Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung

Jaroslaw Urich

Context-Aware Systeme: aktuelle Projekte

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jaroslaw Urich

Context-Aware Systeme: aktuelle Projekte

Seminarausarbeitung im Rahmen der Veranstaltung Anwendungen 2
im Studiengang Informatik (Master of Science)
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

bei Prof. Dr. Kai von Luck

Abgegeben am 27. Februar 2008

Jaroslaw Urich

Thema der Seminarausarbeitung

Context-Aware Systeme: aktuelle Projekte

Stichworte

Context, Context-Awareness, Context-Aware Systeme, Context-Aware Services

Zusammenfassung

Diese Ausarbeitung beschäftigt sich mit Context-Aware Systemen. In diesem Kontext werden die wichtigsten Definitionen erläutert und zwei aktuelle Projekte aus diesem Umfeld präsentiert.

Inhaltsverzeichnis

1 Einleitung

Die Entwicklung mobiler Geräte ermöglicht es verschiedene Dienste überall und jederzeit in Anspruch zu nehmen. So ist es heutzutage beispielsweise möglich den Wetter-Bericht oder die Fahrplanauskunft mit dem Mobiltelefon unabhängig von Ort und Zeit zu erfragen.

Die Sensor-Technologie ermöglicht es die anwenderspezifische Informationen zu erfassen und sie für die Lösung bestimmter Aufgaben zu verwenden. So kann z.B. ein PDA, das über einen GPS-Empfänger[1] verfügt, als Navigationsgerät eingesetzt werden.

Die Kombination von Mobilität und Erfassung der anwenderspezifischen Informationen fördert die Entwicklung von Software-Systemen neuer Art. Solche Systeme werden als Context-Aware Systeme bezeichnet. Mit diesen Systemen wird versucht die beste Lösung für einen bestimmten Benutzer (bzw. Benutzergruppe) in der vorgegebenen Situation zu finden.

In den folgenden Unterkapiteln werden die Motivation (1.1) und die Zielsetzung (1.2) dieser Ausarbeitung dargelegt. Das Unterkapitel 1.3 stellt eine kurze Übersicht der gesamten Ausarbeitung dar.

1.1 Motivation

Die meisten heutigen Software-Systeme bieten eine allgemeine Lösung für ein Problem. Es werden zwar weitere Informationen für die Lösungssuche verwendet (wie beispielsweise das Profil des Benutzers), jedoch weicht diese Lösung von der allgemeinen nicht wesentlich ab. Die Ursache dafür ist, dass diese Systeme nicht in der Lage sind, flexibel auf die unerwarteten Änderungen adäquat zu reagieren. Solche Systeme werden üblicherweise für bestimmte Aufgaben entwickelt und deswegen enthalten sie starre Ablaufregeln, die für die Lösung solcher Aufgaben genügen.

[1] Ein Global Positioning System (GPS) ist jedes weltweite satellitengestützte Navigationssystem. Der Begriff GPS wird aber im allgemeinen Sprachgebrauch speziell für das NAVSTAR-GPS des US-Verteidigungsministeriums verwendet, das Ende der 1980er-Jahre zur weltweiten Positionsbestimmung und Zeitmessung entwickelt wurde (Quelle: Wikipedia).

Die Motivation dieser Ausarbeitung besteht in der Konzeption eines Systems, das in der Lage ist, anhand der aktuellen Situation für einen bestimmten Benutzer (bzw. Benutzergruppe) eine passende Lösung zu finden und diese auszuführen.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Darstellung des aktuellen konzeptionellen und technologischen Standes im Bereich der Entwicklung von Context-Aware Systemen. Dadurch soll ein Überblick über die aktuellen Projekte geschaffen werden.

1.3 Gliederung

Im Kapitel 2 „Context und Context-Awareness“ werden die im Unterkapitel 1.1 besprochenen Systeme definiert und näher erläutert. Hierfür werden die benötigten Begriffe erklärt. Kapitel 3 „Aktuelle Projekte“ gibt die aktuellen Forschungsprojekte wider. Im 4. Kapitel „Fazit und Ausblick“ werden diese bewertet und abschließend wird ein Ausblick auf die im nächsten Semester geplante Masterarbeit gegeben.

2 Context und Context-Awareness

Dieses Kapitel beschäftigt sich mit der Definition und den Eigenschaften eines Context-Aware Systems. Bevor auf die Context-Aware Systeme eingegangen wird, sollen zuerst die Begriffe *Context* und *Context-Awareness* geklärt werden.

