip/charter/, April 2013.

[JBP+11] JANDER, KAI, LARS BRAUBACH, ALEXANDER POKAHR, WINFRIED LAMERSDORF und KARL-JOSEF WACK: *Goal-oriented Processes with GPMN*. International Journal on Artificial Intelligence Tools (IJAIT), 20(6):1021–1041, 12 2011.

[JBS09] JIRKA, SIMON, ARNE BROERING und CHRISTOPH STASCH: *Discovery Mechanisms for the Sensor Web*. Sensors, 9(4):2661–2681, 2009.

[JC09] JAMES, ANNE. und JOSHUA. COOPER: *Challenges for Database Management in the Internet of Things*. IETE Technical Review, 26(5):320–329, 2009.

[JLSU87] JOYCE, JEFFREY, GREG LOMOW, KONRAD SLIND und BRIAN UNGER: *Monitoring distributed systems*. ACM Trans. Comput. Syst., 5(2):121–150, März 1987.

[JNSH10] JIRKA, S., D. NUEST, J. SCHULTE und F. HOUBIE: *Integrating the OGC Sensor Web Enablement Framework into the OGC Catalogue*. In: *In Proceedings of the 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010)*, Como, Italy, 2010.

[Joh93] JOHANSSON, H.J.: *Business Process Reengineering: Breakpoint Strategies for Market Dominance*. Business / Wiley. Wiley, 1993.

[Jös08] JÖST, MATTHIAS: *Adapting to the User*. In: *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, Seiten 282–296. IGI Global, 2008.

[JS12] JU, DEHUA und BEIJUN SHEN: *Internet of Knowledge Plus Knowledge Cloud - A Future Education Ecosystem*. {IERI} Procedia, 2(0):331 – 336, 2012. International Conference on Future Computer Supported Education, August 22-23, 2012, Fraser Place Central - Seoul.

[JSB+09] JANOWICZ, KRZYSZTOF, SVEN SCHADE, ARNE BROERING, CARSTEN KESSLER und CHRISTOPH STASCH: *A Transparent Semantic Enablement Layer for the Geospatial Web*. In: *Terra Cognita (Terra2009)*, October 2009.

[Kal12] KALINOWSKI, JULIAN: *Realisierung der Ad-hoc-Kommunikation zwischen aktiven Komponenten auf mobilen Geraeten*. Diplomarbeit, Universitaet Hamburg, 2012.

[KBFZ11] KUNZ, S., D. BADE, B. FABIAN und H. ZIEKOW: *Smarter Workflows for the Internet of Things - Service Comparison and System Architecture*. unpublished, 2011.

[KC03] KEPHART, JEFFREY O. und DAVID M. CHESS: *The Vision of Autonomic Computing*. Computer, 36(1):41–50, Januar 2003.

[KCKT08] KÜRSCHNER, CHRIS, COSMIN CONDEA, OLIVER KASTEN und FRÉDÉRIC THIESSE: *Discovery service design in the EPCglobal network: towards full supply chain visibility*. In: *Proceedings of the 1st international conference on The internet of things*, IOT'08, Seiten 19–34, Berlin, Heidelberg, 2008. Springer-Verlag.

[KE11] KEMPER, ALFONS und ANDRÉ EICKLER: *Datenbanksysteme - Eine Einführung, 8. Auflage*. Oldenbourg, 2011.

[Kei02] KEIM, DANIEL A.: *Datenvisualisierung und Data Mining*. Datenbank-Spektrum, 2:30–39, 2002.

[KFZB11] KUNZ, S., B. FABIAN, H. ZIEKOW und D. BADE: *From Smart Objects to Smarter Workflows – An Architectural Approach*. In: *Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, Seiten 194 –203, 29 2011-sept. 2 2011.

[KGZ07] KANSAL, AMAN, MICHEL GORACZKO und FENG ZHAO: *Building a sensor network of mobile phones*. In: *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, Seiten 547–548, New York, NY, USA, 2007. ACM.

[KH07] KOSKELA, MIKA und JYRKI HAAJANEN: *Business Process Modeling and Execution: Tools and technologies report for SOAMeS pro-*

ject. Technischer Bericht, VTT Technical Research Centre of Finland, 2007.

[KLL09] KO, RYAN K.L., STEPHEN S.G. LEE und ENG WAH LEE: *Business Process Management (BPM) standards: A survey*. Business Process Management journal, 15(5), 2009.

[KO04] KAENAMPORNPAN, MANASAWEE und EAMONN O'NEILL: *Modelling context: an Activity Theory approach*. In: *2nd European Symposium on Ambient Intelligence, EUSAI 2004, Eindhoven, The Netherlands*, Band 3295 der Reihe *LNCS*, Seiten 367–374. Springer, 2004.

[KPC06] KOFOD-PETERSEN, ANDERS und JÖRG CASSENS: *Using Activity Theory to Model Context Awareness*. In: ROTH-BERGHOFER, THOMAS R., STEFAN SCHULZ und DAVID B. LEAKE (Herausgeber): *Modeling and Retrieval of Context: MRC 2005, Revised Selected Papers*, Band 3946 der Reihe *LNCS*, Seiten 1–17, Edinburgh, 2006. Springer.

