Kontextverzeichnisse für die Entwicklung mobiler kontextbasierter Anwendungen

Ralph Löwe, Peter Mandl
Competence Center Wirtschaftsinformatik
Hochschule München
rloewe@hm.edu, mandl@cs.hm.edu

Schlüsselworte:
Kontextbasierte Dienste, Kontextverzeichnis, Ubiquitous Computing, Hierachische Datenstrukturen, Verzeichnisdienste, Lightweight Directory Access Protocol (LDAP), Java Naming and Directory Interface (JNDI)

Einleitung

Aktuellen Entwicklungen zufolge steigt die Anzahl an Computern pro Person ständig. Weiser (1991) erkannte diesen Trend schon früh und führte den Begriff des allgegenwärtigen Rechnens ein (engl. Ubiquitous Computing). Beziehen Systemlösungen die aktuelle Umgebung - den sog. Kontext – mit ein, so spricht man von kontextbasierten Anwendungssystemen. Der Kontext wird in diesen Systemen über verschiedenste Sensoren erfasst und in Anwendungen verwendet.

Eine frühe Form von kontextbasierten Anwendungen stellen ortsbasierte Dienste dar. Die Anwendungslogik berücksichtigt dabei den Kontext *Ort*, welcher - vermengt mit anderen Informationen – einen Mehrwert schafft. Bei ortsbasierten Diensten spielen Geoinformationssysteme (GIS) eine zentrale Rolle. Mit diesen werden die geographischen Daten gespeichert und es wird ein Auflösen und Rechnen mit Ortsinformationen ermöglicht.

Laut einer Studie von Pettey & Stevens (2010) gelten kontextbasierte Systeme zukünftig für Unternehmen als ein wichtiger Wettbewerbsfaktor. Spätestens 2015 soll der Kontext im Bereich der mobilen Dienste so einflussreich sein wie die Suchmaschine für das Internet. Allerdings stellt die Einbindung von Kontextinformationen auch neue Anforderungen an die bestehende Anwendungslandschaft.

Analog zu Geoinformationssystemen, die ausschließlich den Ortskontext betrachten, kann man etwas verallgemeinert ein System zur Verwaltung von allen möglichen Kontextinformationen beschreiben, das auch als *Kontextverzeichnis* bezeichnet wird (siehe Löwe, R. & Mandl, P. 2010). Kontextverzeichnisse sind in erster Linie für mobile Anwendungen von Bedeutung, denn diese müssen mit Kontextveränderungen umgehen können.

In diesem Beitrag soll nach einer Einführung in grundlegende Aspekte kontextbasierter Anwendungsentwicklung die Idee eines Kontextverzeichnisses als grundlegender Mechanismus zur Verwaltung von Kontextinformationen und als Bestandteil einer kontextbasierten Middleware vorgestellt werden. Die gewonnenen Erkenntnisse stammen aus einem aktuellen Forschungsvorhaben des Competence Center Wirtschaftsinformatik der Hochschule München, das sich mit kontextbasierten und mobilen Informationssystemen auseinandersetzt.

Grundideen kontextbasierter Anwendungsentwicklung

Die Vision von Weiser (1991) zum allgegenwärtigen Rechnen skizziert eine neuartige Soft- und Hardwarelandschaft, welche den eigentlichen Nutzen der Geräte und Anwendungen näher an den Benutzer heranbringen soll. Ein besonderer Aspekt ist dabei die Unsichtbarkeit des Computers. So wie eine Brille in den Hintergrund tritt und ein scharfes Sehen selbstverständlich macht, soll auch ein Computer da sein und unsichtbar den eigentlichen Zweck erfüllen. Dies fordert auch Norman (1998) und beschreibt eine stark benutzerorientierte und wirtschaftliche Sicht. Für ihn steht das Problem im Vordergrund, dass ein Computer viel zu kompliziert für den normalen Benutzer ist. Er fordert kleine spezialisierte Geräte (engl. Information Appliances), welche auf einen Zweck hin optimiert sind.

