Context-aware Computing

Ralph Löwe
Competence Center Wirtschaftsinformatik
Hochschule München
rloewe@hm.edu

Schlüsselworte:

Kontext, Ubiquitous Computing, Context-aware Computing, Context-aware Middleware.

Einleitung

Im Bereich der mobilen Endgeräte sind derzeit zwei Trends zu erkennen. Der erste Trend ist, dass mobile Spezialgeräte immer mehr Sensoren erhalten. Hierzu zählen zum Beispiel Mobiltelefon, Navigationsgerät, Tablet-PC oder der Bordcomputer im Auto. Der zweite Trend ist eine Erweiterung der Software, welche meist durch die Nachinstallation von mobilen Anwendungen (sog. Apps) ermöglicht wird. Diese Anwendungen sind speziell für eine Aufgabe geschrieben und erweitern die Möglichkeiten des Ursprungsystems. Hinzu kommt, dass die Hardware mobiler Endgeräte kontinuierlich um neue Sensoren erweitert wird.

Waren die ursprünglichen Mobiltelefone nur mit einem Mikrophon und Lautsprecher ausgestattet sind die heutigen Geräte mit Kamera, Lichtsensor, Bewegungssensor, Positionsbestimmung und neuerdings auch mit Sensoren für den kontaktlosen Austausch von Daten (engl. Near Field Communication) ausgestattet. Dies ermöglicht neue Anwendungsfälle und kann die Möglichkeiten bestehender mobiler Anwendungen nachhaltig verbessern. Zum Beispiel wird gerade an einer mobilen Zahlungsabwicklung für den europäischen Raum gearbeitet Clark (2009). In einigen asiatischen Ländern wird diese Möglichkeit bereits genutzt.

Grundlegend fällt auf, dass der Trend so schnell voranschreitet, dass es der mobilen Softwareentwicklung noch an den Konzepten und Werkzeugen mangelt. Ein Grundproblem ist dabei mit einer großen Menge an Sensoren und mobilen Anwendungen umzugehen. Das herkömmliche, direkten Eingabe/Ausgabe-Konzept stößt hierbei an seine Grenzen. Durch die Beachtung des Kontextes können zusätzlich indirekte Eingaben durch zum Beispiel einen Ortswechsel erfolgen und eine mobile Anwendung könnte darauf reagieren. Es gibt bereits Anwendungen, die Sensordaten verwenden. Jedoch wird hier der Kontext nicht als umfassendes Konzept wahrgenommen. Allerdings wird eine steigende Anzahl an neuen Sensoren in diesem verteilten Umfeld erwartet. Dies führt dazu, dass die mobile Anwendungsentwicklung komplexer wird.

Context-aware Computing

Diese Grundproblem kann durch eine kontextbewusste Anwendungsentwicklung (engl. Context-aware Computing) weiter eingegrenzt werden. So bietet das Feld fachliche und technische Methoden, um mit der Komplexität vieler, gleichzeitiger, direkter und indirekter Eingaben umzugehen. Das übliche Eingabe/Ausgabe-Prinzip wird dafür um den Kontext erweitert. Als Ursprung dieses wissenschaftlichen Gebietes zählt das Active-Badge-System von Want et al. (1992).

Laut Dey und Abowd (2000) ist Kontext jedwede Information, welche genutzt werden kann, um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, Ort oder Objekt, welche als relevant für die Interaktion zwischen Benutzer und Anwendungen angesehen werden. Dies beinhaltete den Benutzer und die Anwendung selbst.

Diese Definition ist sehr weitläufig gefasst. Sie beschreibt allerdings treffend den umfassenden Aspekt von Kontext. Ein Ziel der kontextbewussten Anwendungsentwicklung wird inhärent mit definiert. So muss Kontext relevant für die Anwendung oder den Benutzer sein. Dadurch ist Kontext ein Hilfsmittel für die Kommunikation zwischen Mensch und Maschine. Eine mögliche Strukturierung von Kontext bieten Dey und Abowd (2000): Ort, Zeit, Aktivität und Identität (siehe Abb. 1).

