Kontextsensitive Dienste auf Basis von Open-Source-Software

Jörn von Ahsen*
jva@w9p.de

Maximilian Bergmann*
maxbergmann@gmail.com

Thorsten Teschke
Hochschule Bremen
thorsten.teschke@hs-bremen.de

Abstract: Dieser Beitrag fasst die Ergebnisse einer Diplomarbeit zusammen, in der ein Konzept für die Bereitstellung von Diensten sowie deren kontextsensitive Auswahl und Nutzung entwickelt und prototypisch umgesetzt wurde. Die Umsetzung erfolgte dabei auf Basis von Open-Source-Software und offenen Standards aus dem Bereich des Semantic Web.

1 Einführung

Der Begriff Kontext hat im Zuge der Diskussionen und Entwicklungen im Ubiquitous Computing [Wei91] eine herausragende Bedeutung erhalten. Ubiquitous Computing hat das Ziel, den Computer „unsichtbar“ in unsere Interaktion mit der Umwelt zu integrieren – sowohl physisch als auch sozial. Damit verbindet Ubiquitous Computing zentrale Ideen des Mobile Computing und des Pervasive Computing miteinander [LY02]. Der Computer wird zu einem alltäglichen und (mehr oder weniger) unsichtbaren Begleiter, den wir entweder mit uns führen (z. B. als PDA oder integriert in Kleidung) oder aber in unserer Umgebung vorfinden (z. B. in Form von Displays und Sensoren). Die Fähigkeit, Modelle seiner Umwelt anzulegen und sein Verhalten daran anpassen, d. h. seinen Kontext zu berücksichtigen, machen den Computer „intelligent“ und eröffnen neue Möglichkeiten jenseits von Desktop-Anwendungen und konventionellen mobilen Anwendungen.

In einer verbreitet verwendeten Definition bezeichnen Dey et al. *Kontext* als „jegliche Information, die genutzt werden kann, um die Situation von Entitäten [...] zu charakterisieren, die als relevant für die Interaktion zwischen einem Anwender und einer Anwendung angesehen werden, einschließlich des Anwenders und der Anwendung selbst“ (übersetzt nach [DAS01]). Unter Entitäten verstehen Dey et al. dabei Personen, Orte und Objekte. Obwohl diese Definition Kontext lediglich als statisches Phänomen darstellt, bei dem Eigenschaften der Welt a priori Kontext *sind* (oder eben nicht), anstatt erst aufgrund von Aktivitäten relevant und damit zu Kontext zu *werden*, wird sie als Grundlage dieser Arbeit verwendet. Ein System ist „*kontextsensitiv*, wenn es Kontext benutzt, um dem Anwender Zugriff auf relevante Informationen und/oder Dienste zu ermöglichen, wobei die Relevanz von der Aufgabe des Anwenders abhängt“ (übersetzt nach [AD99]).

*Jörn von Ahsen und Maximilian Bergmann haben an der Hochschule Bremen Medieninformatik (Diplom (FH)) studiert.

Im Fokus dieser Arbeit stehen die Bereitstellung von möglicherweise nur lokal verfügbaren Diensten sowie deren kontextsensitive Auswahl und Nutzung mit mobilen Endgeräten. Zwei exemplarische Szenarios sollen helfen, das Aufgabengebiet genauer zu umreißen. Ein lokaler *Fahrplan-Dienst*, der z. B. in einem Bahnhof zur Verfügung gestellt wird, kann ortsfremden ankommenden Reisenden aufgrund von Informationen über deren endgültiges Reiseziel (z. B. Termin bei einem Kunden) eine geeignete Verbindung mit dem öffentlichen Personennahverkehr vorschlagen. Angekommen beim Kunden kann ein *Raumplan-Assistent* den Reisenden mit Hilfe von Informationen über den aufzusuchenden Gesprächspartner eine geeignete Wegbeschreibung durch das Gebäude leiten. Diese und ähnliche Szenarios weisen die folgenden charakteristischen Eigenschaften auf:

1. Die angebotenen (lokalen) Dienste werden nicht auf den mobilen Endgeräten der Benutzer installiert, sondern lediglich server-seitig zur Nutzung über ein Netzwerk bereitgestellt. Die Unterstützung eines Anwenders durch Dienste im Sinne des Ubiquitous Computing kann nicht auf der Annahme beruhen, dass Anwender regelmäßig verfügbare Dienste untersuchen, bewerten und ggf. auf ihren Endgeräten installieren (und später wieder entfernen).

