Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung

Jaroslaw Urich

Context-Aware Services: Multimedia-Dienste im Flugzeug

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jaroslaw Urich

Context-Aware Services: Multimedia-Dienste im Flugzeug

Seminarausarbeitung im Rahmen der Veranstaltung Ringvorlesung
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 Services: Multimedia-Dienste im Flugzeug

Stichworte

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

Zusammenfassung

Diese Ausarbeitung beschäftigt sich mit der Realisierung eines Context-Aware Systems. In diesem Kontext werden die wichtigsten Definitionen erläutert und ein konzeptueller Entwurf der Systemarchitektur präsentiert.

Inhaltsverzeichnis

1 Einleitung

Mobile Geräte wie PDAs oder Mobiltelefone spielen im Alltag eine immer größere Rolle. Ihre schneller wachsende Leistungsfähigkeit unterstützt diesen Prozess. Immer mehr Menschen[1] benutzen diese Geräte täglich, sei es fürs Telefonieren, Spielen, als Terminkalender oder sogar als Fotokamera.

Mit einigen der heutigen mobilen Geräte ist es bereits möglich übliche Web-Anwendungen zu benutzen, die vielen Anwendern aus der Notebook-Nutzung bekannt sind. Solche mobile Geräte verfügen beispielsweise über einen Webbrowser[2] und der WLAN-Kommunikationsart. Beispiele hierfür sind das iPhone[3] und das HTC P3600[4].

Diese stetig wachsenden technologischen Erneuerungen unterstützen nicht nur eine angenehme Arbeit auf einem solchen mobilen Gerät, sondern bieten außerdem die Möglichkeit einer direkten Kommunikation zwischen den Geräten. So können beispielsweise Musikdateien, Geschäftsdokumente oder andere Daten von einem Gerät auf das andere direkt übertragen werden. Diese Fähigkeit[5] ist nicht nur für den Datenaustausch interessant, sondern sie ermöglicht die Entwicklung von Anwendungen neuer Art. Diese Ausarbeitung beschäftigt sich mit der Entwicklung eines solchen Systems.

1.1 Systemvision

Die Vision, die im Rahmen dieser Ausarbeitung und der Masterarbeit verfolgt wird, wurde bereits ausführlich in (Urich:AW1, 2007) präsentiert. Hier wird eine Zusammenfassung gegeben.

---

[1] vor allem junge Leute

[2] Webbrowser (oder allgemein auch Browser genannt) sind spezielle Computerprogramme zum Betrachten von Webseiten im World Wide Web (Wikipedia).

[3] ein Mobiletelefon von Apple (siehe mehr unter *http://www.apple.com/de/iphone/*)

[4] ein Smartphone von HTC (siehe mehr unter *http://www.europe.htc.com/de/products/htcp3600.html*)

[5] Gemeint ist die direkte Kommunikation zwischen den Geräten.

Die grundlegende Idee besteht darin, ein System zu entwickeln, das in der Lage ist, sein Verhalten an die aktuelle Situation (bzw. an den gegebenen *Context*[6]) anzupassen. Für ein besseres Verständnis eines solchen Systems wird im Folgenden ein mögliches Beispielszenario entworfen.

Ein Fluggast besitzt ein PDA auf dem einige Filme gespeichert sind, die er sich gerne während des Fluges ansehen würde. Der Passagier besteigt das Flugzeug. Das Flugzeug startet. Die Maschine hat die gewünschte Höhe erreicht und jetzt dürfen elektrische Geräte bedient werden. Der Fluggast schaltet sein PDA ein und startet das Abspielen eines gewünschten Films. Vom System wird erkannt, dass der Bildschirm, der sich gegenüber dem Passagier befindet, größer ist als der des PDA. Das System schlägt dem Fluggast vor diesen Bildschirm zu benutzen. Er akzeptiert das und der Film kann jetzt über den Bordbildschirm weitergesehen werden

1.2 Zielsetzung

Das Ziel dieser Arbeit ist ein konzeptueller Entwurf für das Context-Aware System, das im vorherigen Unterkapitel präsentiert wurde. Hierbei sollen die wesentlichen Systemkomponenten festgelegt werden.

1.3 Gliederung

Im Kapitel 2 (Grundlagen) werden Context-Aware Systeme erläutert. Hierbei werden die grundlegenden Begriffe erklärt.
Das nachfolgende Kapitel beschäftigt sich mit dem konzeptuellen Entwurf des Systems. Hier werden die ersten Überlegungen für die Entwicklung präsentiert.
Kapitel 4.2 (Risikoanalyse) schildert die möglichen Risiken bei der Entwicklung des Systems.
Im 5. Kapitel (Ausblick) wird ein Ausblick auf die im nächsten Semester geplante Masterarbeit gegeben.