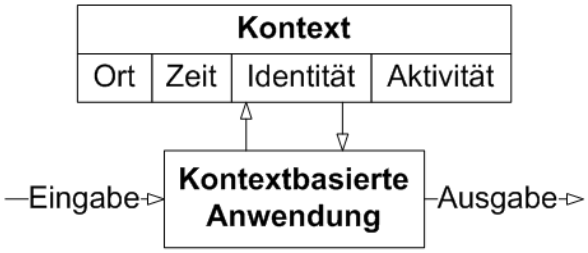

Abb. 1: Kontext in Anlehnung an Dey und Abowd (2000) und Lieberman und Selker (2000).

Die Ermittlung des **Ortes** kann u.a. durch Lokalisationsverfahren wie das Global Positioning System (GPS), mobilfunkgestützt oder auch WLAN erfolgen. Als Ergebnis liegen dann Informationen wie Land, Stadt, Adresse, Bundesland, Gebäude, Raum oder die reine World-Geodetic-System-84-Koordinate vor. Der Bereich der Ortsbasierten Dienste erforscht dieses Anwendungsfeld.

Zeit kann und wird bei vielen Anwendungen in ihrer rohen Form verwendet. In Kombination mit anderen Elementen des Kontext können jedoch neue Formen wie tagsüber, nachts, mittags oder Feiertag, Arbeitstag, Werktag verwendet werden.

Wird die **Aktivität** erfasst, kann eine kontextbewusste Anwendung auf unterschiedliche Art und Weise auf den Zustand des Benutzers reagieren. Ausprägungen der Aktivität sind zum Beispiel: laufend, gehend, autofahrend, schlafend oder redend. So kann beispielsweise ein eingehender Anruf je nach Aktivität unterschiedlich behandelt werden.

Der Aspekt der **Identität** ist sehr relevant für den Bereich der Datensicherheit. Eine Authentifizierung mittels Erkennen des Benutzers über den Kontext, kann die sichere Verwendung von Anwendungen und Geräten stark erleichtern. Weiterhin kann eine Personalisierung durch sog. Profile erfolgen, welche sich insbesondere über die Identität und zusätzlich über weitere Elemente des Kontextes an den Benutzer anpassen können.

Bei der Strukturierung in diese vier Bestandteile handelt es sich um die am weitesten verbreitete Möglichkeit. Zimmermann et al. (2007) erweitern diese Aufteilung um die Beziehungen (engl. relations). Weiterhin fügen sie eine weitere Dimension hinzu, welche den Fokus auf die Gruppen eines konkreten Kontextes beschreibt. Durch einen Wechsel des Fokus kann somit ein Kontextwechsel beschrieben werden. Wie in Abb. 2a beschrieben kann zum Beispiel der Ort unwichtiger werden und dafür die Zeit und Identität wichtiger.

Als weiterer Untersuchungsgegenstand ergibt sich hieraus der gemeinsame Kontext (engl. shared context). Implizit wird dieses Prinzip bei einem großen Online-Händler schon verwendet. Können zwischen den Kontexten von zwei Personen Gemeinsamkeiten festgestellt werden, können Aktionen, Aussagen u.ä. von der einen auf die andere Person übertragen werden. Hat zum Beispiel ein Kunde ein Produkt gekauft und der nächste hat einen ähnlichen Kontext, bietet man ihm das gleiche Produkt an (siehe Abb. 2b).