2. Das mobile Endgerät des Anwenders ist in der Lage, verfügbare Dienste zu finden, ihre Eignung in der gegebenen Situation zu bewerten und ggf. zu nutzen bzw. einen Vorschlag für die Nutzung zu unterbreiten. Dies beinhaltet eine Analyse verfügbarer Kontextinformationen sowie des angebotenen Dienstes.

3. Der Kontext des Anwenders (z. B. Aufenthaltsort, Aufgaben, Historie) wird durch sein mobiles Endgerät stets aktuell erfasst. Dazu gehören in den zuvor genannten Szenarios Informationen über den Aufenthaltsort des Anwenders, seine Aufgaben (Termin mit Kunden an bestimmtem Ort und zu bestimmter Uhrzeit) sowie seine Erfahrungen im Sinne einer „Historie“ (ist der Anwender ortsfremd oder hat er seinen Gesprächspartner früher bereits besucht?).

Ziel dieser Arbeit war die prototypische Entwicklung einer Lösung, die die Bereitstellung von Diensten, die Erfassung des aktuellen Kontexts, die Auswahl von Diensten und deren Nutzung auf Basis von Open-Source-Software und offenen Standards ermöglicht. Die Wahl von Open Source und offenen Standards war dabei u. a. motiviert durch Argumente wie Kosten, Unabhängigkeit / Portabilität und die Möglichkeit, Modifikationen am Quelltext vorzunehmen.

2 Verwandte Arbeiten

Im Umfeld der hier betrachteten Problemstellung finden sich eine Reihe von Arbeiten, die jedoch oft nur Teile des Themengebiets abdecken. So wurde z. B. im Projekt *BAMOS (Basisarchitektur für mobile Anwendungen in spontanen Netzen)* [SPB+05] eine Architektur konzipiert und umgesetzt, in der mobile Endgeräte mittels Bluetooth auf Service Broker zugreifen und über diese verfügbare Dienste finden und ausführen können. Da die

verwendeten Dienstbeschreibungen keinerlei semantische Informationen enthalten, wird die (halb-)automatische Dienstauswahl in BAMOS nicht unterstützt. Für die Dienstauswahl wird in [KR07] ein Verfahren vorgeschlagen, das zum einen auf der Analyse von WSDL-Beschreibungen von Web Services mit Methoden des Information Retrieval basiert und zum anderen das Feedback von Benutzern mit ähnlichem Kontext berücksichtigt. Während letzteres durchaus interessant erscheint, kann die Analyse von syntaktischen Informationen wie z. B. Methoden- oder Parameternamen nach unserer Einschätzung jedoch keine verlässlichen Informationen für eine automatisierte Dienstauswahl liefern. Die Ausführung von Diensten im Anschluss an deren Auswahl wird in der Arbeit von Kuck und Reichartz nicht betrachtet.

