Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2 - SoSe 2010

Sören Voskuhl

Architekturen für Context-Aware Systeme

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einführung

Ein Entwicklungstrend der Informatik zeichnet sich durch immer kleinere und leistungsfähigere Computer aus. Aufgrund dieser Eigenschaften erhalten sie stetig wachsenden Einzug in das alltägliche Leben des Menschen, sodass zunehmend mehr Rechenleistung in die Wohnbereiche des Menschen integriert wird. Dabei soll den Bewohnern eine unaufdringliche und zugleich ansprechende Umgebung geschaffen werden, indem Informationstechnologien in alltägliche Gebrauchsgegenstände eingebunden werden, mit dem Ziel die Personen bei verschiedenen Aufgaben zu unterstützen. Zu diesem Zweck interagieren die Technologien mit den Benutzern auf möglichst natürlichem Wege, bis hin zu einer impliziten Interaktion (Meyer und Rakotonirainy, 2003). Wohnbereiche, die beschriebene Eigenschaften aufweisen, werden als „Smart Homes“ bezeichnet.
Damit der Mensch möglichst exakt hinsichtlich seiner Präferenzen unterstützt werden kann, ist es erforderlich, auf verschiedene Personen in unterschiedlichen Situationen reagieren zu können. Hierzu ist es notwendig, verschiedene Daten über die Umgebung zu sammeln, damit der aktuelle Kontext in das Verhalten der Computer einbezogen werden kann. Systeme dieser Art sind in der Lage, ihre Operationen dem Kontext anzupassen, ohne dass eine explizite Eingabe des Benutzers benötigt wird und steigern somit die Benutzerfreundlichkeit und Effizienz (Baldauf und Dustdar, 2004).
In diesen Zusammenhang gilt es zu betrachten, welche Daten zur Kontexterfassung erhoben werden müssen und somit zum Kontext gehören. Hierzu muss im ersten Schritt definiert werden, was unter dem Begriff „Kontext“ zu verstehen ist. Nach Dey und Abowd (1999) wird Kontext als jede Information, die dazu verwendet werden kann, die Situation einer Instanz zu charakterisieren, beschrieben. Bei dieser Instanz kann es sich um eine Person, einen Ort oder ein zur Interaktion zwischen Benutzer und Computer relevantes Objekt, einschließlich des Benutzers und der Anwendung selbst, handeln.
Bei der Realisierung von Context-Aware Systemen muss eine Kaskade von Prozessen durchlaufen werden, in der die einzelnen Sensorwerte verarbeitet und interpretiert werden, sodass ein Gesamtkontext entsteht. Zur Bereitstellung dieser Daten und Kontexte ist eine Middleware empfehlenswert, die als Kommunikationsbasis der teilnehmenden Applikationen dient und alle möglichen Kontexte kennt. Eine mögliche Blackboard-Architektur sowie deren Vorteile werden in Voskuhl (2010) diskutiert. Aufbauend auf dieser Arbeit sollen in dieser Ausarbeitung ähnliche Architekturen vorgestellt und bewertet werden, mit dem Ziel weitere Ansätze bei der Auswahl einer Middleware für intelligente Wohnbereiche, wie das im Fol-

genden vorgestellte „Living Place Hamburg“, zu erlangen.

Das „Living Place Hamburg“ ist ein Projekt der Hochschule für Angewandte Wissenschaften Hamburg, in dem seit ca. drei Jahren der Bereich des „Smart Home“ untersucht und stetig weiterentwickelt wird. Derzeit wird hier auf ca. 140m^2 eine bewohnbare Wohnung aufgebaut, damit die in verschiedenen Projekten entwickelten Systeme unter realen Bedingungen getestet und angewendet werden können. Ein Grundriss dieser Wohnung wird auf der Abbildung 1.1 dargestellt.