[6] siehe Kapitel 2

2 Grundlagen

Die Systeme, die im Kapitel 1 vorgestellt wurden, werden in der Informatik im Rahmen von *Context-Awareness* analysiert. Dieses Kapitel definiert und erläutert die 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.

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

- *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 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:

[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)

- *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 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.

2.3 Service-orientierte Architektur

Ein möglicher Grundtyp für Context-Aware Systeme ist die Service-orientierte Architektur[5] (SOA). Diese Architektur soll im Folgenden näher erläutert werden.

SOA ist ein Paradigma für die Strukturierung und Nutzung verteilter Funktionalität die von unterschiedlichen Besitzern verantwortet werden kann[6].

Das Hauptmerkmal dieser Architektur ist, dass das System aus mehreren von einander unabhängigen Diensten besteht (bzw. stellt solche zur Verfügung). Die einzelnen Dienste bewältigen eine kleine aber vollständige Aufgabe. Sie sind los gekoppelt und können für die Lösung komplexerer Aufgaben verwendet werden. Auf diese Weise wird das System relativ einfach erweitert.

Die SOA-Architektur hat demnach folgende Eigenschaften:

[5] engl. *service oriented architecture*
[6] vgl. (OASIS:SOA, 2006)

- *Verteilung:* Vom Benutzer verwendete Programme, Daten und sogar Hardware sind immer seltener am selben Ort und direkt miteinander verbunden. Der Verteilungs- und Vernetzungsgrad nimmt zu.
- *Lose Kopplung:* Auf Grund der sich ständig ändernden Anforderungen an das System ist eine lose Kopplung zwischen den Systemkomponenten (bzw. Diensten) sinnvoll. So kann flexibler auf Änderungen reagiert werden.
- *Standards:* Die Verwendung von Standards unterstützt die flexible Interaktion innerhalb einer SOA. Mit Hilfe von Standards kann das funktionierende Zusammenspiel von einzelnen Systemkomponenten gewärleistet[7].
- *Erschwerte Fehlersuche:* In einem SOA-basierten System, so wie allgemein in einem verteilten System, ist die Fehlersuche durch die Verteilung erschwert. Die Fehler können in den einzelnen Komponenten oder bei der Interaktion zwischen den Komponenten auftreten. Deshalb soll diese Tatsache bei der Entwicklung von solchen Systemen berücksichtigt werden.

Weitere Information zum SOA können unter (OASIS:SOA, 2006) oder (IBM:SOA, 2005) gefunden werden.

[7] Hier ist eine allgemeine Schnittstelle für den Zugriff auf die Komponente (bzw. Beanspruchung von Diensten) gemeint.

3 Konzeptueller Entwurf

Dieses Kapitel stellt die Architekturüberlegungen für das im Kapitel 1 vorgestellte Context-Aware System dar. Als erstes werden die wichtigsten Systemkomponenten ermittelt. Anschließend wird die Kommunikationsart zwischen ihnen bestimmt.

3.1 Ermittlung von Systemkomponenten

Zu den Aufgaben eines Context-Aware System gehören[1]:

- Sammlung für die Verarbeitung relevanter Daten
- Analyse der erfassten Daten
- Ziehung von Schlussfolgerungen aus den Analyseergebnissen
- Ausführung generierter Tasks

Diese Teilaufgaben sind in sich abgeschlossen. Dies spricht dafür, dass sie auch von unterschiedlichen Komponenten bewältigt werden sollen. Dementsprechend könnte die Architektur wie in Abbildung 3.1 aussehen.

Die Komponente *Sensor* (siehe Abbildung 3.1)ist für die Erfassung von systemrelevanten Daten zuständig. Hierfür können sowohl physikalische als auch Software-Sensoren verwendet werden (siehe Unterkapitel 2.1.1). Die erfassten Informationen werden von der Komponente *Context-Erzeugung* analysiert und zu einem Context zusammengefasst. Der gewonnene Context wird von der Komponente *Handelsentscheidung* bei der Lösung von Systemaufgaben berücksichtigt. Diese Komponente generiert Tasks die für die gegebene Situation für passend befunden wird. Die Ausführung der Tasks wird von *Service-Agenten* erledigt. Die einzelnen Service-Agenten werden nicht nur für die Lösung bestimmter Aufgaben eingesetzt, sondern können u.a. im Zusammenspiel mit anderen Agenten größere Aufgaben gemeinsam lösen[2].