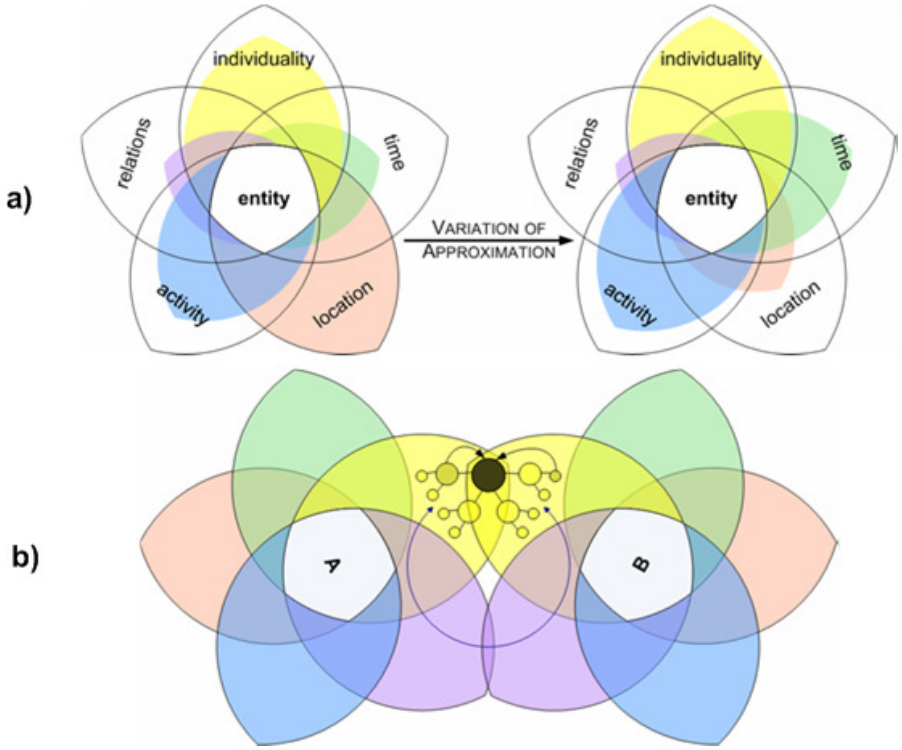

Abb. 2: Kontextwechsel und geteilter Kontext nach Zimmermann et al. (2007).

Damit ist der Kontextwechsel des Benutzers der zentrale Gegenstand der kontextbewussten Anwendungsentwicklung. Durch eine Analyse von ihm können sich Anwendungen besser an den Benutzer anpassen. Die Struktur von Kontext und seine Analyse machen kontextbewusste Anwendungen in der Programmierung komplexer. So sind zum Beispiel graphische Bedienelemente, Vorbelegungen und auch Abläufe ohne einen konkreten Kontext nicht mehr genau vorhersagbar.

Context-aware System

Ein kontextbewusstes System ist ein Hard- und Software-System, welches Kontext nutzt, um sich besser an den Benutzer anzupassen. Dey and Abowd (2000) definieren, dass ein System kontextbewusst ist, wenn es Kontext nutzt, um relevante Informationen und/oder Dienste für den Benutzer anzubieten. Die Relevanz ist dabei abhängig von der Aufgabe des Benutzers (freie Übersetzung des Autors).

Ein wichtiger Faktor ist hierbei die Relevanz. So muss das System auf das Anwendungsziel konzentriert den Kontext auswerten und als Ergebnis darauf reagieren. Dieses Wissen hat meist nur der Anwendungsentwickler selbst und dieser kann dann entscheiden, wie das System reagieren soll. Als Komponenten einer kontextbewussten Anwendungsentwicklung haben Malik und Mahmud (2007) folgende Felder identifiziert. Sie werden von der Erfassung bis zur konkreten daraus folgenden Aktion aufgelistet (siehe Abb. 3).

Zunächst muss eine **Akquisition des Kontext** (engl. Context Acquisition) erfolgen. Hierbei werden mögliche Sensoren aufgerufen und die Bestandteile des Kontextes gesammelt. Dabei können die Sensoren Bestandteil des mobilen Endgerätes oder auch der Umgebung sein.

Anschließend muss eine interne **Darstellung des Kontext** (engl. Context Representation) im System erfolgen. Dabei werden alle Attribute des Kontextes zusammen betrachtet und meist in einem mehrdimensionalen System eingeordnet. Ein Punkt in diesem System entspricht dann genau einem konkreten Kontext. Dieser Bereich wird auch als Kontextmodellierung (engl. Context Modeling) bezeichnet und wurde zum Beispiel von Strang und Linnhoff-popien (2004) beschrieben.

Bei der **Speicherung des Kontext** (engl. Context Storage) werden die historischen Daten des Kontext betrachtet. Da es sich um große Datenmengen handeln kann, muss hier auch ein Löschen von irrelevanten Kontexten erfolgen. Zusätzlich handelt es sich bei dem Kontext eines Benutzers um sehr datenschutzkritische Informationen, welche mit den entsprechenden Maßnahmen vor fremden Zugriff geschützt werden müssen.

Die **Interpretation des Kontext** (engl. Context Interpretation) muss mit Hinblick auf die Relevanz betrachtet werden. Es können zum Beispiel pro Anwendung nur bestimmte Ausschnitte des Kontext relevant sein. Aufgrund der Datenbasis kann dann das Benutzerverhalten prognostiziert werden.