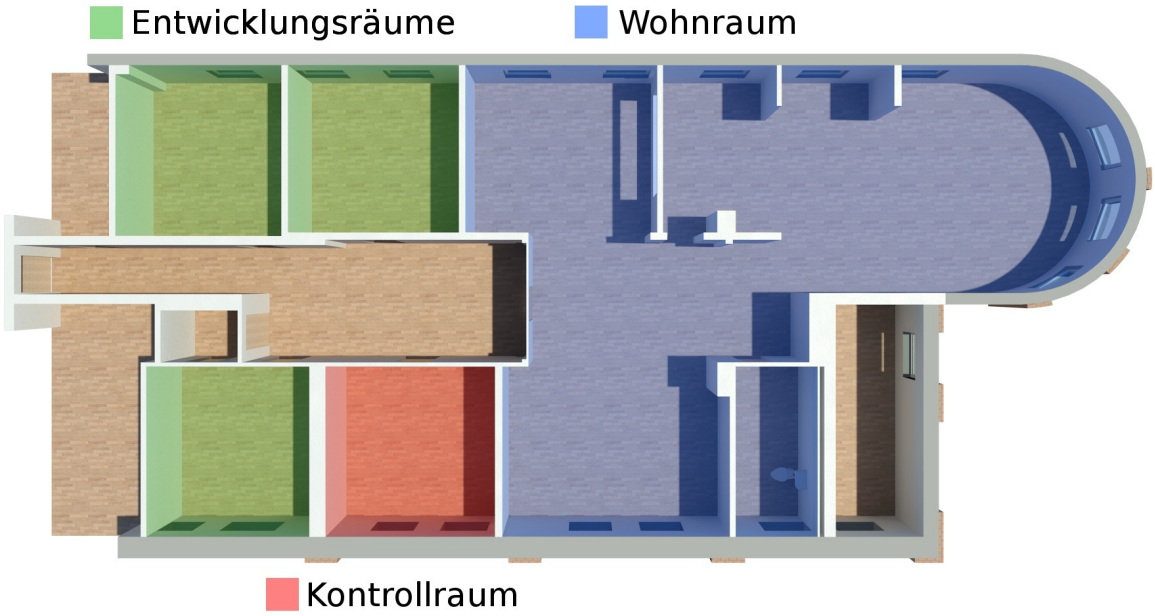

Abbildung 1.1: Grundriss des „Living Place Hamburg“

2 Vergleichbare Arbeiten

Um ein leichteres Verständnis für die im Folgenden vorgestellten Arbeiten zu entwickeln, werden folgend einige Begriffe vorgestellt.

Kontext

Wenn Menschen untereinander kommunizieren, wird eine Vielzahl von Informationen ohne explizite Kommunikation übermittelt. Gesten, Gesichtsausdrücke, Beziehungen zu anderen Menschen und Objekte der Umgebung werden zum Verständnis der expliziten Kommunikation verwendet (Dey u. a., 1999). Leider ist dies in der traditionellen Interaktion zwischen Mensch und Computer nicht möglich, wodurch die Eingabemöglichkeiten des Benutzers zur Kommunikation reduziert werden, was zur Folge hat, dass die Applikationen keine Möglichkeit besitzen, auf Veränderungen des Bezugsrahmens zu reagieren.
Wenn Computer den Kontext in ihr Verhalten einbeziehen, wird somit das Spektrum an Kommunikationsmöglichkeiten erhöht und Anwendungsprogrammen die Möglichkeit offeriert, ihre Reaktionen situationsabhängig durchzuführen. Mit der Realisierung dieser Fähigkeit befassen sich Context-Aware Systeme.

Context Awareness

Context-Aware Systeme beziehen den aktuellen Kontext ein, um anhand dieser Informationen ihr Verhalten zu bestimmen. Zur Entscheidung, welche Informationen zur Erfassung des Kontextes von Bedeutung sind, ist eine Definition des Begriffs „Kontext“ bezüglich der Kommunikation zwischen Computer und Mensch notwendig. Neben der bereits in 1 erwähnten Definition des Begriffs, unterteilen Dey und Abowd (1999) Kontext in vier verschiedene Kategorien, nach denen sich Kontexte bestimmen lassen. Es handelt sich hierbei um den Ort, die Identität, die Aktivität und die Zeit.
Die benötigten Informationen zu Kontexterfassung werden dabei von verschiedenen Sensoren gesammelt. An dieser Stelle ergibt sich eine Kaskade an Prozessen, da aus den einzelnen Sensorwerten ein Gesamtkontext geformt werden muss. Hierzu muss eine Architektur bereitgestellt werden, welche diese Daten entgegennimmt, interpretiert und so aufbereitet, dass diese Informationen als Bezugsrahmen für die unterschiedlichen Anwendungen bereitstehen. Es muss also eine Sensorfusion vorgenommen werden, die aus

mehreren Teilprozessen besteht.
Die folgenden Arbeiten beschreiben eine mögliche Architektur für Context-Aware Systeme.

2.1 SOCAM

Die „Service-Oriented Context-Aware Middleware“ Architektur (SOCAM) (Gu u. a., 2005) unterstützt die Erfassung und Bekanntgabe verschiedener Kontexte von unterschiedlichen Context-Providern, indem die vorliegenden Kontextinformationen interpretiert und Schlüsse daraus gezogen werden. Die von der Architektur angebotenen Dienste sollen die Gestaltung der kontextabhängigen Applikationen erleichtern, welche auf verschiedene Services zugreifen, mit dem Ziel Kontexte unterschiedlicher Komplexität zu erlangen.