[KS12] KLINGER, EVAN und DAVID STARKWEATHER: *pHash - The open source perceptual hash library*. `http://www.phash.org`, September 2012. aufgerufen am 11.09.2012.

[Kun11] KUNZ, S.: *Event Orientation and Secure Federation for Object Lifecycle Management*. Logos Verlag, 2011.

[LAE+10] LEHMANN, DIRK JOACHIM, GEORGIA ALBUQUERQUE, MARTIN EISEMANN, ANDRADA TATU, DANIEL A. KEIM, HEIDRUN SCHUMANN, MARCUS MAGNOR und HOLGER THEISEL: *Visualisierung und Analyse multidimensionaler Datensaetze*. Informatik-Spektrum, 33(6):589–600, Dezember 2010.

[Lam78] LAMPORT, LESLIE: *Time, clocks, and the ordering of events in a distributed system*. Commun. ACM, 21(7):558–565, Juli 1978.

[Lan13] LANDAU, ERAN: *Astyanax - High-level Java Client for Cassandra*. `https://github.com/Netflix/astyanax`, January 2013. aufgerufen am 29.01.2013.

[LCT05] LIANG, STEVE H. L., ARIE CROITORU und C. VINCENT TAO: *A distributed geospatial infrastructure for Sensor Web*. Comput. Geosci., 31(2):221–231, March 2005.

[Le11] LE, HAI-MINH: *Visuelle Unterstuetzung bei der EPL-Anfragestellung fuer Sensordatenstroeme*. Diplomarbeit, Uni-

versitaet Hamburg, Fachbereich Informatik, January 2011.

[Lia08] LIANG, S.H.L.: *A new Peer-to-Peer-based Interoperable Spatial Sensor Web Architecture*. In: *The ISPRS XXIth Congress*, Beijing, China, July 2008.

[LLR12] LLRP TOOLKIT PROJECT: *Homepage*. `www.llrp.org`, April 2012. aufgerufen am 24.04.2012.

[LLRS97] LAKSHMANAN, LAKS V. S., NICOLA LEONE, ROBERT ROSS und V. S. SUBRAHMANIAN: *ProbView: a flexible probabilistic database system*. ACM Trans. Database Syst., 22(3):419–469, September 1997.

[LSI10] LAZAKIDOU, A.A., K.M. SIASSIAKOS und K. IOANNOU: *Wireless Technologies for Ambient Assisted Living and Healthcare: Systems and Applications*. IGI Global, 2010.

[LTC04] LIANG, S. H. L., V. TAO und A. CROITORU: *Sensor Web and GeoSWIFT - An Open Geospatial Sensing Service*. In: *International Society for Photogrammetry and Remote Sensing XXth Congress - GeoImagery Bridging Continents*, Seiten 12–23, 2004.

[Luc02] LUCKHAM, DAVID: *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, May 2002.

[Luc06] LUCKHAM, DAVID: *What's the Difference between ESP and CEP?* `http://www.complexevents.com/2006/08/01/what%E2%80%99s-the-difference-between-esp-and-cep/`, August 2006. aufgerufen am 23.01.2013.

[LWZ04] LAW, YAN-NEI, HAIXUN WANG und CARLO ZANIOLO: *Query languages and data models for database sequences and data streams*. In: *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, Seiten 492–503. VLDB Endowment, 2004.

[Mat89] MATTERN, FRIEDEMANN: *Virtual Time and Global States of Distributed Systems*. In: AL., COSNARD M. ET (Herausgeber): *Proc. Workshop on Parallel and Distributed Algorithms*, Seiten 215–226, North-Holland / Elsevier, 1989. (Reprinted in: Z. Yang, T.A. Marsland (Eds.), "Global States and Time in Distributed Systems", IEEE, 1994, pp. 123-133.).

[May12] MAYBMS: *A Database Management System for Uncertain and Probabilistic Data*. http://www.cs.cornell.edu/database-OLD/maybms/, June 2012. aufgerufen am 01.06.2012.

[MC11] MARGARA, ALESSANDRO und GIANPAOLO CUGOLA: *Processing flows of information: from data stream to complex event processing*. In: *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS '11, Seiten 359–360, New York, NY, USA, 2011. ACM.

[MCF+08] MANDL, D., P. CAPPELAERE, S. FRYE, R. SOHLBERG, L. ONG, S. CHIEN, D. TRAN, A. DAVIES, S. FALKE, S. KOLITZ, P. ZHAO, L. DI, N. CHEN, G. YU, D. SMITHBAUER, S. UNGAR, L. DEREZINSKI und M. BOTTS: *Sensor Web 2.0: Connecting Earth's Sensors via the Internet*. In: *Proceedings of NASA Earth Science Technology Conference*, Adelphi, MD, USA, June 2008. aufgerufen am 18.04.2012.

[Mel10] MELZER, I.: *Service-Orientierte Architekturen Mit Web Services: Konzepte - Standards - Praxis*. Spektrum Akademischer Verlag GmbH, 2010.