2.1 Begriffsklärung

2.1.1 Context

In der Literatur sind eine Vielzahl von Definitionen beschrieben, die den Begriff Context zu erläutern versuchen. Die meist zitierte stammt von Anind K. Dey und lautet wie folgt:

Context ist jede Information, die verwendet werden kann um die Situation einer Entity zu charakterisieren. Unter einer Entity wird eine Person, ein Standort oder ein beliebiges Objekt verstanden, das relevant für die Interaktion zwischen dem Benutzer und der Anwendung ist, einschließlich des Benutzers und der Anwendung selbst (Dey, 2001).

Als eine der Hauptaufgaben in diesem Zusammenhang erweist sich die Modellierung von Context. Hierbei müssen folgende Fragen beantwortet werden:

- *Welche Informationen sind für die jeweilige Interaktion relevant?*
  Es muss entschieden werden, welche Daten für das System von Bedeutung sind.
- *Wie können diese Informationen gewonnen werden?*
  Die Erfassung von Informationen ist entscheidend. Hierfür werden unterschiedliche Sensor-Techniken untersucht. Unter „Sensoren“ werden nicht nur physikalische sondern auch Software-Sensoren[1] verstanden.
- *Wie werden diese Informationen zu einem Context zusammengefasst, verwaltet und gespeichert?*
  Diese Frage ist sehr komplex. Sie befasst sich mit der Analyse und der Deutung der

[1] Applikationen oder Programme, die relevante Informationen liefern.

gewonnenen Informationen (vgl. die ersten beiden Fragen). Das Resultat der Analyse ist die Erkenntnis über die aktuelle Situation (auch als *Context* bezeichnet)[2].

Es gibt unterschiedliche Strategien für die Context-Modellierung, deren Betrachtung den Rahmen dieser Ausarbeitung sprengen würde. An dieser Stelle wird lediglich auf die folgende Literatur verwiesen: (PreBur, 2003) für *Activity-Centric Context*, (Aust, 2004) für *Schichtenmodell* und (BuTM, 2005) für *Ontologie-basierte Context-Modelle*.

2.1.2 Context-Awareness

Unter *Context-Awareness* wird die Fähigkeit einer Anwendung (bzw. eines Systems oder eines Dienstes) verstanden, *Context*[3] wahrzunehmen und ihr Verhalten anhand dessen anzupassen[4].
Dadurch kann die Anwendung (bzw. der Dienst) die Erwartung des Anwenders in größerem Maße erfüllen, als wenn die Context-Informationen nicht berücksichtigt worden wären. So kann das System besser auf die Wünsche des Anwenders unter Berücksichtigung der aktuellen Situation eingehen (bzw. diese erfüllen).

Das folgende Unterkapitel stellt die wichtigsten Eigenschaften dieser Systeme dar.

2.2 Context-Aware Systeme

Context-Aware Systeme sind Systeme, die u.a. auch ihre Umgebung bei der Lösung einer Aufgabe in Betracht ziehen.
Da die Umgebung beliebig oft und unvorhersehbar geändert werden kann, muss ein solches System in der Lage sein spontan auf diese Änderungen zu reagieren. Somit muss das System folgende Aufgaben bewältigen können:

- *Erfassung des Context*
  Mit Hilfe von Sensoren werden die Informationen über die Umgebung erfasst. Hierbei soll beachtet werden, dass Sensoren dynamisch hinzugefügt oder entfernt werden können. Unter Sensoren werden sowohl physikalische als auch Software-Sensoren

[2] Ein Beispiel: *Die Sensoren erfassen, dass eine Person sich im Bett befindet und sich kaum bewegt. Es ist 2 Uhr nachts.* Diese Informationen deuten darauf hin, dass diese Person schläft. *'Die Person schläft'* wäre an dieser Stelle das *Context*.
[3] also die Informationen über die Umgebung
[4] vgl. (Dey, 2001) und (DeyAbowd, 2001)

verstanden (siehe oben). Das System soll in der Lage sein, die mit Hilfe von Sensoren gewonnenen Daten richtig zu interpretieren und zu einem *Context* zusammen zu fassen.

- *Reaktionsentscheidung*
  Anhand des ermittelten *Context* soll das System sein Verhalten anpassen. D.h., dass der *Context* analysiert wird und als Ergebnis dessen der nächste Aktions- bzw. Handelsschritt des Systems generiert wird.

- *Ausführung von Aktionen*
  Nachdem die Aktion generiert wurde, muss sie selbstverständlich ausgeführt werden. Hierfür muss das System die zur Verfügung stehenden Ressourcen ermitteln und sie in Anspruch nehmen. Oft sind die Lösungen so komplex, dass die Dienste nicht nur nach einer sondern nach mehreren Ressourcen verlangen. In diesem Fall spielt das System die Rolle eines Dirigenten, der das Zusammenspiel verschiedener Ressourcen einleitet und überwacht.

Context-Awareness ist ein relativ neues Gebiet in der Informatik. Dennoch gibt es bereits eine Vielzahl von Konzeptionen und Implementierungen für diese Systeme. Das nächste Kapitel präsentiert die aktuellen Projekte in diesem Umfeld.