Verschiedene weitere Arbeiten zur kontextsensitiven Nutzung von Diensten verfolgen insbesondere hinsichtlich ihrer Zielsetzung und der Modellierung von Kontext unterschiedliche Ansätze. Das *Context Toolkit* [DAS01] verwendet einfache Key-Value-Paare, ist dadurch aber in der Ausdrucksmächtigkeit stark reduziert. *CAPEUS* [SMLP01] erweitert das Key-Value-Prinzip um Constraints, mit denen Relationen und Ereignisse formuliert werden können. Das *Java Context Aware Framework (JCAF)* [Bar05], ein Java-Framework zur Entwicklung kontextsensitiver Anwendungen, erfordert die Implementierung einer konkreten Java-Klasse für jede für den Kontext relevante Entität und ist damit vergleichsweise unflexibel. Ontologie-basierte Ansätze wie das *DIANE-Projekt (Dienste in Ad-hoc-Netzen)* [Kla06] zeichnen sich durch eine hohe Ausdrucksmächtigkeit und die Unterstützung bei der Auswahl von Diensten durch Matching-Verfahren aus. Eine detailliertere Vorstellung dieser und weiterer verwandter Arbeiten findet sich in [vAB08].

Grundsätzlich ist festzustellen, dass keine der Arbeiten direkt zur Lösung des vorliegenden Problems genutzt werden kann, da entweder die Ausdrucksmächtigkeit nicht ausreicht, funktionale Beschränkungen vorliegen (z. B. bei Dienstauswahl, -ausführung oder Formulierung von Dienstanfragen), proprietäre Beschreibungssprachen eingesetzt werden oder schlicht Dokumentation, Implementierung oder Kontextmodelle nicht verügbar sind.

3 Konzeption

In Abschnitt 1 wurde die prototypische Realisierung einer Lösung gefordert, die die Bereitstellung von Diensten, die Erfassung des aktuellen Kontexts, die Auswahl von Diensten und deren Nutzung auf Basis von Open-Source-Software und offenen Standards ermöglicht. Abbildung 1 zeigt die logische Architektur, die im Rahmen dieser Arbeit entwickelt wurde und die Grundlage für die prototypische Umsetzung bildet.

Der Kern des mobilen Clients übernimmt primär steuernde Aufgaben und verknüpft die Komponenten der logischen Architektur. Darüber hinaus ist ihm auch die Kommunikation mit dem Dienstanbieter zwecks Dienstauswahl und -ausführung zugeordnet. Die Dienstauswahl (Matching) wird dabei auf dem mobilen Endgerät durchgeführt, um insbesondere nicht alle (privaten!) Kontextinformationen an einen unbekannten Server übermitteln zu müssen. Die Ausführung eines relevanten Dienstes mit aktuellen Kontextinformationen verbleibt dagegen auf dem Server des Dienstanbieters.

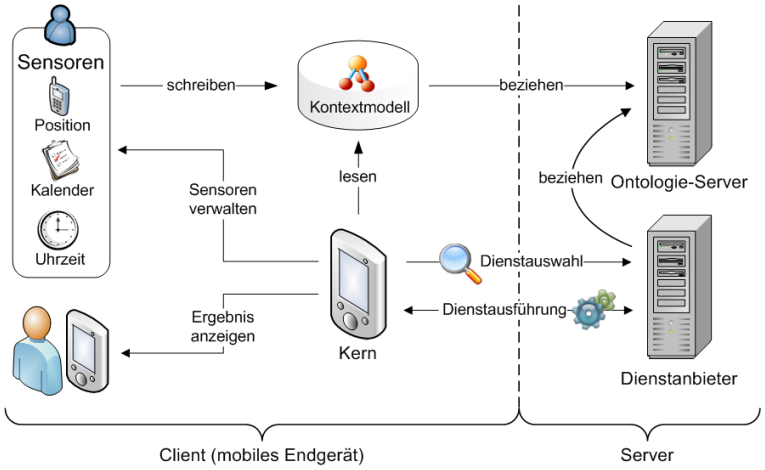

Abbildung 1: Logische Architektur

Als potentielle Dienste sollen existierende Web Services integriert werden können, denen neben ihrer WSDL-Beschreibung zwecks Dienstauswahl noch eine semantische Beschreibung auf Basis von OWL-S[1] zugeordnet werden muss. OWL-S gestattet u. a. die automatische Suche nach Diensten und deren Ausführung.