Kontextmodellierung

Die Kontexte werden hier mit Hilfe der Prädikatenlogik erster Ordnung repräsentiert. Das Basismodell besitzt dabei die Form Prädikat(Subjekt, Wert). Beispielsweise steht *Location(John, bathroom)* dafür, dass John sich im Badzimmer befindet, *Temperature(kitchen, 120)* bedeutet, dass in der Küche 120 °F herrschen.
Dieses Basismodell kann erweitert werden, sodass komplexe Kontexte oder eine Reihe von Zusammenhängen dargestellt werden können. Hierzu werden Prädikate kombiniert und die boolesche Algebra eingesetzt. Als Beispiel können so mit einem Ausdruck *FoodPreference(familyMembers, foodItems)* alle Esspräferenzen einer Familie dargestellt werden: $FoodPreference(John, FoodList\_1) \vee FoodPreference(Alice, FoodList\_2) \vee FoodPreference(Tom, FoodList\_3)$.
Die Strukturen und Eigenschaften der Kontext-Prädikate werden in einer Ontologie beschrieben. Dabei wird eine Ontologie als eine explizite Spezifikation einer Konzeptualisierung beschrieben (Gruber, 1995). Unter einer Konzeption wird in diesem Zusammenhang eine abstrakte, vereinfachte Sicht auf die Domäne, die für einen bestimmten Zweck repräsentiert werden soll, verstanden.
In dieser Architektur wird davon ausgegangen, dass die Verarbeitung und Wartung des Kontextes nicht auf den ubiquitären Geräten durchgeführt wird, da diese häufig in ihrer Rechen- sowie Speicherkapazität eingeschränkt sind. Aus diesem Grund werden die Kontext-Ontologien in zwei Schichten aufgeteilt. In diesem Ansatz existiert eine allgemeine „Oberontologie“ und eine „domänenspezifische Ontologie“. Die verallgemeinerte Ontologie erfasst generelle Kontexte für alle Domänen. Sie wird einmal definiert und von allen Domänen gemeinsam verwendet. Die domänenspezifische Ontologie enthält dagegen eine Sammlung von Low-Level-Ontologien. Hier werden die Details der generellen Konzepte

sowie deren Eigenschaften für jede Domäne definiert. Die Trennung in diese zwei Bereiche ermöglicht eine signifikante Reduktion des benötigten Wissens auf den einzelnen Geräten.

Architektur

Die SOCAM Architektur besteht aus verschiedenen, unabhängigen Komponenten (2.1).

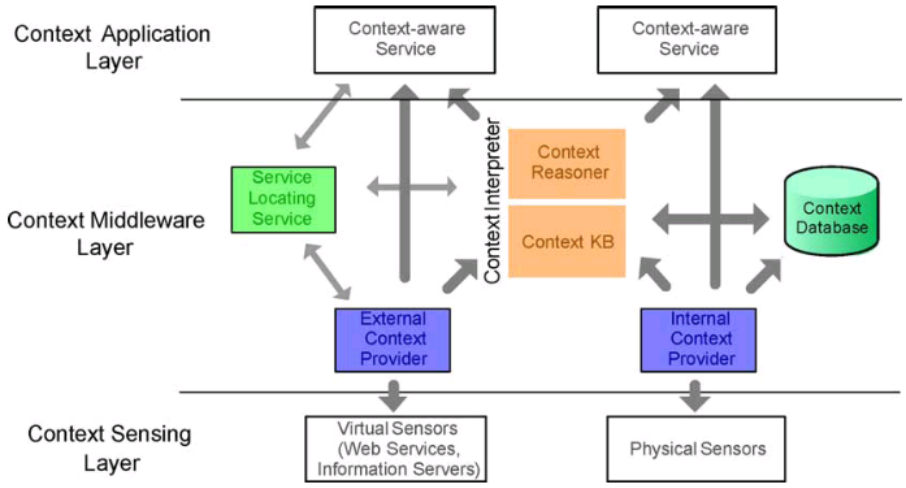

Abbildung 2.1: SOCAM Architektur aus Gu u. a. (2005)