Oft versuchen Anwender den Computer wie einen Menschen zu behandeln. Mit ihm wird geredet, ihm wird gesagt, wenn er falsch liegt und teilweise wird er sogar beschimpft. Bei einer menschlichen Kommunikation kann dies eine normale Reaktion sein. Ein Mensch kann sachliche und emotionale Informationen aus dem Kontext herausfiltern und aus Fehlern lernen. Computer können dies derzeit noch nicht.

Was fehlt also der heutigen Mensch-Maschine-Interaktion (vgl. hierzu Hewett (1992))? Schulz von Thun (1981) beschreibt vier Ebenen der menschlichen Kommunikation. So hat jede Nachricht zwischen Menschen eine Sachebene, eine Selbstkundgabe, eine Beziehungsseite und eine Appellseite. Verallgemeinert ausgedrückt ändert sich im Laufe der Verarbeitung einer Nachricht also ihr Bezug. In der Informatik bezeichnen einige Forscher diesen Bezug als Kontext der Interaktion. Hauptsächlich von Schilit & Theimer (1994) und Dey & Abowd (2000) wurde für die Entwicklung von Anwendungssystemen, die Kontextinformationen berücksichtigen, der Begriff der *kontextbasierten Anwendungsentwicklung* (engl. Context-Aware Computing) eingeführt. Im Rahmen der Verarbeitung von Eingabedaten wird hier der Kontext näher untersucht, mit dem Ziel die Kommunikation zwischen Benutzer und Anwendung zu verbessern und eine entsprechende Ausgabe zu erzeugen (siehe Abbildung 1).

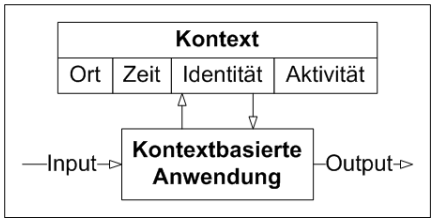

Abbildung 1: Kontext, in Anlehnung an Lieberman & Selker (2000) sowie Dey & Abowd (2000).

Jedoch kann der Kontext sehr vielseitig, komplex sowie je nach Anwendung unterschiedlich beschaffen sein und muss entsprechend verwaltet werden. Man hat erkannt, dass man Kontext-informationen sehr gut hierarchisch organisieren kann. So beschreiben zum Beispiel Schilit & Theimer (1994) und Schmidt, Beigl & Gellersen (1999) eine hierarchische Struktur des Kontextes. Dey & Abowd (2000) gehen darüber hinaus davon aus, dass es vier Wurzelknoten der Hierarchie gibt. Für sie bestehen die Wurzelknoten (auch Primärkontexte) aus Ort, Zeit, Identität und Aktivität.

Jedes einzelne Element einer derartigen Hierarchie kann als Kontextattribut bezeichnet werden. Jedes Kontextattribut kann wiederum zu einem Zeitpunkt nur einen Wert haben. Die Werte müssen erfasst und gepflegt werden. Anwendungen benötigen meist nur eine Untermenge aller Kontextattribute. Geht man nach den Primärkontexten von Dey & Abowd (2000), hat man es mit genau vier Kategorien von Kontextattributen zu tun. Bei der Verwendung in der Praxis gibt es allerdings feine Unterschiede zwischen einzelnen Kontextattributen und ihre Anzahl kann variieren.

Analog zu Weiser (1991) könnte man auch den Kontext als allgegenwärtig (ubiquitär) bezeichnen. Man muss ihn bei der Kommunikation zwischen Mensch und Maschine nur nutzen, was wiederum voraussetzt, dass er abrufbar ist. Durch die steigende Anzahl an verfügbaren Geräten, Sensoren und Schnittstellen zwischen den Anwendungen, kann er immer besser überall und jederzeit erfasst und damit auch abgerufen werden. Komponenten zur Erfassung und Auswertung von Kontextinformationen können auf unterschiedlichster Hardware ablaufen. Hard- und Software bilden gemeinsam kontextbasierte Systeme, die von Natur aus auch verteilte Systeme sind.