Als gemeinsame semantische Grundlage für das client-seitig zu pflegende Kontextmodell und server-seitig angebotene Dienste wurde eine Ontologie angelegt, deren Definition auf einem Server bereitgestellt wird. Kontextinformationen werden client-seitig über Sensoren gesammelt, aggregiert und in das lokale Kontextmodell eingestellt. Formale Grundlage der Ontologie bzw. des Kontextmodells ist OWL[2], wobei die Variante OWL-DL (Description Logics) gewählt wurde, die einerseits eine ausreichende semantische Ausdruckskraft aufweist und andererseits beweisbar bleibt. Das Kontextmodell ist zusätzlich mit Hilfe von Reasoning-Techniken dafür zuständig, die Konsistenz der Daten sicherzustellen und ggf. neue Informationen aus anderen abzuleiten.

Die Dienstauswahl basiert im Kern auf dem semantischen Matching von Parameter- und Rückgabetypen von Diensten mit Informationen aus dem Kontextmodell eines Anwenders. Den Typinformationen aus der WSDL-Beschreibung eines angebotenen Dienstes werden in den bereits genannten OWL-S-Dienstbeschreibungen Konzepte aus der Ontologie und damit eine Semantik zugeordnet. Das Matching erfolgt durch Vergleich der von einem Dienst geforderten bzw. zurückgegebenen Konzepte mit den im Kontextmodell des Anwenders gespeicherten Konzepte, wobei Subtyp-Beziehungen berücksichtigt werden. Durch Verwendung verschiedener Matcher können unterschiedlich „scharfe“ Übereinstimmungen geprüft und somit auch ein qualitatives Feedback zur Dienstauswahl gegeben werden.

[1] Web Ontology Language for Web Services, `http://www.w3.org/Submission/OWL-S/`
[2] Web Ontology Language, `http://www.w3.org/TR/owl-features/`

4 Umsetzung

Die im vorangegangenen Abschnitt skizzierten Konzepte wurden mittels der in Abbildung 2 dargestellten Software-Architektur prototypisch umgesetzt. Ein wesentliches Ziel bei der Umsetzung war es, auf Open-Source-Software und offene Standards aus dem Gebiet des Semantic Web zurückzugreifen. Da diese oftmals nicht für mobile Plattformen wie Java ME[3] verfügbar waren, ist der Prototyp auf Basis von Java SE entstanden.

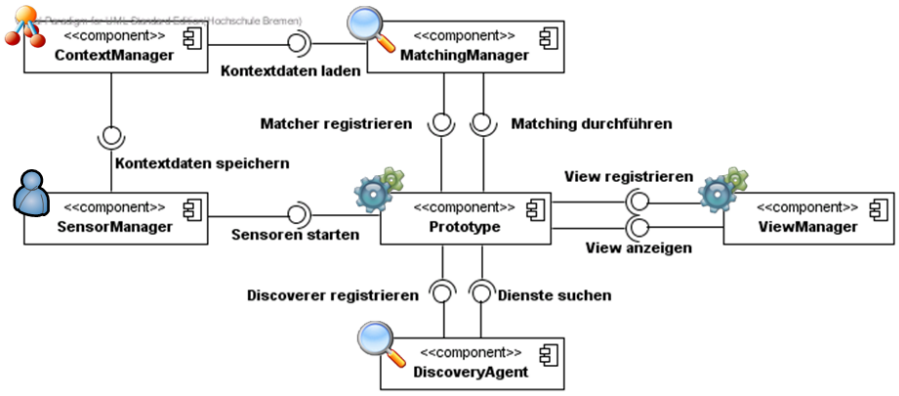

Abbildung 2: Software-Architektur

Kontexterfassung und -verwaltung Die Erfassung und Verwaltung von Kontext wird in der Software-Architektur durch die Komponenten *SensorManager* und *ContextManager* realisiert. Das Kontextmodell in der ContextManager-Komponente basiert auf einer OWL-DL-Ontologie, die mittels *Protegé*[4] realisiert wurde. Protegé ist ein Open-Source-Werkzeug der Stanford University zur Modellierung und Visualisierung von Ontologien. Die eigentliche Kontextdatenbank im ContextManager wurde auf der Grundlage des *OWL-S-API*[5] umgesetzt, einer freien Java-Bibliothek zur Verarbeitung von OWL-S-Dokumenten. OWL-S-API kapselt das freie Semantic-Web-Framework *Jena*[6]. Die Kontextdatenbank nutzt darüber hinaus den freien OWL-Reasoner *Pellet*[7] zur Konsistenzüberwachung.