- **Context Provider:** Sie abstrahieren Kontexte aus heterogenen Quellen und konvertieren diese in die benötigte Form der Ontologie, sodass die Kontextinformationen an weitere Service-Komponenten verteilt und wiederverwendet werden können.
- **Context Interpreter:** Diese Komponente berechnet den Kontext, indem aus den vorliegenden Kontextinformationen Schlüsse gezogen werden. Der Context Interpreter enthält einen „Context Reasoner“ und eine „Context Knowledge Base (KB)“. Ersterer stellt die abgeleiteten Kontexte zur Verfügung und bemerkt Inkonsistenten und Konflikte in der Context KB. Die Knowledge Base bietet eine Reihe von API's für weitere Service-Komponenten an, mit dem Ziel ihnen das Abfragen, Hinzufügen und Modifizieren von Kontextinformationen zu ermöglichen. Sie enthält hierzu die domänenspezifische Ontologie sowie die zugehörigen Instanzen.

- **Context Database:** Sie speichert Kontext-Ontologien und aufgetretene Kontexte einer domänenspezifischen Ontologie.
- **Service locating service:** Diese Komponente bietet einen Mechanismus an, mit dem Context Provider und Context Interpreter ihre Präsenz bekannt geben können. Außerdem ermöglicht sie es Benutzern und Applikationen Services zu finden.

Ähnliche Architekturen werden in „A Middleware for Context-Aware Agents in Ubiquitous Computing Environments“ (Ranganathan und Campbell, 2003) und in „iFlat - Eine dienstorientierte Architektur für intelligente Räume“ (Stegelmeier u. a., 2009) vorgestellt.

Bewertung

In dieser Architektur lassen sich leicht neue Kontexte hinzufügen, indem sie in die Ontologie der Context Database aufgenommen werden. Durch das stetige Wachstum des „Living Place Hamburg“ bietet sich eine solche Architektur an, da hier die Integration von neuen Arbeiten ohne Weiteres ermöglicht wird. Jedoch werden hierfür Applikationen, die in der Lage sind, mit den bisherigen Services zu kommunizieren, vorausgesetzt.
In dieser Middleware ergibt sich ein „Single Point of Failure“, sodass bei einem Ausfall der Context Database der Betrieb der involvierten Geräte weiterhin gewährleistet werden muss. Des Weiteren ist es in einer Architektur dieser Form eine essentielle Fragestellung, wie sich das System beim Zugriff vieler Sensoren und Anwendungen verhält. In diesem Fall muss sichergestellt werden, dass es zu keinen Einschränkungen in der Performance kommt.

2.2 UbiHome

Die UbiHome Infrastruktur betrachtet einen weiteren Ansatz, wie Sensordaten in einem „ubiquitous home network“ zur Verfügung gestellt werden können (Ha u. a., 2007). Während SOCAM (2.1) von einer Bereitstellung von statisch vorprogrammierten Services für eine spezielle Umgebung ausgeht, werden bei UbiHome dynamische, Ad hoc und heterogene Service-Umgebungen betrachtet.
Zur automatischen Integration von Endgeräten, Sensoren und anderen Ubiquitous Computing Ressourcen werden semantische Web Services implementiert. Die Beschreibung von Funktionen und Schnittstellen der Web Services werden in OWL-S (Web Ontology Language for Services) definiert. Diese Daten werden in der „Knowledge Registry“, welches die Wissensbasis für alle beteiligten Service-Agents innerhalb des Wohnbereichs darstellt, gespeichert. Diese Service-Agents offerieren ihre Services durch eine automatische Interaktion mit den Endgeräten und Sensoren via SOAP (Simple Object Access Protocol), einem

Web Service Ausführungsprotokoll.

Architektur

An dieser Stelle werden die in dieser Architektur verwendeten Komponenten genauer betrachtet. Die Beziehungen dieser untereinander werden in der Abbildung 2.2 dargestellt.

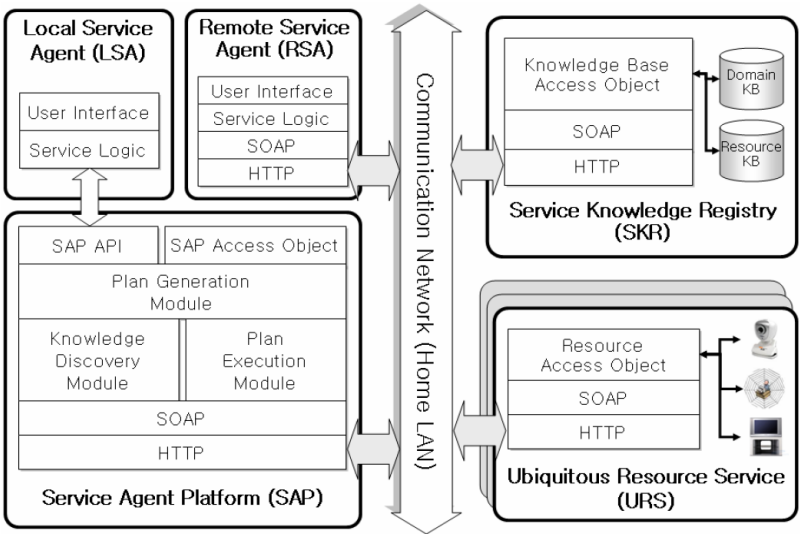

Abbildung 2.2: UbiHome Architektur aus Ha u. a. (2007)