Für Entwickler kontextbasierter Systeme sollte die Verteilung der Software möglichst transparent sein. Diese Transparenz kann durch eine *kontextbasierte Middleware* unterstützt werden. Eine solche Softwareschicht stellt den Kontext, unabhängig von seinem Ursprung, zur Verfügung und separiert die Erfassung der Kontextattribute von deren Verwendung.

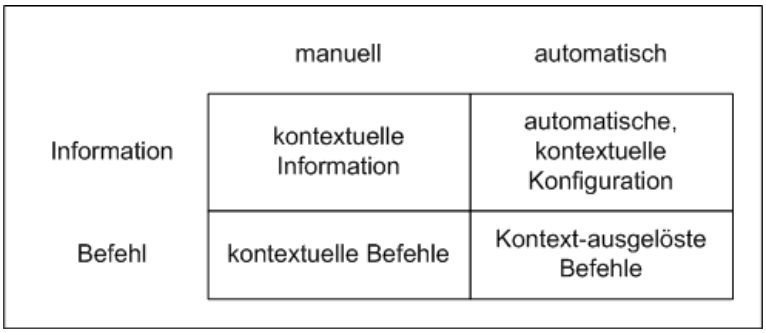

Abbildung 2: Dimensionen kontextbasierter Software. In Anlehnung an Schilit, Adams & Want (1994).

Für kontextverarbeitende Anwendungen werden durch eine kontextbasierte Middleware spezielle Dienste zur Verfügung gestellt, welche die Nutzung von Kontextinformationen vereinfachen. Schilit, Adams & Want (1994) unterscheiden zwischen vier Kategorien von kontextbasierter Verarbeitung (siehe Abbildung 2), von denen mögliche Dienste einer kontextbasierten Middleware abgeleitet werden können:

- **Kontextuelle Information:** Hierunter versteht man die Bereitstellung von kontextbasierten Inhalten durch ein Anwendungssystem. Zum Beispiel wird derzeit in vielen Anwendungen eine Historie von Begriffen, die in Bedienelementen eingegeben wurden, geführt. Diese Historie könnte auch aus dem aktuellen Kontext ermittelt werden. Eine kontextbasierte Middleware kann dafür eine Auswertung von möglichen Informationen basierend auf der aktuell vorliegenden Umgebung anbieten.

- **Kontextuelle Befehle:** In Abhängigkeit des Kontextes können Anweisungen an Geräte unterschiedlich ausfallen. Beispielsweise verändern sich Bedienelemente mobiler Endgeräte je nachdem wie die aktuelle Ausrichtung eines Gerätes, die durch einen Lage-Sensor ermittelt werden kann, ist. Die graphische Oberfläche kann dadurch optimal an den Blickwinkel des Benutzers angepasst werden.

- **Automatische, kontextuelle Konfiguration:** Damit ist eine automatische Veränderung bzw. Anpassung der Konfiguration einer Anwendung aufgrund des aktuellen Kontextes gemeint. Zum Beispiel könnten beim Drucken über ein mobiles Endgerät automatisch die Treiber der Geräte in der Nähe installiert und die vorhandenen Drucker als Standarddrucker ausgewählt werden. Eine kontextbasierte Middleware könnte eine auf den Kontext basierende Auswahl von Programmen unterstützen.

- **Kontext-ausgelöste Befehle:** Hierunter versteht man den kontextbasierten Aufruf von Anwendungen. Beispielsweise kann das Benutzerverhalten analysiert werden und die Middleware kann daraus folgern, welche Anwendungen oft in einem bestimmten Kontext benutzt wurden. Aufgrund dieser Information können dem Benutzer bestimmte Anwendungen angeboten werden. Ähnliche Mechanismen werden schon beim Vorschlagswesen in Online-Shopsystemen eingesetzt. Darüber hinaus könnte ein Dienst einer kontextbasierten Middleware auch eine automatische Aktion initiieren.