Der SensorManager zur Ermittlung von Kontextinformationen wurde mit einem flexiblen Plugin-Mechanismus ausgestattet. Dies ermöglicht es, Sensoren für unterschiedliche Datenquellen zur Laufzeit in das System nachzuladen oder daraus zu entfernen. Für den Prototypen wurden u. a. Sensoren für die Bestimmung der aktuellen Position mittels eines Bluetooth-GPS-Empfängers und das Auslesen von Termininformationen aus einem Outlook-Kalender implementiert.

[3] Java Platform, Micro Edition, `http://java.sun.com/javame/index.jsp`
[4] `http://protege.stanford.edu/`
[5] `http://www.mindswap.org/2004/owl-s/api/`
[6] `http://jena.sourceforge.net/`
[7] `http://pellet.owldl.com/`

Dienstauswahl Die Dienstauswahl ist in der Architektur auf zwei Komponenten verteilt. Der *DiscoveryAgent* sucht z. B. über WLAN verfügbare Dienste und gibt entsprechende *Service*-Referenzen auf gefundene Dienste an die zentrale *Prototype*-Komponente zurück. Das Einlesen der mit dem Dienst verbundenen Beschreibung stützt sich dabei auf das OWL-S-API ab. Für die Prüfung, ob und wie ein gefundener Dienst im gegebenen Kontext sinnvoll aufgerufen werden kann, verwendet der *MatchingManager* verschiedene „Matcher", die sequentiell ausgeführt werden, so lange eine gewünschte Übereinstimmung zwischen Anwenderkontext und angebotenem Dienst nicht ermittelt werden konnte. Während *intentionsbasierter* Matcher versucht, die Bedürfnisse des Anwenders gemäß dessen Kontext automatisch zu erkennen und durch geeignete Dienstauswahl zu befriedigen, bindet ein *interaktiver* Matcher den Anwender in den Auswahlprozess ein und gestattet bei relevanten Diensten mit unvollständigem automatischem Matching die manuelle Ergänzung von Parameterwerten.

Dienstausführung Die Ausführung eines Dienstes mit der im Rahmen der Dienstauswahl gefundenen Zuordnung von Konzepten aus dem Anwenderkontext wird durch die *Prototype*-Komponente angestoßen. Die *Service*-Klasse, die den aufzurufenden Dienst repräsentiert, nutzt zu diesem Zweck OWL-S-API, das schon beim Einlesen von Dienstbeschreibungen zum Einsatz kommt. Vor und nach dem eigentlichen Aufruf des Dienstes führt OWL-S-API Umwandlungen von Kontextinformationen in Eingabeparameter bzw. von Rückgabewerten in Kontextinformationen durch. Diese Umwandlungen sind in Form von XSL-Transformationsregeln in der OWL-S-Beschreibung des Dienstes enthalten.

Für die Visualisierung der Ergebnisse eines Dienstaufrufs verwaltet die *ViewManager*-Komponente verschiedene *Views*, die abhängig vom Rückgabetyp eines Dienstes ausgeführt werden. Neben zwei spezifischen Views wurde im Rahmen der prototypischen Umsetzung auch ein generischer View realisiert, der die vollständigen, in einem Rückgabewert enthaltenen Instanzen auf der Konsole ausgibt.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurden ein Konzept für die Bereitstellung von Diensten und deren kontextsensitive Auswahl und Nutzung sowie eine prototypische Umsetzung dieses Konzepts auf Basis von Open-Source-Software und offenen Standards vorgestellt. Eine erste Evaluation der erreichten Ergebnisse hat gezeigt, dass der verfolgte Ansatz zwar grundsätzlich funktioniert, jedoch auch noch einige Schwächen aufweist.