3 Aktuelle Projekte

Dieses Kapitel beschäftigt sich mit den aktuell laufenden oder vor kurzem ausgelaufenen Projekten im Bereich des Context-Awareness.

Es existieren bereits eine Reihe von Forschungsprojekten, in denen Context-Aware Systeme konzipiert und sogar prototypisch implementiert worden sind. Eine Betrachtung aller Projekte würde den Rahmen dieser Ausarbeitung sprängen. Diese Arbeit wird zwei Systeme exemplarisch darstellen.

3.1 SWIRLS

SWIRLS (Supporting Wards with Interactive Resources and Logic-based System) ist eine Anwendung, die zwei Ziele verfolgt[1]:

- Unterstützung der Mitarbeiter eines Krankenhauses in ihrer täglichen Arbeit ohne ihren gewöhnlichen Ablaufsmuster zu unterbrechen, und
- Analyse einer neuartigen Integrationsmiddleware mit Hilfe einer geeigneten Anwendung

SWIRLS unterstützt einerseits die Verwaltung eines Krankenhauses (wie z.B. Verwaltung der Patientenhefte, Laboranfragen und -ergebnisse oder die ToDo-Listen der Ärzte), andererseits bietet es ein Wissensportal und ein Colaborative Workspace an, mit deren Hilfe die Ärzte nach Wissen forschen und gemeinsam Probleme lösen können.

Die Interaktion mit dem System kann mit unterschiedlichen Medien durchgeführt werden, wie z.B. Mobiltelefone, Digital Pens oder auch SmartBoards. Das System sorgt dafür, dass die gleiche Information auf unterschiedlichen Geräten auch auf unterschiedliche Art präsentiert wird. Hiermit wird erreicht, dass erstens die Wahrnehmung der Information vereinfacht

[1] vgl. (SWIRLS)

und zweitens die Präsentation auf einigen Geräten überhaupt ermöglicht wird[2].

Im Folgenden wird die Architektur von SWIRLS kurz erläutert.

3.1.1 Architektur

SWIRLS besteht aus fünf wesentlichen Komponenten: *Application Client*, *Interaction Manager*, *Middleware*, *Ontology Manager* und *Event Manager* (siehe Abbildung 3.1).