- **Service Agent Platform (SAP):** Das „Plan Generation Module“ der SAP empfängt Service-Anfragen in Form von OWL-S Request-Profilen entweder über die SAP API oder über das SAP Zugriffsobjekt. Daraufhin werden die benötigten Informationen bei dem „Knowledge Discovery Module“ erfragt, mit dem Ziel einen Plan für den angeforderten Service zu erstellen oder aktuelle Kontextinformationen, wie z.B. den Standort einer Person, zu erlangen. Das befragte Modul untersucht hierzu die „Service Knowledge Registry“ (SKR), um die erforderlichen Informationen und Dienste zu erhalten. Nachdem diese Informationen zur Verfügung stehen, transformiert das „Plan Execution Module“ den Service-Plan in ein ausführbares Format und lässt ihn von den benötigten Geräten, welche es über seinen Web Service Protocol Stack erreicht, durchführen.
- **Service Agent (SA):** In dieser Architektur existieren zwei Typen von Service Agents. Zum einen den „Local SA“ (LSA) und zum anderen den „Remote SA“ (RSA). Ers-

terer ist direkt auf der SAP API und folglich auf dem gleichen Gerät wie die SAP implementiert. Hierbei könnte es sich beispielweise um einen Home-Server mit entsprechender Leistung handeln. Dagegen werden RSA's auf einem zusätzlichen Gerät implementiert. Dabei kommunizieren sie mit der Service Agent Platform über das SAP Zugriffsobjekt. RSA's werden auf Geräte mit geringer Rechenleistung, wie einem PDA implementiert.
Beide SA's besitzen sowohl eine Benutzeroberfläche als auch eine Service-Logik. Letztere nimmt die Befehle der Benutzeroberfläche entgegen und kodiert diese in ein OWL-S Request-Profil, um dieses dann zur SAP zu übertragen.

- **Ubiquitous Ressource Service (URS):** Bei einer URS handelt es sich um eine konkrete Implementation eines Web-Services für ein ubiquitäres Gerät wie einer Lichtanlage oder einer Klimaanlage. Dabei besitzt jeder Service ein Zugriffsobjekt für korrespondierende Geräte, welches entweder allein oder als Gruppe mehrerer zusammenwirkender auftreten können. Außerdem verfügt jeder URS über einen Web-Service Protokoll Stack, über den er mit dem Service Agent kommuniziert.

- **Service Knowledge Registry (SKR):** Die SKR speichert das Wissen über die einzelnen Services innerhalb des Wohnbereichs. Dabei wird zwischen Domänen- und Gerätewissen unterschieden. Ersteres enthält Prozessmodelle, die Input, Output, Vorbedingungen und Effekte der Services bestimmen. Das Gerätewissen umfasst dagegen die atomaren Beschreibungen der Dienste, die von einem ubiquitären Gerät ausgeführt werden können.

Von den einzelnen Sensordaten bis hin zu einer Reaktion eines Gerätes auf den Kontext werden drei Schritte durchlaufen. Zunächst wird die „Knowledge Discovery Phase“ ausgeführt, in der die erforderlichen Daten beim SKR über SOAP-Nachrichten erfragt werden. In der darauf folgenden „Plan Generation Phase“ wird der Ausführungsplan durch Hierarchical Task Network (HTN) generiert. Das HTN Verfahren erstellt hierzu ein „Task Network“, welches eine bestimmte Menge von Aufgaben und deren Vorbedingungen definiert (Erol u. a., 1994). Dabei wird zwischen Aufgaben, die direkt ausgeführt werden können („Primitive task“) und solchen, die sich aus einer Sequenz von primitiven Aufgaben zusammensetzen („Compound task“), unterschieden.
Im letzten Schritt, der „Plan Execution Phase“, werden die „primitive Tasks“ durch Operationen der Web-Services in ein ausführbares Format transformiert und ihre Ausführung initiiert.