[MF10] MATTERN, FRIEDEMANN und CHRISTIAN FLÖRKEMEIER: *Vom Internet der Computer zum Internet der Dinge*. Informatik Spektrum, 33(2):107–121, 2010.

[MFHH05] MADDEN, SAMUEL R., MICHAEL J. FRANKLIN, JOSEPH M. HELLERSTEIN und WEI HONG: *TinyDB: an acquisitional query processing system for sensor networks*. ACM Trans. Database Syst., 30(1):122–173, März 2005.

[MG04] MANDL, D. und NASA G. GASFC: *Experimenting with sensor Webs using Earth Observing*. In: *IEEE Aerospace Conference Proceedings*, Band 1, Seiten 176–183, 2004.

[Mic12] MICROSOFT CORPORATION: *Windows Azure*. www.windowsazure.com, September 2012. aufgerufen am 26.09.2012.

[Moz98] MOZER, M.: *The neural network house: An environment that adapts to its inhabitants*. In: *Proceedings of the American Association for Artificial Intelligence*, Seiten 110–114, 1998.

[MRT06] MCFERREN, G., S. ROOS und A. TERHORST: *Fire Alerts on the Geospatial Semantic Web*. In: *Workshop Terra Cognita 2006*, 2006.

[MS06] MOODLEY, DESHENDRAN und INGO SIMONIS: *A New Architecture*

for the Sensor Web: The SWAP Framework. In: *In Proceedings of the 5th International Semantic Web Conference (ISWC) Athens, Georgia*, 2006.

[MSB11] MYERS, G.J., C. SANDLER und T. BADGETT: *The Art of Software Testing*. John Wiley & Sons, 2011.

[MSPS08] MOE, K., S. SMITH, G. PRESCOTT und R. SHERWOOD: *Sensor Web Technologies for NASA Earth Science*. In: *Aerospace Conference, 2008 IEEE*, Seiten 1 –7, march 2008.

[MTJ+10] MILI, HAFEDH, GUY TREMBLAY, GUITTA BOU JAOUDE, ÉRIC LEFEBVRE, LAMIA ELABED und GHIZLANE EL BOUSSAIDI: *Business process modeling languages: Sorting through the alphabet soup*. ACM Computing Surveys, 43(1):4:1–4:56, Dezember 2010.

[MTS+06] MOODLEY, D., A. TERHORST, I. SIMONIS, G. MCFERREN und F. VAN DEN BERGH: *Using the sensor web to detect and monitor the spread of wild fires*. In: *2nd International symposium on geoinformation for disaster management, Goa, India,*, September 2006.

[Mus12] MUSICG: *Lightweight Java API for audio analysing*. `http://code.google.com/p/musicg/`, September 2012. aufgerufen am 11.09.2012.

[MW12] MERRIAM-WEBSTER: *Merriam-Webster Online Dictionary*. http://www.m-w.com, April 2012.

[Nat12a] NATIONAL ICT AUSTRALIA LTD.: *NICTA Open SensorWeb Architecture*. `http://www.nicta.com.au/research/archive/business_areas/environment_management/nicta_open_sensorweb_architecture`, Januar 2012. aufgerufen am 03.01.2012.

[Nat12b] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *IEEE 1451 Smart Transducer Interface Standard*. `http://www.nist.gov/el/isd/ieee/ieee1451.cfm`, Mai 2012.

[NEC09] NEC EUROPE INC.: *Siafu - An Open Source Context Simulator*. `http://siafusimulator.sourceforge.net`, July 2009.

[Nit09] NITTEL, SILVIA: *A Survey of Geosensor Networks: Advances in Dynamic Environmental Monitoring*. Sensors, 9(7):5664–5678, 2009.

[Obj09] OBJECT MANAGEMENT GROUP, OMG: *Business Process Model and Notation (BPMN) 1.2*. http://www.omg.org/spec/BPMN/1.2/, November 2009. aufgerufen am 15.02.2009.

[Obj11a] OBJECT MANAGEMENT GROUP, OMG: *Business Process Model and Notation (BPMN) 2.0*. http://www.omg.org/spec/BPMN/2.0/, Januar 2011. aufgerufen am 08.08.2012.

[Obj11b] OBJECT MANAGEMENT GROUP, OMG: *Unified Modeling Language 2*. http://www.omg.org/spec/UML/2.4.1/, August 2011. aufgerufen am 08.08.2012.

[Obj12] OBJECT MANAGEMENT GROUP, OMG: *BPMN Implementers*. http://www.bpmn.org/#tabs-implementers, August 2012.

[OMG13] OBJECT MANAGEMENT GROUP, OMG: *Interface Definition Language 3.5*. http://www.omg.org/spec/IDL35/, February 2013. aufgerufen am 01.07.2013.

[Onl99] ONLINE, BUSINESS WEEK: *21 Ideas for the 21st Century - The Earth will don an Electronic Skin*. http://www.businessweek.com/1999/99_35/b3644024.htm, August 1999. aufgerufen am 16.05.2013.