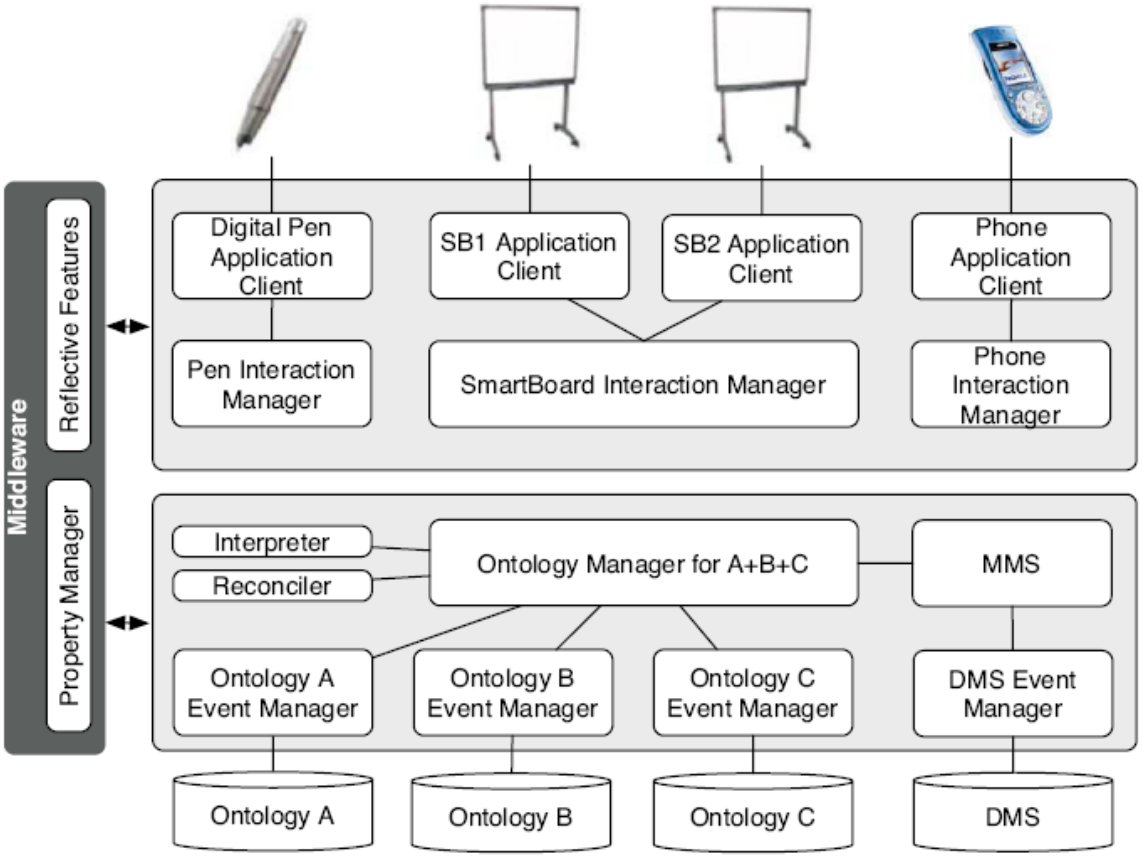

Abbildung 3.1: Architektur von SWIRLS (SWIRLS)

Der *Application Client* ist für die Darstellung von Dokumenten und die Benutzerinteraktion mit dem System zuständig. Diese Komponente präsentiert die Information auf bestimmten

[2]Manche Geräte sind nicht in der Lage Informationen beliebiger Art darzustellen. In diesem Fall muss diese Information auf eine andere Weise dargestellt werden (vergleiche DigitalPen und die als einzig mögliche Textausgabe).

Gerätetypen. So werden Dokumente beispielsweise auf einem Digital Pen anders als auf einem Mobiltelefon dargestellt.

Der *Interaktion Manager* sorgt für die Kommunikation eines bestimmten Gerätetypes mit dem System. Er bereitet die Information so vor, dass diese dann von dem Application Client möglichst effizient und für den Anwender verständlich auf dem gewünschten Gerät präsentiert wird. Hiermit wird erreicht, dass Geräte unterschiedlicher Art mit dem System agieren können.

Die Aufgabe von *Middleware* besteht darin, die Kommunikation zwischen den Applikationen und Diensten herzustellen. Außerdem stellt diese Komponente eine Reihe von Features für die Einpassung unterschiedlicher Dienste (bzw. anderer Komponenten) in das System und für die Wissensverwaltung zur Verfügung.

Die Wissensbasis in SWIRLS besteht aus den verfügbaren Ontologien[3], die von dem *Ontology Manager* verwaltet werden, und den Dokumenten, die in dem *Document Management System* gespeichert sind. Ontologien werden für die Wissensrepräsentation einer bestimmten Domäne verwendet, die durch die Formulierung von Konzepten und die Definition von Relationen zwischen den Termen der Domäne gegeben sind.

Event Managers verwalten die Informationsflüsse der Ressourcen, die nicht vom System erfasst sind, bzw. von externen Resourcen. Mit Hilfe von Event Managers können dieselben Ressourcen den unterschiedlichen Diensten zur Verfügung gestellt werden. Auf diese Weise können Informationen zwischen den einzelnen Systemkomponenten oder sogar zwischen den unterschiedlichen Systemen ausgetauscht werden.

3.1.2 Implementierung

In einem italienischen Krankenhaus wurden mehrere Studien durchgeführt, um die Anforderungen genauerer zu spezifizieren. Anhand der gewonnenen Erkenntnisen wurde die erste Version des Systems implementiert. Für die Realisierung wurden unter anderem folgende

[3] Unter *Ontologie* wird in der Informatik eine Wissensrepräsentation eines formal definierten Systems von Begriffen und Relationen verstanden (vgl. Wikipedia).

Frameworks verwendet: *MMS*[4] für die Informationssuche (bzw. -verwaltung), MAIS[5] für die Unterstützung unterschiedlicher Gerätetypen und DJess[6] für die Implementierung von *Interaktion Manager* (siehe vorheriges Unterkapitel).

Mit SWIRLS wurde eine Umgebung geschaffen, in der die ärztlichen Fachkräfte bei ihrer täglichen Arbeit unterstützt werden. Hierbei wurde versucht, den üblichen Arbeitsablauf der Ärzte nicht wesentlich zu verändern. Weitere Informationen zu SWIRLS finden sich unter (SWIRLS).

3.2 CAMUS

Ein weiteres Projekt, das hier vorgestellt wird, stammt aus der koreanischen Universität *Kyung Hee University*[7]. Es trägt den Namen *CAMUS (Context-aware Middleware for Ubiquitous Computing Systems)*. CAMUS ist eine Middleware, mit deren Hilfe sich Context-Aware Systeme relativ einfach entwickeln lassen.

CAMUS stellt eine Reihe von Werkzeugen zur Verfügung, mit deren Hilfe es möglich ist die Umgebung zu erfassen, die somit gewonnenen Daten zu analysieren und anhand des Analyseergebnisses eine resultierende Aktion auszuführen.

Im Folgenden wird die Architektur von CAMUS genauer erläutert.