So bereitet die Zuordnung von Kontextinformationen zu Parametern eines Dienstes Probleme, wenn z. B. die Startzeit eines Termins im Fahrplan-Dienst auf eine Ankunftszeit abgebildet werden muss. Eine potentielle Lösung dieses Problems könnte in dem in [KR07] vorgestellten kollaborativen Ansatz, Feedback von verschiedenen Anwendern zu sammeln, in Verbindung mit einem selbstlernenden System bestehen. Darüber hinaus hat sich – wie erwartet – gezeigt, dass die Modellierung einer Ontologie ein schwieriges Unterfangen ist, von dessen Erfolg aber nicht zuletzt die Qualität von Kontextmodell, Dienst-

beschreibungen und Dienstauswahl abhängt. Die Dienstausführung und -visualisierung ist in der vorgestellten Arbeit nur am Rande betrachtet worden. Hier fehlt eine umfassende Lösung, die wie im Projekt BAMOS Interaktionen im Sinne von Workflows unterstützt.

Die in dieser Arbeit entwickelten Konzepte wurden bislang noch nicht auf mobilen Plattformen umgesetzt, weil einerseits die Leistungsfähigkeit der zur Verfügung stehenden Endgeräte nicht für ausreichend befunden wurde und andererseits entsprechende „mobile“ Alternativen zu den genutzten Open-Source-Bibliotheken aus dem Bereich des Semantic Web nicht verfügbar waren. Vor dem Hintergrund der weiter steigenden Leistungsfähigkeit mobiler Endgeräte und mächtigeren Software-Plattformen ist künftig zu untersuchen, inwieweit sich die Ergebnisse auch technisch auf mobile Plattformen übertragen lassen. Hier erscheint uns die Android-Plattform aufgrund ihres Open-Source-Ansatzes und des Einsatzes von Java als Programmiersprache ein vielversprechender Kandidat zu sein.

Literatur

[AD99] Gregory D. Abowd und Anind K. Dey. Towards a Better Understanding of Context and Context-Awareness. Technical Report GIT-GVU-99-22, Georgia Tech, 1999.

[Bar05] Jakob E. Bardram. The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In *In Hans Gellersen, Roy Want, and Albrecht Schmidt, editors, Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), Lecture Notes in Computer Science*. Springer Verlag, 2005.

[DAS01] Anind K. Dey, Gregory D. Abowd und Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2):97–166, 2001.

[Kla06] Friederike Klan. Context-aware service discovery, selection and usage. In *18. Workshop über Grundlagen von Datenbanken*, 2006.

[KR07] Julia Kuck und Frank Reichartz. A collaborative and featurebased approach to Context-Sensitive Service Discovery. In *16th International World Wide Web Conference, 5th WWW Workshop on Emerging Applications for Wireless and Mobile Access (MobEA'07)*, 2007.

[LY02] Kalle Lyytinen und Youngjin Yoo. Issues and challenges in ubiquitous computing: Introduction. *Communications of the ACM*, 45(12):62–65, 2002.

[SMLP01] Michael Samulowitz, Florian Michahelles und Claudia Linnhoff-Popien. CAPEUS: An Architecture for Context-Aware Selection and Execution of Services. In *In Proceedings of the Third IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2001)*, Seiten 23–39, 2001.

[SPB+05] Martin Schmiedel, Oliver Pawlowski, Ralf Bruns, Jürgen Dunkel und Frank Nitze. Mobile Services in Adhoc Networks. In *Proceedings of Net.ObjectDays 2005*, 2005.

[vAB08] Jörn von Ahsen und Maximilian Bergmann. Zentrale Kontexterfassung und kontextsensitive Auswahl lokaler Informationsdienste. Diplomarbeit, Hochschule Bremen, 2008.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.