Für eine kontextbewusste Anwendung kann ein Kontextwechsel eine **Adaption an den Kontext** (engl. Context Adaption) bedeuten. Diese Aufgabe liegt sehr nah an der Anwendungsschicht. Wie stark die Anwendung sich an den Kontext anpasst, hängt von der Art und dem Grad der Kontextbewusstheit der Anwendung ab. Beides obliegt der Entscheidung des Entwicklers.

Diese Komponenten bilden ein kontextbewusstes Gesamtsystem. Bei jedem solcher Systeme müssen die einzelnen Komponenten vorhanden sein, um es kontextbewusst zu machen. In Abb. 3 ist das Gesamtsystem schematisch dargestellt.

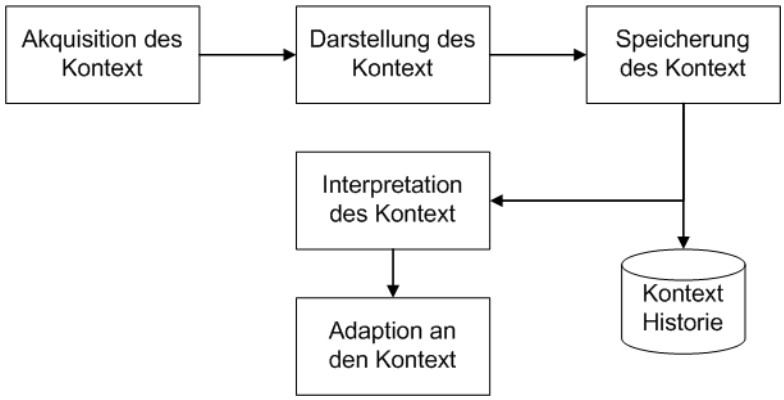

Abb. 3: Kontextbewusstes System nach Malik et al. (2007) (vereinfacht).

In der Vergangenheit wurden kontextbewusste System hauptsächlich als monolithische Software entwickelt. Dabei wurde das System meist für genau einen Zweck geschrieben und die einzelnen Softwarekomponenten sind nur schwierig auf andere Fragestellungen zu übertragen. Um die Besonderheiten bei der Entwicklung für mehrere, kontextbewusste Anwendungen gezielt zu lösen, hat sich eine Tendenz zur kontextbewussten Middleware entwickelt.

Context-aware Middleware

Um eine wiederholte Programmierung der von Malik und Mahmud (2007) genannten Aufgaben zu vermeiden, gibt es bereits mehrere Ansätze für eine kontextbewusste Middleware (engl. Context-aware Middleware). Dabei wird der Kontext umfassend gesammelt und steht für mehrere Anwendungen gleichzeitig zur Verfügung.

Zu den frühesten Versuchen eine Middleware zu erstellen zählt das Context Toolkit von Salber et al. (1999). Es demonstriert mit Teilen des Anwendungsfalls der Active-Badge-Anwendung von Want et al. (1992), dass es für die Erstellung von kontextbewussten Anwendungen genutzt werden kann. Es gilt als Meilenstein in der Forschung, hat sich aber in der Praxis nicht durchgesetzt. Die Abbildung der Kontextverarbeitung erfolgt hierbei mit sog. Context-Widgets, Generatoren, Interpretern und Servern. Hierdurch kann eine vollständige kontextbewusste Anwendung implementiert werden. Ein Mangel ist jedoch das Fehlen von Adaptern für aktuelle mobile Technologien.

Ein weiterer Ansatz ist das Java Context Awareness Framework (JCAF) von Bardram (2005). Es bietet einen Baukasten, um kontextbewusste Anwendungen leichter zu programmieren. Hauptbestandteil ist hierbei die Kommunikation und Veränderung des Kontextes. Es basiert aktuell auf der Java Platform, Standard Edition, Version 1.5 und bietet mehrere, verteilte Komponenten. Leider wird es derzeit nicht länger gewartet und ist seit seiner Veröffentlichung im Jahr 2005 in der Version 1.5.

Weiterhin gibt es andere Ansätze, um speziell für einen Anwendungsbereich eine kontextbewusste Middleware zu schaffen. Zu den bekannteren zählen zum Beispiel die Service-oriented Context-Aware Middleware (SCAM) von Tao et al. (2004), das CAMPUS-Projekt von Hisazumi et al. (2003) oder der Context Management Service (CMS) von da da Silva Santos et al. (2007).