[Ope07a] OPEN GEOSPATIAL CONSORTIUM: *OpenGIS Catalogue Services Specification 2.0.2*. http://www.opengeospatial.org/standards/specifications/catalog, Februar 2007. aufgerufen am 03.01.2012.

[Ope07b] OPEN GEOSPATIAL CONSORTIUM: *OpenGIS Sensor Model Language (SensorML)*. http://www.opengeospatial.org/standards/sensorml, July 2007. aufgerufen am 15.02.2013.

[Ope07c] OPEN GEOSPATIAL CONSORTIUM: *OpenGIS Web Processing Service*. http://www.opengeospatial.org/standards/wps, June 2007. aufgerufen am 22.02.2013.

[Ope08a] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Event Service Interface Specification*. http://portal.opengeospatial.org/files/?artifact_id=29576, December 2008. aufgerufen am 02.01.2013.

[Ope08b] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Web Enablement Architecture*. http://www.opengeospatial.org/projects/groups/sensorwebdwg, August 2008. aufgerufen am 01.11.2011.

[Ope10a] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Instance Registry Discussion Paper*. http://www.opengeospatial.org/standards/dpd, October 2010. aufgerufen am 10.05.2012.

[Ope10b] OPEN GEOSPATIAL CONSORTIUM: *OpenGIS Filter Encoding 2.0 Encoding Standard (FES)*. http://www.opengeospatial.org/standards/filter, November 2010. aufgerufen am 08.01.2013.

[Ope11] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Planning Service Implementation Standard*. http://www.opengeospatial.org/standards/sps, March 2011. aufgerufen am 09.05.2012.

[Ope12a] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Observation Service Interface Standard*. http://www.opengeospatial.org/standards/sos, April 2012. aufgerufen am 09.05.2012.

[Ope12b] OPEN GEOSPATIAL CONSORTIUM: *OGC Sensor Web Enablement*. http://www.opengeospatial.org/ogc/markets-technologies/swe, November 2012. aufgerufen am 16.11.2012.

[Ope12c] OPENSTACK.ORG: *OpenStack*. http://www.openstack.org/, September 2012. aufgerufen am 26.09.2012.

[Ope13] OPEN HANDSET ALLIANCE: *Android Overview*. http://www.openhandsetalliance.com/android_overview.html, February 2013. aufgerufen am 26.02.2013.

[Ora12] ORACLE: *Oracle PL/SQL*. http://www.oracle.com/technetwork/database/features/plsql/, Mai 2012. aufgerufen am 10.05.2012.

[Ora13a] ORACLE: *Java Logging Technology*. http://docs.oracle.com/javase/6/docs/technotes/guides/logging/index.html, January 2013. aufgerufen am 15.01.2013.

[Ora13b] ORACLE: *Java Speech API*. http://www.oracle.com/technetwork/java/jsapifaq-135248.html, June 2013. aufgerufen am 10.06.2013.

[Org07] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS, OASIS: *Web Services Business Process Execution Language*. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, April 2007. aufgerufen am 08.08.2012.

[Org12] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS, OASIS: *UDDI Version 3.0.2 Specification*. `http://uddi.org/pubs/uddi_v3.htm`, July 2012. aufgerufen am 06.07.2012.

[PAF09] PUNGILA, CIPRIAN., OVIDIU. ARITONI und TEODOR-FLORIN. FORTIS: *Benchmarking Database Systems for the Requirements of Sensor Readings*. IETE Technical Review, 26(5):342–349, 2009.

[Par12] PARROT SA: *AR.Drone Homepage*. `http://ardrone2.parrot.com/`, June 2012. aufgerufen am 14.06.2012.

[Pas98] PASCOE, MR. JASON: *Adding Generic Contextual Capabilities to Wearable Computers*. In: *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, ISWC '98, Washington, DC, USA, 1998. IEEE Computer Society.

[PB11] POKAHR, ALEXANDER und LARS BRAUBACH: *Active Components: A Software Paradigm for Distributed Systems*. In: *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2011)*, 2011.

[PB13] POKAHR, ALEXANDER und LARS BRAUBACH: *The Active Components Approach for Distributed Systems Development*. International Journal of Parallel, Emergent and Distributed Systems, 2013.

[Pel03] PELTZ, C.: *Web services orchestration and choreography*. Computer, 36(10):46 – 52, October 2003.

[PHPS10] PSCHORR, JOSHUA, CORY HENSON, HARSHAL PATNI und AMIT SHETH: *Sensor Discovery on Linked Data*. Group, 2010. aufgerufen ueber `http://www.mendeley.com/research/sensor-discovery-linked-data/`.

[PHR+12] PARAISO, FAWAZ, GABRIEL HERMOSILLO, ROMAIN ROUVOY, PHILIPPE MERLE und LIONEL SEINTURIER: *A Middleware Platform to Federate Complex Event Processing*. IEEE 16th International Enterprise Distributed Object Computing Conference, 0:113–122, 2012.

[PL05] PAHLAVAN, K. und A.H. LEVESQUE: *Wireless information networks*. Wiley series in telecommunications and signal processing. John Wiley, 2005.