Eine ähnliche Architektur, in der ebenfalls Web-Services zur Kommunikation unter den Teilnehmern verwendet werden, wird in „Towards Context-Aware Adaptable Web Services“ (Keidl und Kemper, 2004) vorgestellt. In „Using Semantic Web Services for Context-Aware Mobile Applications“ (Sheshagiri u. a., 2004) wird wie in der vorgestellten Architektur ein Ausführungsplan erstellt, der von den beteiligten Geräten im entsprechenden Kontext aus-

geführt wird. Die Repräsentation der Kontextinformationen wird hier ebenso in OWL-S definiert.

Bewertung

Die UbiHome Architektur erlaubt eine Kommunikation zwischen verschiedenartigen Sensoren und Applikationen. Durch den Einsatz von OWL-S wird eine standardisierte Beschreibung von Funktionen und Schnittstellen der Web Services eingeführt. Dieser Ansatz erfordert allerdings eine hohe Sorgfalt bei der Implementation der einzelnen Anwendungen, da jede Applikation ihren eigenen Web Service benötigt.
Im „Living Place Hamburg“ kommunizieren viele heterogene Systeme miteinander, die sich in ihrer Rechenleistung stark unterscheiden. Diese Problematik wird hier durch die Einführung von LSA's und RSA's gelöst.
Durch den Einsatz der „Service Agent Platform“ entsteht auch hier, wie bereits im ersten System, ein „Single Point of Failure“, sodass dieses System bei einem Ausfall dieser Komponente nicht mehr lauffähig ist.

2.3 Dezentrale Context-Aware Middleware

In (Dey u. a., 1999) wird eine weitere Architektur für Context-Aware Systeme vorgestellt. Diese besteht aus drei Komponenten, die im folgenden Abschnitt genauer betrachtet werden. Abbildung 2.3 präsentiert das Zusammenspiel sowie die Abhängigkeiten der einzelnen Komponenten.

- **Context Widget:** Ein Widget definiert sich zum einen durch seine Attribute. Diese sind Teil eines Kontextes und machen das Widget für weitere Geräte erreichbar. Auf der anderen Seite besitzt es „Callbacks“, welche die Typen der Ereignisse repräsentieren, die vom Widget verwendet werden können, um andere Geräte anzusprechen. Hierdurch ist jedes Context Widget neben dem Abfragen von Informationen darüber hinaus in der Lage, Interessenten über aktuelle Kontextdaten zu informieren.
  Aufgrund der Unabhängigkeit unter den einzelnen Widgets, besteht die Möglichkeit, jederzeit weitere Geräte in ein bestehendes System zu integrieren. Hierzu müssen lediglich die Attribute und Callbacks des Widgets definiert werden sowie das Format zur Kommunikation mit weiteren Teilnehmern bekannt sein muss.
- **Context Server:** Diese Komponente wird dazu verwendet, alle Kontextinformationen über eine Entität abzulegen. Daher müssen die Context Widgets ihr Interesse an Kontextinformationen nicht jedem einzelnen Widget mitteilen, sondern können über ein Objekt mit weiteren Teilnehmern kommunizieren. Der Context Server informiert nun

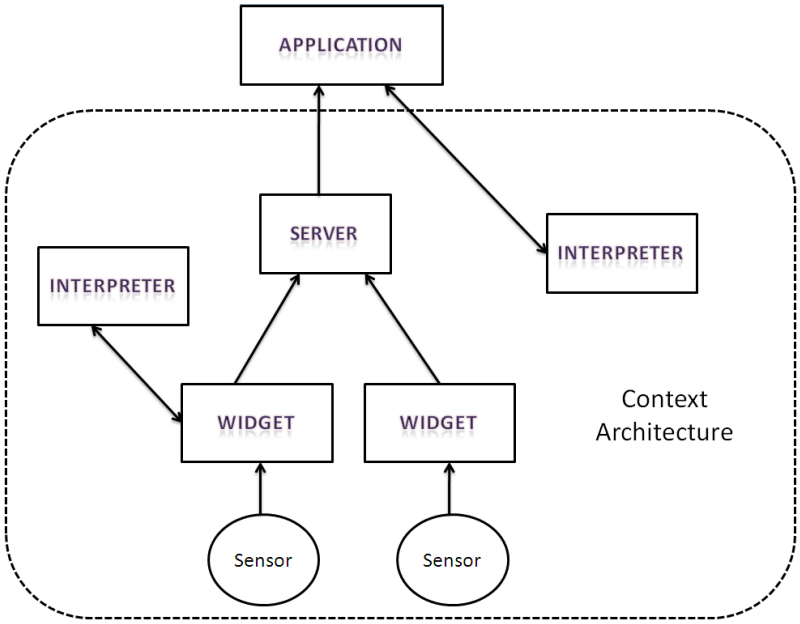

Abbildung 2.3: Architektur nachempfunden aus Dey u. a. (1999)

die Interessenten bei entsprechenden Änderungen der vorliegenden Daten und fungiert somit als ein Proxy zur Applikation.
Für den Fall, dass ein neuer Server in die Architektur eingebunden werden soll, muss dieser ausschließlich die Widgets bekanntgeben, die über ihn zu erreichen sind.