3.2.1 Architektur

CAMUS besteht aus vier wesentlichen Teilen: *CAMUS Main Server (CAMUS-MS)*, *Service Agent Manager (SAM)*, *Service Agent (SA)* und dem Kommunikationsframework *Planet* (siehe Abbildung 3.2).

[4] *Metadata Management System (MMS)* ist ein Framework, das im Rahmen eines Projektes namens *MILK* entwickelt wurde. MMS verwaltet die Information (bzw. das Wissen), die (bzw. das) über Dokumente, Menschen, Gruppen und Projekte, die in einer Beziehung miteinander stehen. Die weiteren Informationen können aus (MILK) entnommen werden.

[5] siehe (MAIS) für genauere Informationen

[6] Middleware für Pervasive Computing (vgl. mit (DJess))

[7] Die Homepage des Projektes ist unter der folgenden Adresse http://oslab.khu.ac.kr/camus/ gefunden werden.

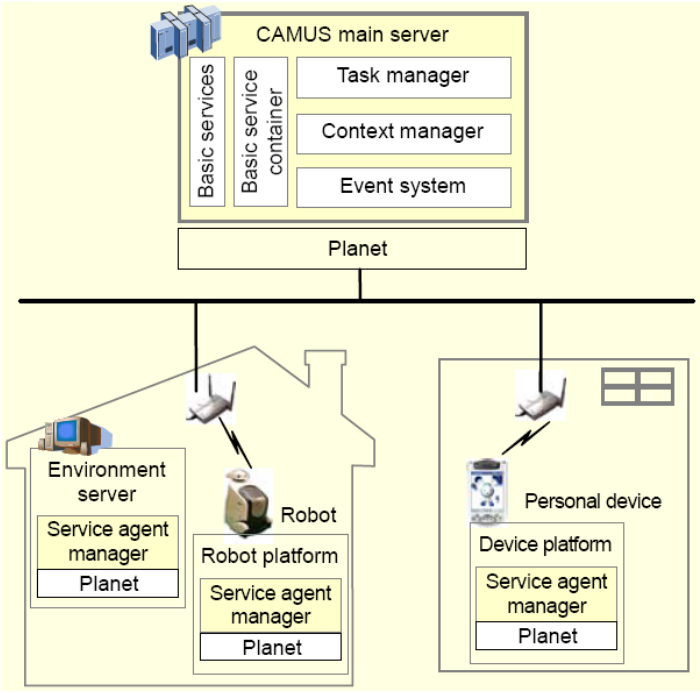

Abbildung 3.2: Systemarchitektur von CAMUS (CAMUS)

CAMUS-MS ist ein Framework, das context-relevante Informationen sammelt und sie zu einem Context zusammenfasst. Das Framework bietet unter anderem eine Reihe von Funktionen für die Entwicklung von Context-Aware Applikationen an. Im Besonderen verwaltet (bzw. kontrolliert) CAMUS-MS alle Informationen über den Benutzer-Context (einschließlich des Context der Umgebung) und die Benutzer-Präferenzen, die für die Handelsempfehlung erforderlich sind. Es sendet Events[8] im Falle einer Context-Änderung an die Applikationen und somit hilft CAMUS-MS den Applikationen eine in der vorgegeben Situation passende Aktion auszuführen.

SAM ist ein Programm, das SAs innerhalb der Umgebung verwaltet und kontrolliert. Hierfür wird SAM in unterschiedlichen Umgebungen innerhalb einer Location installiert. SAM erhält Informationen von unterschiedlichen Sensoren in der Umgebung, sendet diese an CAMUS-MS und kontrolliert die SAs in der Umgebung. SAM kann in einer beliebigen Location installiert werden: in einem Wohnzimmer oder Büro oder auf einem mobilen Gerät wie z.B. PDA.

SAs führen die Anweisungen von CAMUS-MS aus. Ein SA ist ein Proxy für ein Gerät

[8] Events dienen in der Softwaretechnik zur Steuerung des Programmflusses.

eine oder Applikation, welches mit CAMUS-MS interagiert. Mit Hilfe von SAs werden beispielsweise Applikationen oder Sensoren angesprochen (bzw. angetrieben oder ausgeführt).

Planet ist ein Kommunikationsframework eines Netzwerk-basierten Robotersystems[9]. CAMUS-MS kommuniziert mit SAM über Planet. Eine Besonderheit von Planet ist, dass die Nachrichtenkodierung auf der Binärebene stattfindet. Dies ermöglicht einen kompakten[10] und schnellen Nachrichtenaustausch.

Im Folgenden wird die Arbeitsweise von CAMUS näher erläutert. Die Abbildung 3.3 zeigt die konzeptuelle Architektur von CAMUS.