[PM87] P. MOCKAPETRIS, NETWORK WORKING GROUP, IETF: *RFC 1034,*

Domain Names - Concepts and Facilities. http://tools.ietf.org/html/rfc1034, November 1987.

[PMHY07] PARK, JOONSEOK, MIKYEONG MOON, SEONGJIN HWANG und KEUNHYUK YEOM: *CASS: A Context-Aware Simulation System for Smart Home*. ACIS International Conference on Software Engineering Research, Management and Applications, 0:461–467, 2007.

[Pos12] POSTGRESQL: *PL/pgSQL - SQL Procedural Language*. http://www.postgresql.org/docs/current/static/plpgsql.html, Mai 2012. aufgerufen am 10.05.2012.

[PrD12] PRDB: *Managing and Querying Large-scale Uncertain Databases*. http://www.cs.umd.edu/~amol/PrDB/, June 2012. aufgerufen am 01.06.2012.

[Pro13] PROJECT, OPENINTENTS: *Sensor Simulator for simulating sensor data in real time*. http://code.google.com/p/openintents/wiki/SensorSimulator, July 2013. aufgerufen am 01.07.2013.

[PSB03] PIETZUCH, PETER R., BRIAN SHAND und JEAN BACON: *A framework for event composition in distributed systems*. In: *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, Seiten 62–82, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[PSM+03] PRIEDE, I.G., M. SOLAN, J. MIENERT, R. PERSON, T.C.E. VAN WEERING, O. PFANNKUCHE, N. O'NEILL, A. TSELEPIDES, L. THOMSEN, P. FAVALI, F. GASPARONI, N. ZITELLINI, C. MILOT, H.W. GERBER, J.M.A. DE MIRANDA und M. KLAGES: *ESONET-European sea floor observatory network*. In: *The 3rd International Workshop on Scientific Use of Submarine Cables and Related Technologies*, Seiten 263 – 265, june 2003.

[PZL08] PAUTASSO, CESARE, OLAF ZIMMERMANN und FRANK LEYMANN: *Restful web services vs. "big" web services: making the right architectural decision*. In: *Proceedings of the 17th international conference on World Wide Web*, WWW '08, Seiten 805–814, New York, NY, USA, 2008. ACM.

[Qua13a] QUALITY OPEN SOFTWARE: *Logback*. http://logback.qos.ch, April 2013. aufgerufen am 29.04.2013.

[Qua13b] QUALITY OPEN SOFTWARE: *Simple Logging Facade (SLF4j)*. http://www.slf4j.org/, April 2013. aufgerufen am 29.04.2013.

[RD01] ROWSTRON, ANTONY und PETER DRUSCHEL: *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*. In: GUERRAOUI, RACHID (Herausgeber): *Middleware 2001*, Band 2218 der Reihe *Lecture Notes in Computer Science*, Seiten 329–350. Springer Berlin Heidelberg, 2001.

[Res12] RESEARCH IN MOTION LTD.: *BlackBerry Push Service*. `http://developer.blackberry.com/services/push/`, August 2012. aufgerufen am 07.08.2012.

[RG95] RAO, ANAND S. und MICHAEL P. GEORGEFF: *BDI Agents: From Theory to Practice*. In: *ICMAS*, Seiten 312–319, 1995.

[RH08] ROWLEY, J.E. und RICHARD J. HARTLEY: *Organizing knowledge: an introduction to managing access to information*. Ashgate Publishing Company, 2008.

[Rif12] RIFIDI: *Homepage*. `www.rifidi.org`, April 2012. aufgerufen am 24.04.2012.

[Rot05] ROTH, JOERG: *Mobile Computing - Grundlagen, Technik, Konzepte*. dpunkt.verlag, Heidelberg, 2nd Auflage, 2005.

[Rou08] ROUSSOS, GEORGE: *Networked RFID: Systems, Software and Services*. Springer Publishing Company, Inc., 2008.

[RR06] ROBERTSON, S. und J. ROBERTSON: *Mastering the requirements process*. ACM Press Books. Addison-Wesley, 2006.

[Sal12] SALESFORCE.COM INC.: *Salesforce Cloud*. `http://www.salesforce.com`, September 2012. aufgerufen am 26.09.2012.

[SAT+99] SCHMIDT, ALBRECHT, KOFI ASANTE AIDOO, ANTTI TAKALUOMA, URPO TUOMELA, KRISTOF VAN LAERHOVEN und WALTER VAN DE VELDE: *Advanced Interaction in Context*. In: *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, Seiten 89–101, London, UK, UK, 1999. Springer-Verlag.

[Sat10] SATOH, ICHIRO: *Mobile Agents*. In: NAKASHIMA, HIDEYUKI, HAMID AGHAJAN und JUANCARLOS AUGUSTO (Herausgeber): *Handbook of Ambient Intelligence and Smart Environments*, Seiten 771–791. Springer US, 2010.