Für die Implementierung und die Schnittstellen einer solchen Software gibt es noch keinen Standard. So weist Ye et al. (2009) darauf hin, dass nur einzelne Lösungen existieren, welche meist nur mit Mehraufwand kompatibel gemacht werden können. Die Notwendigkeit für einen solchen Standard wird steigen, wenn Kontext in der Praxis stärker genutzt wird und sich kontextbewusste Middleware etabliert.

Fazit und Ausblick

Hier wurde ein Einblick in die Grundlagen der kontextbewussten Systeme gegeben. Der Begriff des Kontextes wurde erklärt und auf die Schwachstellen bei den Definitionen hingewiesen. Es wird geraten den Begriff für die kontextbewusste Anwendungsentwicklung einzuschränken.

Bei der Entwicklung einer Anwendung ist zu beachten, dass Kontextbewusstheit nicht das zentrale Ziel einer Anwendung sein sollte. Vielmehr soll sie Hilfsmittel für die Kommunikation sein. Die Praxis hat gezeigt, dass es sinnvoll sein kann, erst eine nicht kontextbewusste Anwendung zu programmieren. Durch Rückmeldung der Benutzer können dann in nächsten Versionen an sinnvollen Stellen kontextbewusste Funktionen hinzugefügt werden.

Die steigende Anzahl an ubiquitären Geräten ermöglicht neue Anwendungsmöglichkeiten. Inwieweit eine umfassende Nutzung der Möglichkeiten geschehen kann, hängt auch stark von den Marktführern der Branche ab. Die Notwendigkeit für eine Middleware zur Auswertung von Kontext steigt mit der Anzahl an Geräten und Sensoren.

Literaturverzeichnis

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

Bonino da Silva Santos LO, van Wijnen RP, Vink P (2007) A service-oriented middleware for context-aware applications. Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing held at the ACM/IFIP/USENIX 8th International Middleware Conference - MPAC '07, S. 37-42. doi: 10.1145/1376866.1376873

Clark S (2009) French retailers, financial services providers and mobile operators publish NFC adoption plans. In: NFC World. http://www.nfcworld.com/2009/10/08/31935/french-retailers-financial-services-providers-and-mobile-operators-publish-nfc-adoption-plans/. Zugriff: 7 Okt 2011.

Dey AK, Abowd GD (2000) Towards a better understanding of context and context-awareness. CHI 2000 workshop on the what, who, where, when, and how of context-awareness. S. 304-307.

Hisazumi K, Nakanishi T, Kitasuka T, et al. (2003) CAMPUS: A Context-Aware Middleware. The 2nd CREST Workshop on Advanced Computing and Communicating Techniques for Wearable Information Playing. Citeseer, S. 2-9.

Lieberman H, Selker T (2000) Out of context: Computer systems that adapt to, and learn from, context. IBM Systems Journal 39:617632.

Malik N, Mahmud U (2007) Future challenges in context-aware computing. proceedings of the IADIS 306-310.

Salber D, Dey AK, Abowd GD (1999) The Context Toolkit: Aiding the Development of Context-Enabled Applications. Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99. ACM Press, Pittsburgh, Pennsylvania, United States, S. 434-441.

Strang T, Linnhoff-popien C (2004) A Context Modeling Survey. Proceedings of UbiComp: 1st International Workshop on Advanced Context Modelling, Reasoning and Management

Tao G, Hung Keng P, Da Qing Z (2004) A middleware for building context-aware mobile services. 2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No.04CH37514). IEEE, S. 2656-2660.

Want R, Hopper A, Falcão V, Gibbons J (1992) The active badge location system. ACM Transactions on Information Systems 10, S. 91-102. doi: 10.1145/128756.128759

Ye C, Cheung SC, Wei J, et al. (2009) A study on the replaceability of context-aware middleware. Proceedings of the First Asia-Pacific Symposium on Internetware - Internetware '09. ACM Press, New York, New York, USA, S. 1-10.

Zimmermann A, Lorenz A, Oppermann R (2007) An operational definition of context. Proceedings of the 6th international and interdisciplinary conference on Modeling and using context. S. 558-571.

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