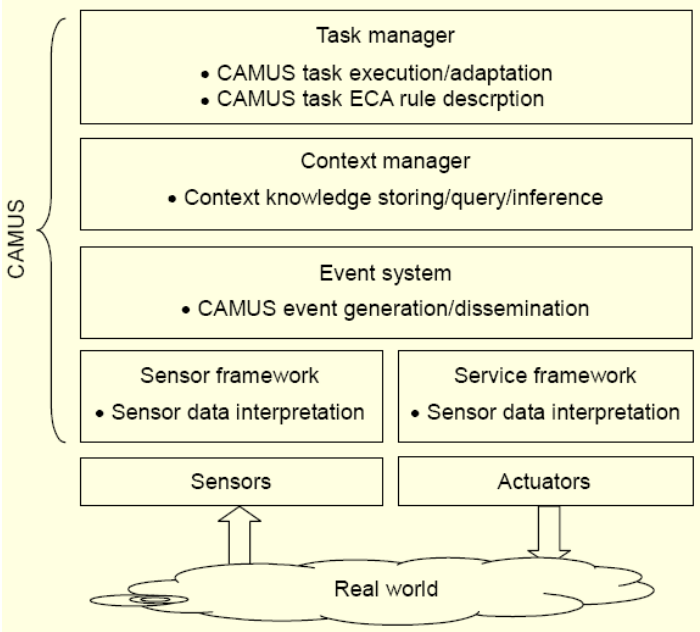

Abbildung 3.3: Konzeptuelle Architektur von CAMUS (CAMUS)

Das Sensor-Framework verwaltet Inputdaten, die aus unterschiedlichen Quellen kommen können. Solche Quellen sind beispielsweise physikalische Sensoren, externe Applikationen oder der Benutzer (d.h. direkte Benutzereingaben). Das Sensor-Framework überträgt diese Informationen an den Context-Manager.

[9] vgl. mit (CAMUS)
[10] Die Nachrichtenzahl wird durch die binäre Kodierung der Nachrichten reduziert.

Das Event-System generiert und verwaltet Events von physikalisch getrennten Umgebungen und ist für den Nachrichtenaustausch zwischen den CAMUS-Komponenten zuständig. Vor allem transportiert es die Events zu dem Context-Manager und dem Task-Manager, so dass die CAMUS-Tasks[11] fähig sind, die Situationsänderung zu erkennen und das existierende Context-Modell zu aktualisieren.

Der Context-Manager verwaltet die kontextabhängige Information, die von dem Sensor-Framework erfasst wurde. Wenn die kontextabhängige Information in der Umgebung geändert wird, schickt der Context-Manager Events an den Event-Manager. Diese Events werden dann an den Task-Manager übermittelt, um die kontextabhängige Information bei der Taskausführung zu berücksichtigen.

Der Task-Manager initiiert individuelle Tasks, verwaltet die eingehenden Task-Prozesse und führt die aktuellen Tasks aus. Außerdem besitzt der Task-Manager eine *Inference Engine*[12], um die Fakten und Regeln eines Task zu analysieren und eine richtige Entscheidung zu treffen.

Das Service-Framework koordiniert verschiedene Sensoren und Applikationen. Es verwaltet den SAM und die SAs und kontrolliert unterschiedliche Endgeräte.

3.2.2 Implementierung

Für die Evaluierung des Frameworks wurde ein *Context-Aware TV Task* implementiert. Es handelt sich um eine Anwendung, die automatisch den Fernseher mit dem präferierten Programm anschaltet, wenn der Benutzer den Raum betritt und den Fernseher ausschaltet, wenn der Benutzer den Raum verlässt. Der Anwender kann außerdem den Fernseher per Sprache ansteuern. Die Abbildung 3.4 zeigt das Zustandsdiagramm dieses Tasks an.

CAMUS unterstützt *PLUE (Programming Language for Ubiquitous Environment)* als Programmiersprache für die Implementierung für Context-Aware Tasks. PLUE ist eine Erweiterung der Programmiersprache Java. Somit können sich Java-Entwickler schnell in diese Sprache einarbeiten.

[11] Unter einem Task ist hier eine bestimmte Anwendung gemeint.

[12] Eine Engine ist ein eigenständiger Teil eines Computerprogramms. Hier ist eine Engine gemeint, die in der Lage ist anhand der gegebenen Informationen Schlussfolgerungen zu ermitteln.