[SaV+04] SOUTO, EDUARDO, GERMANO GUIMAR AES, GLAUCO VASCONCELOS, MARDOQUEU VIEIRA, NELSON ROSA und CARLOS FERRAZ: *A*

message-oriented middleware for sensor networks. In: *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, MPAC '04, Seiten 127–134, New York, NY, USA, 2004. ACM.

[SAW94] SCHILIT, BILL, NORMAN ADAMS und ROY WANT: *Context-Aware Computing Applications*. In: *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[SCH+05] SOBEIH, A., WEI-PENG CHEN, J.C. HOU, LU-CHUAN KUNG, N. LI, HYUK LIM, HUNG YING TYAN und HONGHAI ZHANG: *J-Sim: a simulation environment for wireless sensor networks*. In: *Proceedings of the 38th Annual Simulation Symposium, 2005*, Seiten 175–187, 2005.

[Sch07] SCHLOSSNAGLE, T.: *Scalable Internet Architectures*. Developer's Library. Developer's Library/Sams, 2007.

[Sch09] SCHAJEW, ALEX: *Entwurf eines Szenario-Simulators zur Unterstuetzung der Entwicklung von RFID-Anwendungen*. Diplomarbeit, University of Hamburg, 2009.

[Sch10] SCHOLL, H.J.: *E-Government: Information, Technology, and Transformation*. M. E. Sharpe Incorporated, 2010.

[SF12] SADALAGE, P.J. und M. FOWLER: *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education, 2012.

[SGCB05] SACHS, K., P. GUERRERO, M. CILIA und A. P. BUCHMANN: *RFID Seminar*. Technischer Bericht, Technische Universit¨at Darmstadt, 2005.

[SGM02] SZYPERSKI, C., D. GRUNTZ und S. MURER: *Component Software: Beyond Object-Oriented Programming*. Component Software Series. Prentice Hall, 2002.

[SH05] SINGH, M.P. und M.N. HUHNS: *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.

[SHG+08] STILLER, BURKHARD, DAVID HAUSHEER, JAN GERKE, PETER RACZ, CRISTIAN MORARIU und MARTIN WALDBURGER: *Accounting and Charging*. In: *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, Seiten 302–336. IGI Global, 2008.

[SL08] SONG, E.Y. und K.B. LEE: *STWS: A Unified Web Service for IEEE 1451 Smart Transducers*. Instrumentation and Measurement, IEEE Transactions on, 57(8):1749–1756, aug. 2008.

[SL09] SICILIA, MIGUEL-ÁNGEL und MILTIADIS D. LYTRAS (Herausgeber): *Metadata and Semantics, Post-proceedings of the 2nd International Conference on Metadata and Semantics Research, MTSR 2007, Corfu Island in Greece, 1-2 October 2007*. Springer, 2009.

[SLP04] STRANG, THOMAS und CLAUDIA LINNHOFF-POPIEN: *A Context Modeling Survey*. In: *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The 6th International Conference on Ubiquitous Computing, Nottingham / England*, 2004.

[SLZ08] SHENG, Q.Z., XUE LI und S. ZEADALLY: *Enabling Next-Generation RFID Applications: Solutions and Challenges*. Computer, 41(9):21 –28, sept. 2008.

[SMK+01] STOICA, ION, ROBERT MORRIS, DAVID KARGER, M. FRANS KAASHOEK und HARI BALAKRISHNAN: *Chord: A scalable peer-to-peer lookup service for internet applications*. SIGCOMM Comput. Commun. Rev., 31(4):149–160, August 2001.

[SMMP09] SCHULTZ-MØLLER, NICHOLAS POUL, MATTEO MIGLIAVACCA und PETER PIETZUCH: *Distributed complex event processing with query rewriting*. In: *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, Seiten 4:1–4:12, New York, NY, USA, 2009. ACM.

[SMZ07] SOHRABY, KAZEM, DANIEL MINOLI und TAIEB ZNATI: *Wireless Sensor Networks: Technology, Protocols, and Applications*. Wiley-Interscience, 2007.

[SN13] SANFILIPPO, SALVATORE und PIETER NOORDHUIS: *Redis Homepage*. `http://redis.io`, January 2013. aufgerufen am 29.01.2013.

[SOK11] SUCIU, D., D. OLTEANU und C. KOCH: *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Som12] SOMMERVILLE, I.: *Software Engineering*. International Computer Science Series. Addison-Wesley, 2012.

[Son09] SONATYPE COMPANY: *Maven: The Definitive Guide*. O'Reilly Media, 2009.

[Sou12] SOUNDHOUND INC.: *Instant Music Search and Discovery*. http://www.soundhound.com, Mai 2012. aufgerufen am 08.05.2013.

[SPL+04] SHNEIDMAN, JEFFREY, PETER PIETZUCH, JONATHAN LEDLIE, MEMA ROUSSOPOULOS, MARGO SELTZER und MATT WELSH: *Hourglass: An Infrastructure for Connecting Sensor Networks and Applications*. Technischer Bericht, Harvard, 2004.

[SSH10] SAAKE, G., K.U. SATTLER und A. HEUER: *Datenbanken – Konzepte und Sprachen*. mitp Professional. mitp/bhv, 2010.