Im Rahmen von Forschungsprojekten entstanden einige Ansätze für die Realisierung einer kontextbasierten Middleware. Als vergleichbare Erkenntnisse sind hier vor allem Dey, Abowd & Salber (2001), Ailisto et al. (2002) und Indulska & Sutton (2003) zu nennen. Jedoch erweist sich die Umsetzung einer generischen Middleware-Architektur als schwierig. Auf der einen Seite ist es nicht leicht, die Komplexität des Kontextes in Software abzubilden, andererseits müssen die Schnittstellen der Middleware für einen Entwickler leicht erlernbar sein. Die Forschung im Bereich der kontextbasierten Middleware ist noch lange nicht abgeschlossen und daher gibt es heute auch noch zu wenig konkrete Erfahrungen in der Entwicklung von kontextbasierten Anwendungen. Konkrete Ansätze, die man nennen kann, sind das Java Context-Awareness Framework (JCAF) von Bardram (2005) und das Context-Toolkit von Dey, Abowd & Salber (2001). Ein wichtiger Aspekt im Rahmen der Weiterentwicklung des Themenkomplexes könnte die Konkretisierung von Kontextverzeichnissen sein. Sinn und Zweck derartiger Verzeichnisse sowie deren Anwendbarkeit sollen im Anschluss anhand von konkreten Anwendungsszenarien aus der Hochschule erörtert werden.

Hochschul-Anwendungsszenarien

Im Rahmen unseres Forschungsprojekts untersuchen wir anhand mehrerer typischer Anwendungsszenarien aus dem Hochschulumfeld, wie man Kontextverzeichnisse praktisch nutzen kann. Folgende Szenarien werden derzeit näher betrachtet:

1. Das schwarze Brett einer Fakultät oder Hochschule zeigt immer aktuelle Nachrichten über Termine, Fristen, Prüfungsinformationen, Stundenplanänderungen und andere Ankündigungen. Es ist heute meist über das Internet erreichbar. Jedoch kann es passieren, dass ein Studierender eine Nachricht verpasst, da er nicht regelmäßig diese Seite abruft. Je nach Kontext des Studierenden könnte man über ein Kontextverzeichnis über die Attribute Ort, Zeit und Aktivität die Information auswerten und unterschiedlich darauf reagieren. So ist z.B. eine Nachricht über eine Klausur in der Prüfungszeit wichtiger als außerhalb der Prüfungszeit.

2. Wenn die Hochschule betreten wird, könnte eine auf dem Mobiltelefon laufende kontextbasierte Anwendung den Studierenden automatisch in einen Chatraum einloggen. Dort könnte der Studierende sofort erkennen, welche anderen Kommilitonen in der Hochschule sind. Der aktuelle Aufenthaltsraum des Studierenden könnte erfasst und anderen Studierenden mitgeteilt werden. So wäre eine effektivere Projektarbeit und Kommunikation möglich. Der Kontext könnte über ein Kontextverzeichnis ermittelt werden.

3. Eine sehr wichtige Information für Studierende ist der Stundenplan. Die Studierenden müssen am Anfang eines Semesters nachsehen, wo welche Vorlesung stattfindet und was welcher Dozent anbietet. Um diese Information jederzeit und überall zu erhalten, wäre eine mobile Anwendung sinnvoll, welche einen Zugriff auf den Stundenplan ermöglicht und je nach Kontext des Studierenden besondere Hinweise gibt. Der Kontext könnte ebenso über ein Kontextverzeichnis ermittelt werden.