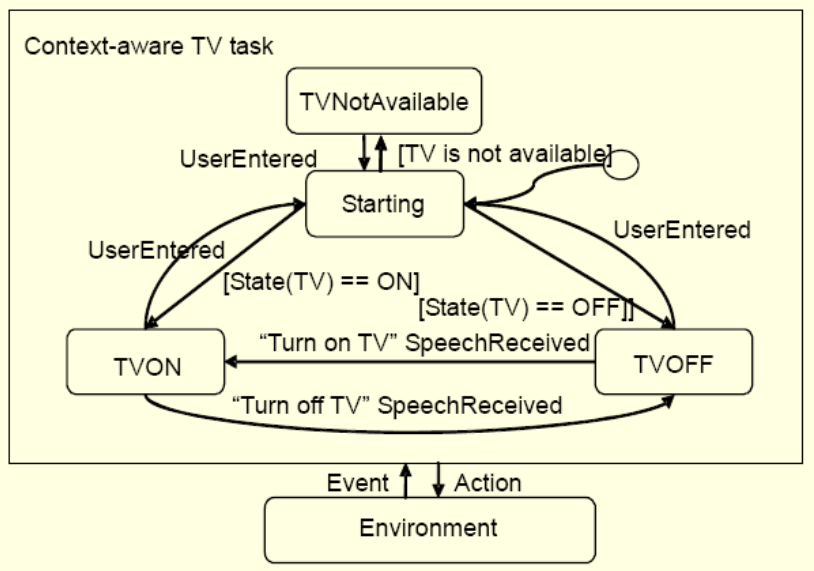

Abbildung 3.4: Zustandsdiagramm von Context-Aware TV Task (CAMUS)

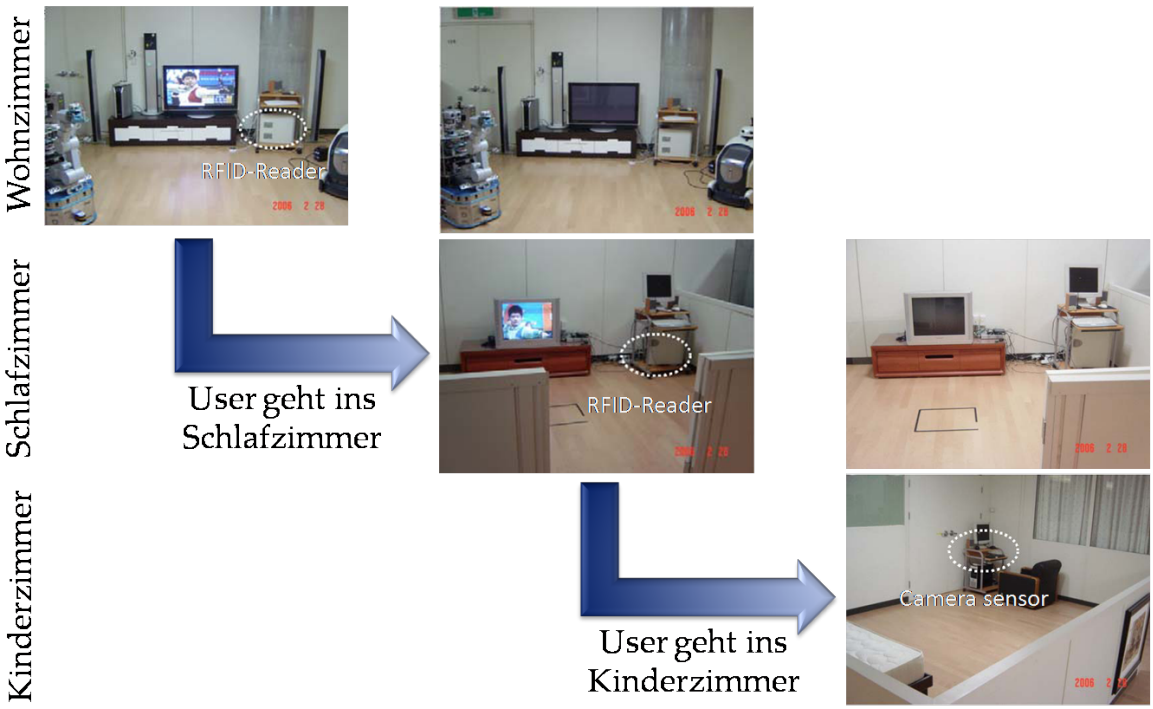

Abbildung 3.5: Context-Aware TV Task: das Experiment

Die Abbildung 3.5 präsentiert das Experiment des implementierten TV-Tasks. Das System wurde in drei Zimmern installiert: Wohnzimmer, Schlafzimmer und Kinderzimmer. Nur das Wohnzimmer und Schlafzimmer haben einen Fernseher. Dementspechend werden nur im Wohnzimmer und im Schlafzimmer die Geräte an- und ausgeschaltet, wenn der Anwender die jeweiligen Zimmern betritt, bzw. sie verlässt (vgl. mit Abbildung 3.5).
Für die Erkennung der Benutzer-Location wurden zwei Arten von Sensoren implementiert. Ein Sensor basiert auf der RFID-Technologie[13], der andere arbeitet mit der Bilderkennung. Sie beide erkennen ob der Anwender sich im jeweiligen Zimmer befindet (siehe Abbildung 3.5).