[1] vgl. mit Kapitel 2.2 (Context-Aware Systeme)
[2] vgl. SOA aus dem Unterkapitel 2.3

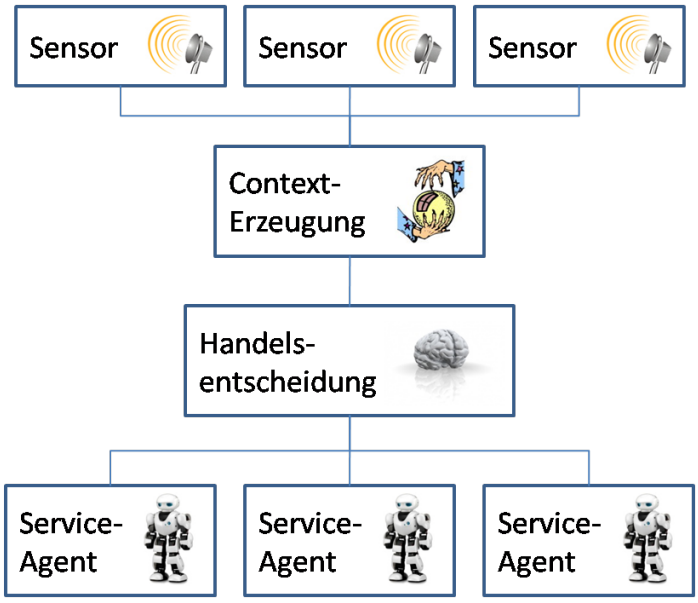

Abbildung 3.1: Architekturentwurf

An dieser Stelle werden die Architekturüberlegungen beendet. Die genaue Modellierung des Systems ist die Aufgabe, die im Rahmen der anstehenden Masterarbeit bewältigt werden soll.

3.2 Kommunikation im System

Dieses Kapitel beschäftigt sich mit der Kommunikation innerhalb des Systems. Hier werden zwei Kommunikationsarten vorgestellt und ihre mögliche Verwendung im System erläutert.

3.2.1 Event-basierte Kommunikation

Bei der Analyse aktueller Projekte, die im Umfeld der Context-Awareness stattgefunden haben, wurde festgestellt, dass die meisten Systeme eine Event-basierte Kommunikationsart verwenden[3].

Bei dieser Kommunikationsart werden Nachrichten mit Hilfe von Events übertragen. Die Besonderheit dieser Kommunikationsart ist, dass die Systemkomponenten nicht direkt

[3] siehe (Urich:AW2, 2008)

miteinander kommunizieren. Die Kommunikation findet über einen sogenannten *Event-Manager* statt. Eine Systemkomponente generiert einen Event mit der Information, die an die anderen Komponenten übertragen werden soll. Der Event-Manager stellt diesen Event an alle Komponenten zu, die sich für diesen Eventtyp bei ihm registriert haben (siehe Abbildung 3.2).

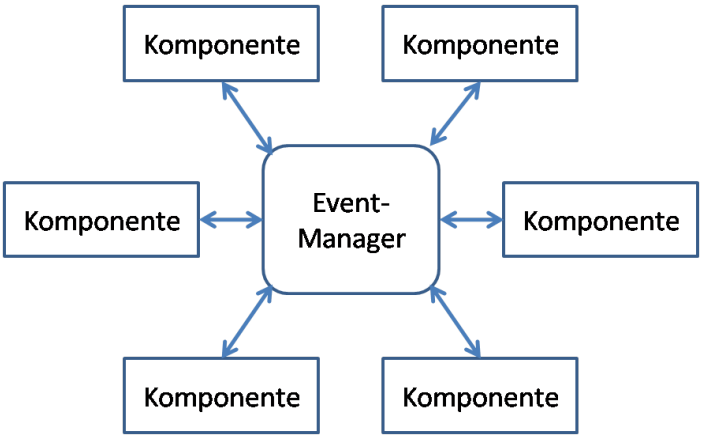

Abbildung 3.2: Event-basierte Kommunikation

Die Event-basierte Kommunikation hat somit folgende Eigenschaften:

- *unbekannter Sender/Empfänger*
  Die Events werden von dem Event-Manager transportiert. Deshalb wenn die Information über den Sender nicht im Event kodiert ist, kann der Empfänger den Sender nicht erkennen. Das Gleiche gilt auch für den Sender. Der Sender kennt den Empfänger nicht direkt. Er erzeugt ein Event und übergibt dieses an den Event-Manager, der dann an die Interessenten weiterleitet. Nur der Event-Manager ist darüber informiert, wer mit wem kommuniziert.
- *mehrere Sender/Empfänger*
  Aufgrund der oben genannten Eigenschaft kann ein Event an mehrere Empfänger zugestell werden, bzw. kann ein Empfänger von mehreren Sendern Events erhalten.
- *Transport von begrenzten Datenmengen*
  Da ein Event nur als Ganzes verschickt wird, kann es nur eine bestimmte Datenmenge übertragen. Diese Menge hängt von den Eigenschaften des jeweiligen Event-Managers ab.

Ein Beispiel für diese Kommunikationsart findet sich bei (iROS, 2003).

3.2.2 Direkte Kommunikation

Eine gewöhnliche Kommunikationsart zwischen den einzelnen Komponenten ist die direkte Kommunikation (siehe Abbildung 3.3). Hier ist die Kommunikation sowohl innerhalb einer Virtuellen Maschine gemeint, als auch PC-übergreifend (vgl. Netzwerke). Diese Kommunikationsart ist durch die folgenden Eigenschaften charakterisiert:

- *Sender und Empfänger sind bekannt*
  Der Sender muss den Empfänger kennen. Es können u.U. auch mehrere Empfänger sein. Der Empfänger kennt dementsprechend den Sender.
- *unbegrenzte Datenmenge*
  Die Datenmenge, die bei dieser Kommunikationsart übertragen werden kann, ist unbegrenzt. Es kann u.a. ein Streaming verwendet werden, um die Daten kontinuierlich als ein Datenstrom zu übertragen.

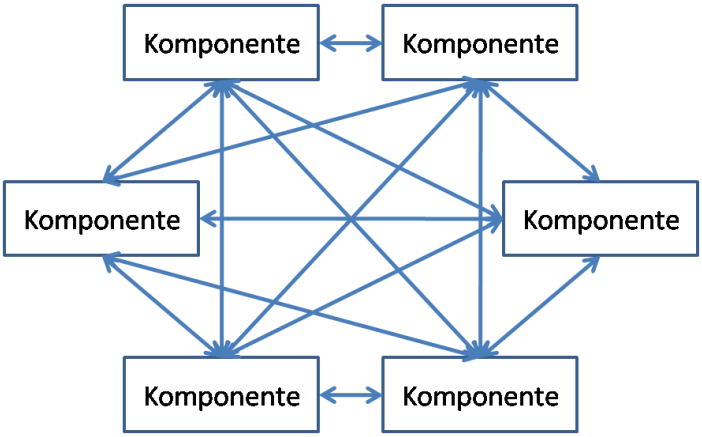

Abbildung 3.3: Direkte Kommunikation

3.2.3 Kommunikation innerhalb des Systems

Beide Kommunikationsarten können für das System von Nutzen sein. Die Event-basierte Kommunikationsart kann beispielsweise für die Benachrichtigung der eingetroffenen Ereignisse oder für den Transport kleinerer Datenmengen, wie z.B. die Daten, die von den Sensoren empfangen wurden oder die Steueranweisungen, verwendet werden.

Die direkte Kommunikationsart kann hingegen für den Transport großer Datenmengen oder für das Streaming eigesetzt werden, wie z.B. bei einer Videoübertragung(vgl. Kapitel

1.1). Eine solche Kommunikation sollte mit Hilfe von Events zustande kommen, um die Flexibilität, bzw. die Unabhängigkeit von bestimmten Komponenten zu gewährleisten[4]. Die Event-basierte Kommunikation dient eher lediglich dem Kommunikationsaufbau.

Die genaue Festlegung der Kommunikation innerhalb des Systems erfolgt in der Masterarbeit.

[4] vgl. mit dem SOA-Ansatz

4 Masterarbeit

Im nächsten Semester schreibt der Autor dieser Arbeit seine Masterarbeit, im Rahmen deren die oben vorgestellte Vision realisiert werden soll. In diesem Kapitel werden das Vorhaben und die möglichen Risiken näher erläutert.