- **Context Interpreter:** Ein Context Interpreter ist, wie bereits in 2.1 vorgestellt, für die Interpretation von Kontexten verantwortlich. Allerdings wird in dieser Architektur davon ausgegangen, dass für jede Applikation ein Context Interpreter existiert. Die Interpretation der Kontextinformationen wird den Anwendungen hier entzogen, mit dem Ziel einen Context Interpreter für mehrere Applikationen zu verwenden. Um diese Wiederverwendbarkeit herzustellen, bleibt die Verarbeitung von individuellen Interpretationen i.d.R. bei den Anwendungen.

Ein essentieller Unterschied zur in 2.1 vorgestellten Architektur liegt darin, dass in dieser Struktur bis auf den Kommunikationsserver keine zentralen Komponenten existieren, wie beispielsweise eine „Context Database“, die alle möglichen Kontexte kennt. Aufgrund dieser Eigenschaft besitzt jede Applikation somit einen eigenen „Context Interpreter“. Diese Struktur setzt sich somit aus mehreren, voneinander unabhängigen Inseln zusammen.

Eine ähnliche Handhabung von Kontexten wird in „A Collaborative Wearable System with Remote Sensing“ (Bauer u. a., 1998) und in „A system architecture for context-aware mobile

computing“ (Schilit, 1995) vorgestellt.

Bewertung

Die hier vorgestellte Architektur überlässt die individuelle Interpretation von Daten den Anwendungen selbst. Hierdurch wird der „Single Point of Failure“ zu einem großen Teil abgebaut. Zwar sind die Applikationen einerseits vom Context Server abhängig, wenn sie die Daten anderer Teilnehmer in die Kontexterstellung einbeziehen wollen, andererseits sind sie aber im Fall eines Ausfalls dieser Komponente in der Lage, auf die eigenständig gesammelten Daten zu reagieren, da zu jeder Anwendung ein eigener „Context Interpreter“ vorliegt und keine zentrale Einheit benötigt wird.
Der Nachteil dieser Architektur liegt darin, dass der Entwickler einer neuen Anwendung mit den Details der Sensoren vertraut sein muss, damit er auf die entsprechenden Daten zugreifen kann. Zudem muss er die möglicherweise komplexen Zusammenhänge des Kontextes selbst implementieren.

3 Fazit

3.1 Zusammenfassung

Die hier vorgestellten Arbeiten zeigen, dass es für die Realisierung einer Middleware für Context-Aware Systeme verschiedene Ansätze gibt. Während bei SOCAM (2.1) von statisch vorprogrammierten Services ausgegangen wird, beschäftigt sich die UbiHome Architektur (2.2) mit dynamischen, Ad hoc und heterogenen Service-Umgebungen. In der dezentralen Architektur (2.3) wird versucht den „Single Point of Failure“ aufzulösen, indem den Applikationen die individuelle Interpretation der Daten selbst überlassen wird, was allerdings die Implementation neuer Applikationen für den Entwickler erschwert.
Die Vor- und Nachteile dieser Ansätze zeigen, dass es bisher keine einheitliche Lösung für die Architektur von Context-Aware Systemen gibt, sodass es empfehlenswert ist, weitere Projekte in diesem Bereich zu untersuchen und darauf aufbauend eigene Prototypen zu realisieren, mit dem Ziel anhand dieser bisher erlangte Kenntnisse zu erweitern.