CAMUS befindet sich noch in der Entwicklung. Jedoch sind bereits die wesentlichen Systemkomponenten implementiert worden. Die weiteren Informationen über CAMUS können auf der folgenden Seite eingesehen werden: *ttp://oslab.khu.ac.kr/camus/*.

[13] RFID (Radio Frequency Identification) ist ein Verfahren zur automatischen Identifizierung von Gegenständen oder Lebewesen.

4 Fazit und Ausblick

In diesem Kapitel werden zunächst die Ergebnisse dieser Ausarbeitung präsentiert (siehe Unterkapitel 4.1). Abschließend wird dann im Unterkapitel 4.2 ein Ausblick auf die anstehende Masterarbeit gegeben.

4.1 Bewertung und Zusammenfassung

Das Ziel dieser Ausarbeitung war, Context-Aware Systeme vorzustellen. Hierbei wurden die wesentlichen Begriffe erläutert und zwei Projekte vorgestellt, die sich mit solchen Systemen auseinandergesetzt haben. Im Rahmen dieser Projekte wurden Frameworks entwickelt, die die Entwicklung von Context-Aware Systemen unterstützen, bzw. vereinfachen.

Das folgende Unterkapitel gibt einen kurzen Ausblick auf die anstehende Masterarbeit.

4.2 Ausblick auf die Masterarbeit

Im Rahmen der anstehenden Masterarbeit soll ein Context-Aware System entwickelt werden. Dieses System soll es ermöglichen Multimedia-Daten (z.B. Audio oder Video) auf unterschiedlichen Geräten darzustellen. Diese Ausgabe-Geräte können zur Laufzeit in das System eingebracht, bzw. aus dem System entfernt werden. Dies soll das System registrieren und auf Änderungen adäquat reagieren. Ein Beispielsszenario für ein derartiges System kann unter (Urich:AW1, 2007) entnommen werden.

Für die Entwicklung des Systems können sich u.a. die Frameworks, die in dieser Ausarbeitung vorgestellt wurden, als nützlich erweisen. In welchem Maße sie wirklich für ein solches System relevant sind, wird ein Aspekt der Untersuchung sein.

Literaturverzeichnis

[Aust 2004] AUSTALLER, G. (Hrsg.): *Web Services als Bausteine für kontextabhängige Anwendungen*. 2004

[BuTM 2005] RENATO BULCÃO NETO, Maria da Graça Campos P. (Hrsg.): *A Semantic Web-Based Infrastructure Supporting Context-Aware Applications*. 2005

[CAMUS] A. MOON, H. K. (Hrsg.) ; LEE, S. (Hrsg.): *Context-Aware Active Services in Ubiquitous Computing Environments*. – URL http://etrij.etri.re.kr/Cyber/servlet/GetFile?fileid=SPF-1175650242476

[Dey 2001] DEY, Anind K. (Hrsg.): *Understanding and Using Context*. 2001. – URL http://www.cc.gatech.edu/fce/ctk/pubs/PeTe5-1.pdf

[DeyAbowd 2001] DEY, Anind K. (Hrsg.) ; ABOWD, Gregory D. (Hrsg.): *Towards a Better Understanding of Context and Context-Awareness*. 2001. – URL http://www.cc.gatech.edu/fce/ctk/pubs/PeTe5-1.pdf

[DJess] F. CABITZA, M. S. (Hrsg.) ; SENO, B. D. (Hrsg.): *DJess - A Context-Sharing Middleware to Deploy Distributed Inference Systems in Pervasive Computing Domains*. – URL http://ieeexplore.ieee.org/iel5/10064/32279/01506416.pdf?tp=&isnumber=&arnumber=1506416

[MAIS] TEAM, The M. (Hrsg.): *MAIS: Multichannel Adaptive Information Systems*. – URL http://ieeexplore.ieee.org/iel5/8885/28063/01254499.pdf?arnumber=1254499

[MILK] : *Multimedia interaction for learning and knowing*. – URL http://www.milkforum.com/,http://klee.siti.disco.unimib.it/milk/

[PreBur 2003] PREKOP, Paul (Hrsg.) ; BURNETT, Mark (Hrsg.): *Activities, Context and Ubiquitous Computing*. 2003. – URL http://arxiv.org/ftp/cs/papers/0209/0209021.pdf

[SWIRLS] R. BOSELLI, F. De P. (Hrsg.) ; LOREGIAN, M. (Hrsg.): *An Adaptive Middleware to Support Context-Aware Knowledge Sharing*. – URL http://ieeexplore.ieee.org/iel5/9817/30953/01437196.pdf

[Urich:AW1 2007] URICH, J. (Hrsg.): *Context-Aware Services: Multimedia-Unterstützung im Flugzeug*. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/ulrich/bericht.pd