In Abbildung 3 sind die drei genannten Anwendungsfälle skizziert. Die dickeren Linien sollen eine Instanziierung der Anwendung bei Eintreffen eines Kontextes andeuten. Sobald ein Studierender die Hochschule betritt, werden in unseren Szenarien das Chat-Programm und der Stundenplan

automatisch aufgerufen und können – mit oder ohne Interaktion des Studierenden – Aktionen durchführen.

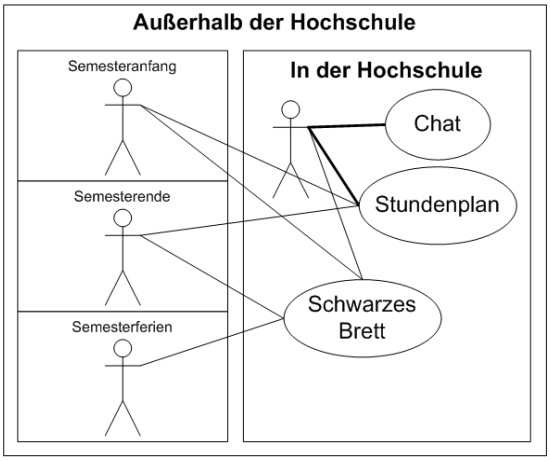

Abbildung 3: Kontextbasierte, mobile Informationsversorgung an der Hochschule.

Anforderungen und Implementierungsüberlegungen für Kontextverzeichnisse

Ein Kontextverzeichnis dient als wichtiger Bestandteil einer kontextbasierten Middleware. Sie ermöglicht eine zentralisierte Verwaltung von allen Kontextinformationen nicht nur für eine dedizierte Anwendung, sondern global für beliebige kontextbasierte Anwendungen. Dabei stellt sich die Frage, wie man Kontext in einer für die elektronische Verarbeitung geeigneten Form speichert.

Strang & Linnhoff-Popien (2004) geben hier einen Überblick über mehrere Alternativen. Sie untersuchten verschiedene Entwurfsmuster auf ihre Tauglichkeit für kontextbasierte Anwendungen, von denen vor allem der Ontologie-basierte und der Markup-Schema-Entwurf interessant sind. Der Ontologie-basierte Entwurf (engl. Ontology Base Model) erschien vor allem für die Darstellung von Kontext als sehr geeignet. Der Markup-Schema-Entwurf (engl. Markup Scheme Model) scheint allerdings besser für die Anwendbarkeit in existierenden Umgebungen zu sein. In mehreren Projekten wurde gezeigt, dass diese Struktur sehr geeignet für die Verwaltung von Kontextinformationen ist (siehe Held, Buchholz & Schill (2002) und Indulska et al. (2003)). Der Aufbau des Kontexts ist bei diesem Ansatz baumartig, als Notation wird die Extensible Markup Language (XML) verwendet.

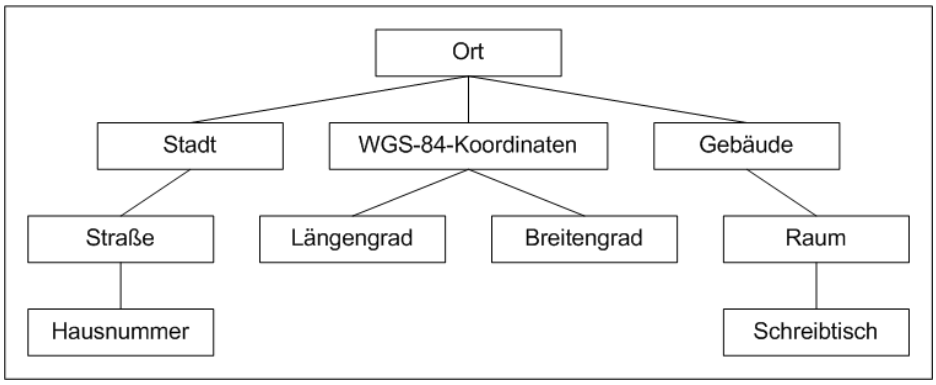

Abbildung 4: Beispiel des Kontext Ort.