[Sun02] SUN MICROSYSTEMS INC.: *Java Message Service*. http://www.oracle.com/technetwork/java/jms, April 2002. aufgerufen am 21.04.2012.

[Sun12] SUN MICROSYSTEMS INC.: *SunSPOT Homepage*. http://www.sunspotworld.com, June 2012.

[SWSVR05] SGROI, MARCO, ADAM WOLISZ, ALBERTO SANGIOVANNI-VINCENTELLI und JAN RABAEY: *A service-based universal application interface for ad hoc wireless sensor and actuator networks*. In: W. WEBER, J. RABAEY, E. AARTS (Herausgeber): *Ambient intelligence*. Springer Verlag, Berlin, 2005.

[Tan13] TAN, Y.K.: *Energy Harvesting Autonomous Sensor Systems: Design, Analysis, and Practical Implementation*. Taylor & Francis, 2013.

[Tiw11] TIWARI, S.: *Professional NoSQL*. Programmmer to programmer. Wiley, 2011.

[TS07] TANENBAUM, ANDREW S. und MAARTEN VAN STEEN: *Distributed Systems: Principles and Paradigms, 2nd Edition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[Twi13] TWITTER INC.: *Homepage*. http://twitter.com/, June 2013. aufgerufen am 10.06.2013.

[UHM11] UCKELMANN, DIETER, MARK HARRISON und FLORIAN MICHAHELLES: *An Architectural Approach Towards the Future Internet of Things*. In: *Architecting the Internet of Things*, Seiten 1–24. Springer, 2011.

[Uni12] UNITED STATES DEPARTMENT OF THE INTERIOR - BUREAU OF LAND MANAGEMENT: *Glossar: Mobile Computing*. http://

www.blm.gov/wo/st/en/prog/more/bea/Glossary.html, Oktober 2012. aufgerufen am 30.10.2012.

[UPn12] UPnP FORUM: *Universal Plug and Play*. http://www.upnp.org, August 2012. aufgerufen am 10.08.2012.

[VHP06] VAN HALTEREN, A. und P. PAWAR: *Mobile Service Platform: A Middleware for Nomadic Mobile Service Provisioning*. In: *Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006). IEEE International Conference on*, Seiten 292–299, 2006.

[VPG08] VIEGAS, V., M. PEREIRA und P. GIRAO: *A brief tutorial on the IEEE 1451.1 Standard - Part 13 in a series of tutorials in instrumentation and measurement*. Instrumentation Measurement Magazine, IEEE, 11(2):38 –46, april 2008.

[W3C04a] W3C: *OWL-S Semantic Markup for Web Services*. http://www.w3.org/Submission/OWL-S/, November 2004.

[W3C04b] W3C: *OWL Web Ontology Language Reference*. http://www.w3.org/TR/owl-ref/, February 2004. aufgerufen am 13.01.2013.

[W3C04c] W3C: *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/rdf-schema/, February 2004. aufgerufen am 13.01.2013.

[W3C04d] W3C: *Resource Description Framework (RDF)*. http://www.w3.org/RDF/, February 2004. aufgerufen am 13.01.2013.

[W3C06] W3C: *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. http://www.w3.org/TR/wsdl20/, March 2006. aufgerufen am 01.07.2013.

[W3C07] W3C: *Simple Object Access Protocol (SOAP) Version 1.2*. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/, April 2007. aufgerufen am 04.06.2012.

[W3C08] W3C: *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/, January 2008. aufgerufen am 13.01.2013.

[Wei91] WEISER, MARK: *The computer for the 21st century*. Scientific American, Special Issue(3):94–104, September 1991.

[Wei93] WEISER, MARK: *Some computer science issues in ubiquitous computing*. Commun. ACM, 36(7):75–84, Juli 1993.

[Whi97] WHITE, JAMES E.: *Mobile Agents*. In: BRADSHAW, JEFFREY M. (Herausgeber): *Software Agents*, Kapitel 19, Seiten 437–472. AAAI Press / The MIT Press, 1997.

[Wie07] WIEHLER, G.: *Mobility, Security und Web Services: Neue Technologien und Service-orientierte Architekturen fuer zukunftsweisende IT-Loesungen*. Wiley, 2007.

[Wik10] WIKIPEDIA: *Wireless Sensor Network*. http://en.wikipedia.org/wiki/Wireless_sensor_network, August 2010. aufgerufen am 26.08.2010.

[Wik12a] WIKIBOOKS: *The Computer Revolution / The Internet Revolution*. http://en.wikibooks.org/wiki/The_Computer_Revolution/The_Internet_%Revolution, November 2012. aufgerufen am 08.11.2012.

[Wik12b] WIKIPEDIA: *Business Process Execution Language*. http://en.wikipedia.org/wiki/Business_Process_Execution_Language, August 2012. aufgerufen am 09.08.2012.

[Wik12c] WIKIPEDIA: *Probabilistic Database*. http://en.wikipedia.org/wiki/Probabilistic_database, June 2012. aufgerufen am 01.06.2012.