4.1 Vorhaben

Im Rahmen der Masterarbeit soll ein Context-Aware System entwickelt werden, das in dieser Ausarbeitung vorgestellt wurde. Ein konzeptueller Entwurf der Systemarchitektur wurde bereits in den vorherigen Kapiteln vorgestellt. Das Ziel der Masterarbeit liegt darin, diesen Weg weiter zu verfolgen: die Architektur zu entwickeln und ein Prototyp des Systems zu implementieren.

Nach einer ausführlichen Anforderungsanalyse soll eine präzise Systemspezifikation erstellt werden. Anhand der Anforderungen soll eine genaue Systemarchitektur entwickelt werden. Die Implementierung eines Prototyps soll eine Grundlage für die Evaluierung des Systems schaffen.

Für die Implementierung des Systems soll ein Framework verwendet werden, das im Rahmen von Anwendungen 2 präsentiert wurde (CAMUS, siehe (Urich:AW1, 2007)). Das Framework deckt die gewünschte Funktionalität nicht komplett ab[1]. Aus diesem Grund soll das Framework erweitert werden. Anschließend soll das entwickelte System getestet werden. Hierbei soll nicht nur die Funktionalität des Systems geprüft werden, sondern auch dessen Benutzbarkeit.

[1] Das Frame bietet nur eine Event-basierte Kommunikationsart zwischen den einzelnen Systemkomponenten. Für die Multimedia-Anwendungen ist aber Streaming erforderlich (vgl. Kapitel 3.2).

4.2 Risikoanalyse

In diesem Kapitel werden verschiedene Risiken identifiziert und bewertet, die die in dieser Ausarbeitung dargelegte Konzeption gefährden könnten.

- *Umfang der Systemvision*
  Die in Kapitel 1.1 dargelegte Systemvision kann unter Umständen nicht im gewünschten Maße umgesetzt werden. In diesem Fall wird die Implementierung des Systems nur auf die vorgestellte Anwendung reduziert[2].
- *Technische Infrastruktur*
  Für die Realisierung des Systems werden unterschiedliche Geräte verwendet, u.a. mobile Geräte. Der Kommunikationsaufbau zwischen ihnen kann nicht wie erhofft verlaufen. In diesem Fall sollte auf die schnurlose Kommunikation verzichtet und auf die LAN-basierte Kommunikation zurückgegriffen werden.
- *Technologie*
  Das im letzten Unterkapitel vorgestellte Framework (bzw. andere verwendete Frameworks) kann sich während der Implementierung als nicht geeignet für das System erweisen. In diesem Fall sollte nach einem geeigneten Framework gesucht werden oder im schlimmsten Falle wird das System eigenständig implementiert.

Die Betrachtung der möglichen Risiken wird in den weiteren Arbeiten fortgesetzt und gegebenfalls aktualisiert.

[2] Die Systemvision sieht unterschiedliche Anwendungsszenarien vor. In (Urich:AW1, 2007) wurde nur eins vorgestellt.

5 Fazit und Ausblick

Das Ziel dieser Ausarbeitung war, anhand eines Beispielszenarios, das im Rahmen von Anwendungen 1 präsentiert wurde (siehe (Urich:AW1, 2007)), die wichtigsten Aspekte eines Context-Aware Systems zu identifizieren. Hierbei wurden die wesentlichen Begriffe in diesem Kontext erläutert und ein konzeptueller Entwurf der Systemarchitektur vorgestellt.

Im Rahmen der anstehenden Masterarbeit wird die hier vorgestellte Systemvision (siehe Kapitel 1.1) genauer spezifiziert und anschliessend prototypisch umgesetzt. Die grundlegende Vorgehensweise und der Inhalt der Arbeit werden sich dabei an der in Kapitel 3 dargelegten Konzeption orientieren und diese noch wesentlich verfeinern.

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

[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

[IBM:SOA 2005] IBM (Hrsg.): *Service-Oriented Architecture*. 2005. – URL http://www.research.ibm.com/journal/sj44-4.html

[iROS 2003] S. R. PONNEKANTI, E. K. (Hrsg.) ; FOX, A. (Hrsg.): *Portability, Extensibility and Robustness in iROS*. 2003. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-aw/urich/bericht.pdf

[OASIS:SOA 2006] C. M. MACKENZIE, F. McCabe P. F. Brown R. M. (Hrsg.) ; HAMILTON, B. A. (Hrsg.): *Reference Model for Service Oriented Architecture 1.0*. 2006. – URL http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

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

[Urich:AW2 2008] URICH, J. (Hrsg.): *Context-Awareness: aktuelle Projekte*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-aw/urich/bericht.pdf