Möchte man beispielsweise die Kontextinformation für den *Ort* beschreiben, könnte man sich eine Hierarchie wie in Abbildung 4 vorstellen. Anwendungen für den Bürobereich könnten beispielsweise direkt die Kontextattribute des „Schreibtisches“ nutzen. Weiterhin fällt hierbei auf, dass eine Ableitung von Attributen aus anderen Attributen in der Hierarchie leicht möglich ist. So kann z.B. aus der World-Geodetic-System-84-Koordinate mit Hilfe von Kartenstrukturen die Adresse abgeleitet werden.

Wichtig für die Implementierung eines Kontextverzeichnisses sind vor allem eine gute Anpassungsfähigkeit, um aus dem Verhalten des Benutzers zu lernen, eine einfache Erweiterbarkeit (z.B. über eine Plugin-Architektur), Plattformunabhängigkeit (Vielzahl an mobilen Geräten müssen unterstützt werden) und die Beachtung der Einschränkungen mobiler Geräte (Bildschirmgröße, Speicherknappheit, Stromverbrauch, nicht immer am Netz, ...).

In Abbildung 5 ist eine mögliche Struktur einer kontextbasierten Middleware mit einzelnen Komponenten dargestellt. Wie man sieht ist das Kontextverzeichnis das Herzstück der Architektur.

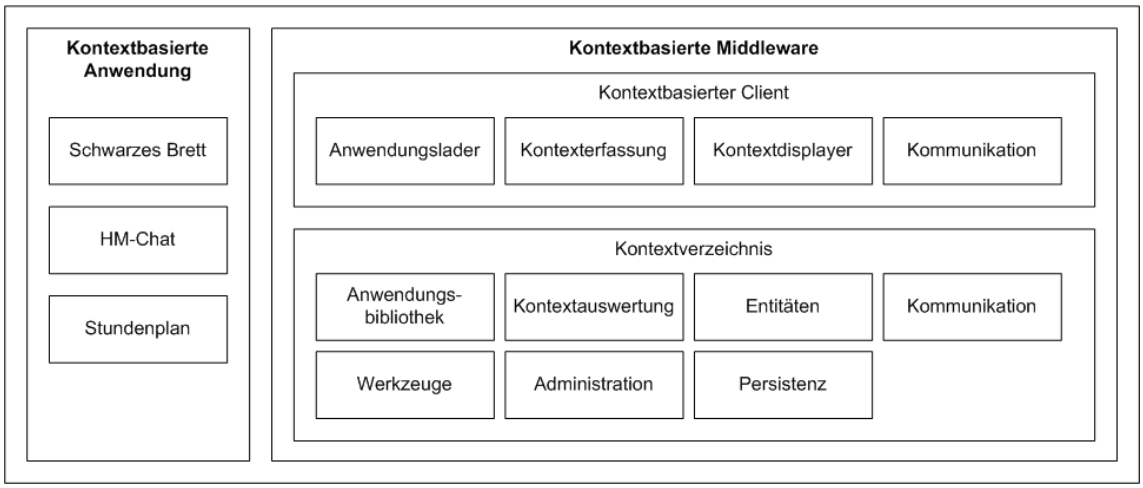

Abbildung 5: Architektur einer kontextbasierten Middleware mit einem Kontextverzeichnis

Die kontextbasierte Middleware kann im Sinne des Client/Server-Modells in eine Client- und eine Serverseite aufgeteilt werden. Der Server verwaltet das Kontextverzeichnis und stellt die Dienste für die Nutzung durch Clients bereit. Auf dem Client, der auf einem mobilen Gerät platziert wird, befindet sich ein Hintergrundprozess für die Erfassung der Kontextdaten und für den automatischen Aufruf von Anwendungen.