[Wik12d] WIKIPEDIA: *Quantified Self*. http://de.wikipedia.org/wiki/Quantified_Self, October 2012. aufgerufen am 01.01.2012.

[Wik12e] WIKIPEDIA: *Siri (Software)*. http://de.wikipedia.org/wiki/Siri_%28Software%29, Mai 2012. aufgerufen am 01.05.2012.

[Wik13a] WIKIPEDIA: *Lego Mindstorms*. http://de.wikipedia.org/wiki/Lego_Mindstorms, February 2013. aufgerufen am 27.02.2013.

[Wik13b] WIKIPEDIA: *Wissen*. http://de.wikipedia.org/wiki/Wissen, April 2013. aufgerufen am 16.04.2013.

[Win01] WINOGRAD, TERRY: *Architectures for context*. Hum.-Comput. Interact., 16(2):401–419, Dezember 2001.

[Win05] WINTER, M.: *Methodische objektorientierte Softwareentwicklung: Eine Integration klassischer und moderner Entwicklungskonzepte*. Dpunkt-Verlag, 2005.

[Wis09] WISCHWEH, JAN: *Aktivitaetsorientierte Kontextadaption in mobilen Anwendungen*. Diplomarbeit, Universitaet Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Juli 2009.

[WJ95] WOOLDRIDGE, MICHAEL und NICHOLAS R. JENNINGS: *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10(2):115–152, 1995.

[WJ03] WESTKAEMPER, E. und L. JENDOUBI: *Smart Factories - Manufacturing Environments and Systems of the Future*. In: *Inproceedings of the 36th CIRP-International Seminar on Manufacturing Systems*, June 2003.

[Wor10] WORLD WIDE WEB CONSORTIUM (W3C): *XML Path Language (XPath) 2.0*. `http://www.w3.org/TR/xpath20/`, Dezember 2010. aufgerufen am 21.09.2011.

[Wor12] WORKFLOW MANAGEMENT COALITION, WFMC: *XML Process Definition Language (XPDL) 2.2 Specification*. `http://www.xpdl.org`, February 2012. aufgerufen am 09.08.2012.

[WXCJ98] WOLFSON, OURI, BO XU, SAM CHAMBERLAIN und LIQIN JIANG: *Moving Objects Databases: Issues and Solutions*. In: *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, SSDBM '98, Seiten 111–122, Washington, DC, USA, 1998. IEEE Computer Society.

[XMP12] XMPP STANDARDS FOUNDATION: *Homepage*. `http://xmpp.org/`, April 2012. aufgerufen am 20.04.2012.

[YC79] YOURDON, EDWARD und LARRY L. CONSTANTINE: *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st Auflage, 1979.

[Yos13] YOSHIDA, SHIGEO: *ARDroneForP5*. `http://kougaku-navi.net/ARDroneForP5/index_en.html`, February 2013. aufgerufen am 27.02.2013.

[YSMMW11] YASMINA SANTOS, MARIBEL, JOSÉ MENDES, ADRIANO MOREIRA und MONICA WACHOWICZ: *Towards a Spatio-Temporal Informa-*

tion System for Moving Objects. In: MURGANTE, BENIAMINO, OSVALDO GERVASI, ANDRÉS IGLESIAS, DAVID TANIAR und BERNADY APDUHAN (Herausgeber): *Computational Science and Its Applications - ICCSA 2011*, Band 6782 der Reihe *Lecture Notes in Computer Science*, Seiten 1–16. Springer Berlin / Heidelberg, 2011.

[ZBH+09] ZEEB, E., R. BEHNKE, C. HESS, D. TIMMERMANN, F. GOLATOWSKI und K. THUROW: *Generic sensor network gateway architecture for plug and play data management in smart laboratory environments*. In: *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, Seiten 1–8, sept. 2009.

[Zig12] ZIGBEE ALLIANCE: *Homepage*. `http://www.zigbee.org`, June 2012. aufgerufen am 04.06.2012.

[ZLO07] ZIMMERMANN, ANDREAS, ANDREAS LORENZ und REINHARD OPPERMANN: *An operational definition of context*. In: *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*, CONTEXT'07, Seiten 558–571, Berlin, Heidelberg, 2007. Springer-Verlag.

[ZPT+11] ZAHARIADIS, THEODORE, DIMITRI PAPADIMITRIOU, HANNES TSCHOFENIG, STEPHAN HALLER, PETROS DARAS, GEORGE D. STAMOULIS und MANFRED HAUSWIRTH: *The Future Internet*. In: DOMINGUE, JOHN, ALEX GALIS, ANASTASIUS GAVRAS, THEODORE ZAHARIADIS und DAVE LAMBERT (Herausgeber): *Future Internet Assembly*, Band 7281 der Reihe *Lecture Notes in Computer Science*, Kapitel Towards a future internet architecture, Seiten 7–18. Springer-Verlag, Berlin, Heidelberg, 2011.

[ZSM09] ZYL, T. L. VAN, I. SIMONIS und G. MCFERREN: *The Sensor Web: systems of sensor systems*. International Journal of Digital Earth, 2(1):16–30, 2009.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 23. Oktober 2013

Dirk Bade