3.2 Ausblick

Durch das „Living Place Hamburg“ bietet sich die Möglichkeit, an einem innovativen Themenbereich mitzuwirken, der eine Vielzahl von Teilbereichen aufweist, in denen derzeit an der HAW Hamburg verschiedene Projekte stattfinden.
Durch meine bisher gesammelten Erfahrungen über Context-Aware Systeme sowie die dazugehörige Architektur, möchte ich mich in den weiteren Semestern weiter vertiefend in dieses Thema einarbeiten und bei der Realisierung der real bewohnbaren Wohnung teilnehmen. Hierbei möchte ich meinen Fokus auf die zu entwickelnde Middleware setzen, indem ich mein bisheriges Wissen einbringe und es durch die Zusammenarbeit mit verschiedenen Personen, die in diesem Kontext tätig sind, erweitern. Dabei ist das Ziel, eine Middleware zu entwickeln, die für den dauerhaften Einsatz in einem „Smart Home“ geeignet ist, indem sie eine Kommunikationsbasis für verschiedene Applikationen darstellt und Kontexte sowie Kontextinformationen unter ihnen kommuniziert.
Darüber hinaus ist mein Ziel, bei der Realisierung von Szenarien, wie z.B. in Voskuhl (2010) beschrieben, mitzuwirken. Hierzu müssen verschiedene Sensortechnologien untersucht und

in Prototypen verwendet werden, damit verlässliche Aussagen über die Verwendbarkeit getroffen und die Bewohner eines „Smart Homes“ mit neuartigen Technologien bei ihren alltäglichen Aufgaben unterstützt werden können.

Literaturverzeichnis

[Baldauf und Dustdar 2004] BALDAUF, Matthias ; DUSTDAR, Schahram: A Survey on Context-aware systems. In: *INTERNATIONAL JOURNAL OF AD HOC AND UBIQUITOUS COMPUTING* (2004), S. 263 – 276

[Bauer u. a. 1998] BAUER, Martin ; HEIBER, Timo ; KORTUEM, Gerd ; SEGALL, Zary: A Collaborative Wearable System with Remote Sensing. In: *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1998, S. 10. – ISBN 0-8186-9074-7

[Dey u. a. 1999] DEY, A.K. ; SALBER, D. ; FUTAKAWA, M. ; ABOWD, G.D.: An Architecture to Support Context-Aware Applications, Georgia Institute of Technology, 1999

[Dey und Abowd 1999] DEY, Anind K. ; ABOWD, Gregory D.: *Towards a Better Understanding of Context and Context-Awareness*. 1999. – URL http://www.it.usyd.edu.au/~bob/IE/99-22.pdf. – Zugriffsdatum: 27.02.2009

[Erol u. a. 1994] EROL, Kutluhan ; HENDLER, James ; NAU, Dana S.: HTN Planning: Complexity and Expressivity. In: *In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94*, AAAI Press, 1994, S. 1123–1128

[Gruber 1995] GRUBER, Thomas R.: Toward principles for the design of ontologies used for knowledge sharing. In: *Int. J. Hum.-Comput. Stud.* 43 (1995), Nr. 5-6, S. 907–928. – ISSN 1071-5819

[Gu u. a. 2005] GU, Tao ; PUNG, Hung K. ; ZHANG, Da Q.: A service-oriented middleware for building context-aware services. In: *J. Netw. Comput. Appl.* 28 (2005), Nr. 1, S. 1–18. – ISSN 1084-8045

[Ha u. a. 2007] HA, Young-Guk ; SOHN, Joo-Chan ; CHO, Young-Jo: ubiHome: An Infrastructure for Ubiquitous Home Network Services, 2007, S. 1–6

[Keidl und Kemper 2004] KEIDL, Markus ; KEMPER, Alfons: Towards context-aware adaptable web services. In: *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA : ACM, 2004, S. 55–65. – ISBN 1-58113-912-8

[Meyer und Rakotonirainy 2003] MEYER, Sven ; RAKOTONIRAINY, Andry: A survey of research on context-aware homes. In: *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2003, S. 159–168. – ISBN 1-920682-00-7

[Ranganathan und Campbell 2003] RANGANATHAN, Anand ; CAMPBELL, Roy H.: A middleware for context-aware agents in ubiquitous computing environments. In: *Middleware '03: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. New York, NY, USA : Springer-Verlag New York, Inc., 2003, S. 143–161. – ISBN 3-540-40317-5

[Schilit 1995] SCHILIT, William N.: *A system architecture for context-aware mobile computing*. New York, NY, USA, Dissertation, 1995

[Sheshagiri u. a. 2004] SHESHAGIRI, Mithun ; SADEH, Norman M. ; G, Fabien: Using Semantic Web Services for Context-Aware Mobile. In: *Applications, MobiSys 2004 Workshop on Context Awareness*, 2004

[Stegelmeier u. a. 2009] STEGELMEIER, Sven ; WENDT, Piotr ; LUCK, Kai von: iFlat - Eine dienstorientierte Architektur für intelligente Räume. (2009), S. 1 – 5. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/aal2009.pdf

[Voskuhl 2010] VOSKUHL, Sören: Bereitstellung einer Sensorwolke. (2010). – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/Voskuhl/bericht.pdf