Auf eine detaillierte Beschreibung aller Komponenten der Middleware kann in diesem Beitrag nicht eingegangen werden Daher werden nur die wichtigsten Aspekte für eine Implementierung kurz erläutert. Bei der Implementierung der Serverseite sind folgende Anforderungen zu beachten:

- Der Kontext und die Benutzerinformationen sowie eine Kontexthistorie des Benutzers und seiner jeweiligen Anwendungsaufrufe müssen persistent gespeichert werden.
- Ein Auswerten der Kontextattribute und der Kontexthistorie, das Feststellen von gleichartigem Nutzerverhalten und das Prognostizieren von zukünftigen Nutzerverhalten auf Basis des Kontextes muss unterstützt werden.
- Administrationsmöglichkeiten müssen angeboten werden, welche die Benutzer- und Kontext-verwaltung ermöglichen. Eine Konfigurierung der Auswertungsmöglichkeiten sollte gegeben sein.

- Eine asynchrone und eine synchrone Kommunikation zwischen Client und Server müssen über ein standardisiertes Protokoll möglich sein. Bei der Kommunikation müssen die Einschränkungen der mobilen Funknetze beachtet werden. So ist es z.B. nicht möglich einen Client direkt anhand seiner IP-Adresse zu identifizieren, da er sich von einer Zelle in die Nächste oder komplett aus dem Abdeckungsbereich hinaus bewegen könnte.

Auf der Clientseite sind vor allem folgende Funktionen wichtig:

- Die Benutzerdaten und Kontextattribute müssen an den Server übergeben werden. Dabei ist es hilfreich, möglichst viele Kontextattribute zu sammeln und aktuell zu halten.

- Zur Analyse des Kontextes kann eine Oberfläche hilfreich sein, welche auch das manuelle Überschreiben eines Kontextattributes erlaubt. Dadurch kann sich ein Anwender erkundigen, wie sein Kontext gerade beschaffen ist.

- Ein Anwendungslader für das automatische Aufrufen von kontextbasierten Anwendungen aus einer Anwendungsbibliothek heraus muss vorhanden sein, der auch Benutzer-Interaktionen wie z.B. das Ablehnen eines Anwendungsstarts zulässt.

Fazit und Ausblick

In diesem Beitrag wurde ein Einblick in die kontextbasierte Entwicklung und speziell in die Idee und in die Funktionsweise von Kontextverzeichnissen als Bestandteil einer kontextbasierten Middleware gegeben. Anforderungen und erste Architekturansätze wurden skizziert und die grobe Funktionalität eines Kontextverzeichnisses beschrieben.

Anhand praktischer Anwendungsszenarien für den studentischen Alltag soll im weiteren Projektverlauf die Nützlichkeit von Kontextverzeichnissen untersucht und erprobt werden. Das nächste Ziel ist es, eine kontextbasierte Middleware einschließlich eines Kontextverzeichnisses prototypisch zu implementieren. In einem Seminar im Bachelorstudiengang Wirtschaftsinformatik wird ein Kontextverzeichnis und ein zugehöriger Client für die mobile Plattform *Google Android* entwickelt. Hierbei wird das *Google Android Plugin* für Eclipse zum Einsatz kommen. Die Serverseite wird komplett in Java programmiert, wobei die Architektur auf der *Open Services Gateway Initiative* (OSGi) basiert. Für die erste Implementierung wird das Open-Source-Projekt *Apache Felix* verwendet. Zur Speicherung der Kontextdaten soll eine *Couch-DB*-Instanz als OSGi-Service genutzt werden.

Literaturverzeichnis

Ailisto, H. et al., 2002. Structuring Context Aware Applications : Five-Layer Model and Example Case. *Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing* S. 1-5.

Bardram, J.E., 2005. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In *PERVASICE 2005*. Berlin, Heidelberg: Springer S. 98-115.

Dey, A.K. & Abowd, G.D., 2000. Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*. S. 304 - 307. Verfügbar unter: http://nl.ijs.si/~damjan/NeOn/99-22.pdf.

Dey, A.K., Abowd, G.D. & Salber, D., 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2) S. 97 - 166.

Held, A., Buchholz, S. & Schill, A., 2002. Modeling of Context Information for Pervasive Computing Applications. In *Proceedings of SCI 2002/ISAS 2002*. S. 1-6.

Hewett, T.T., 1992. ACM SIGCHI curricula for human-computer interaction. p. 162. Verfügbar unter: http://portal.acm.org/citation.cfm?id=133967.

Indulska, J. & Sutton, P., 2003. Location Management in Pervasive Systems. *Conferences in Research and Practice in Information Technology Series*, 34 S. 143 - 151. Verfügbar unter: http://portal.acm.org/citation.cfm?id=827987.828003.

Indulska, J. et al., 2003. Experiences in Using CC/PP in Context-Aware Systems. *Lecture Notes In Computer Science; Vol. 2574* S. 247. Verfügbar unter: http://portal.acm.org/citation.cfm?id=747270.

Lieberman, H. & Selker, T., 2000. Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3-4) S. 617632. Verfügbar unter: http://www.hctp.utoronto.ca/Pdf/Technology Course Readings/Out_of_Context.pdf.

Löwe, R. & Mandl, P., 2010. Mobile, kontextbasierte Informationsbewertung und -versorgung am Beispiel von MEC +. *erscheint vorraussichtlich 2010 in PIK - Praxis der Informationsverarbeitung und Kommunikation*.

Norman, D.A., 1998. *The invisible computer*, Cambridge, Massachusetts, USA: MIT PRESS.

Pettey, C. & Stevens, H., 2010. Gartner Says Context-Aware Computing Will Provide Significant Competitive Advantage. *Gartner Press Releases* S. 1. Verfügbar unter: http://www.gartner.com/it/page.jsp?id=1190313.

Schilit, B., Adams, N. & Want, R., 1994. Context-aware computing applications. In *Workshop on Mobile Computing Systems and Applications*. IEEE Comput. Soc. Press S. 85-90. Verfügbar unter: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=512740.

Schilit, B. & Theimer, M., 1994. Disseminating active map information to mobile hosts. *IEEE network*, 8(5) S. 22–32. Verfügbar unter: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.6762&rep=rep1&type=pdf.

Schmidt, A., Beigl, M. & Gellersen, H., 1999. There is more to context than location. *Computers & Graphics*, 23(6) S. 893-901. Verfügbar unter: http://linkinghub.elsevier.com/retrieve/pii/S009784939900120X.

Schulz Von Thun, F., 1981. *Miteinander reden: Stoerungen und Klaerungen: Psychologie der zwischenmenschlichen Kommunikation, Band 2*, Rowohlt. Verfügbar unter: http://www.amazon.com/Miteinander-reden-Psychologie-zwischenmenschlichen-Kommunikation/dp/3499174898.

Strang, T. & Linnhoff-popien, C., 2004. A Context Modeling Survey. In *Proceedings of UbiComp: 1st International Workshop on Advanced Context Modelling, Reasoning and Management*. Nottingham, UK.

Weiser, M., 1991. The computer for the 21st Century. *IEEE Pervasive Computing*, 99(1) S. 19-25. Verfügbar unter: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=993141.

Kontakt

Ralph Löwe
Hochschule München
Lothstraße, 34
D-80335 München

Telefon: +49 (0) 89 1265 3776
Fax: +49 (0) 89 1265 3780
E-Mail rloewe@hm.edu
Internet: http://www.cs.hm.edu

Prof. Dr. Peter Mandl
Hochschule München
Lothstraße, 34
D-80335 München

Telefon: +49 (0) 89 1265 3704
Fax: +49 (0) 89 1265 3780
E-Mail mandl@cs.hm.edu
Internet: http://www.cs.hm.edu