Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Jaroslaw Urich

smart:shelf

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jaroslaw Urich

smart:shelf

Projektbericht im Rahmen der Veranstaltung Projekt
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

Projektname

smart:shelf

Stichworte

Lokalisierung der Gegenstände, RFID-Tag, RFID-Reader, Event-Manager, Context-Awareness

Zusammenfassung

Diese Ausarbeitung stellt das Projekt *smart:shelf* vor. Im Rahmen des Projektes wurde ein System entwickelt, das bei der Positionsbestimmung von Gegenständen hilft. In dieser Ausarbeitung werden die Architektur und die Implementierung des Systems präsentiert.

Inhaltsverzeichnis

1 Einleitung

Die Entwicklung in der Elektronik bringt immer mehr neuartige elektronische Geräte hervor. Neue funktionsreichen Software-Systeme sind das Ergebnis des Fortschritts in der Informatik. Immer mehr Anwender nutzen diese technologischen Eroberungen.
Die Bedienung der einzelnen Geräte (bzw. Software-Systeme) ist für die meisten Anwender keine besondere Herausforderung. Wenn jedoch die Benutzung mehrerer Geräte (bzw. die Anwendung verschiedener Software-Programme) für eine bestimmte Aufgabe gewünscht wird, wird der Einsatz kompliziert. Eine Abhilfe (bzw. ein vollständig automatisiertes Zusammenspiel der Geräte bzw. Software-Systeme) ist hier erwünscht.

Dieses Problem wird im Rahmen von *Context-Awareness* behandelt. Ein Context-Aware System zeichnet sich dadurch aus, dass es in der Lage ist seine Umgebung[1] wahrzunehmen und der gegebenen Situation entsprechend zu reagieren (DeyAbowd, 2001). Eine weitere Eigenschaft des Systems ist die Fähigkeit die vorhandenen Dienste (bzw. Ressourcen) aufsuchen zu können um diese dann in Anspruch zu nehmen.

So können diese Systeme unterschiedliche Geräte ansteuern, bzw. sich andere Software-Systeme zunutze machen. Auf diese Weise kann der Anwender auf eine einfache Weise Geräte steuern (wie z.B. per Sprache) oder die Steuerung der Geräte wird sogar vom System automatisch durchgeführt[2].

1.1 Motivation

Eine der Hauptaufgaben eines Context-Aware Systems ist die Umgebungsanalyse. Dies kann mit Hilfe von Sensoren erfolgen. So kann beispielsweise die Anwesenheit des Benutzers, der einen RFID-basierten Ausweis bei sich trägt, mittels eines RFID-Readers registriert werden.

[1] einschließlich der Benutzer-Anweisungen

[2] Das System untersucht beispielsweise die Umgebung und das Verhalten des Benutzers. Anhand dieser Analyse werden die Geräte (bzw. Software-Programme) entsprechend in Anspruch genommen.

Weitere Eigenschaften eines Context-Aware Systems sind das Service Discovery und die Möglichkeit einer relativ einfachen Systemerweiterung. So kann beispielsweise das System ein Fernsehgerät automatisch erkennen und dieses ansteuern. Eine Erweiterung des Gesamtsystems um eine Stereoanlage geschieht ohne aufwendige Konfigurationen.

Die Motivation dieser Ausarbeitung lag darin, die hier genannten Eigenschaften eines Context-Aware Systems in der Praxis zu erproben. Hierfür wurde ein Projekt namens *smart:shelf* durchgeführt. Im Weiteren werden das Projekt und dessen Ergebnisse präsentiert.

1.2 Gliederung

Kapitel 2 stellt das Projekt *smart:shelf* vor. Hierbei werden die Zielsetzung und die Systemanforderungen erörtert, die sich im Kapitel 3 in der Systemarchtektur widerspiegeln. Kapitel 4 beschäftigt sich mit der Implementierung. Hier werden die wichtigsten Vorgehensweisen und verwendeten Technologien erläutert. Mit dem Kapitel 5 wird diese Ausarbeitung abgeschlossen. Dieses Kapitel präsentiert die Ergebnisse des Projektes und gibt einen Ausblick auf die bevorstehende Masterarbeit.

2 smart:shelf

2.1 Systemvision

smart:shelf[1] ist ein System, das das Orten von Gegenständen ermöglicht. Es besteht aus Regalen, in denen sich diese Gegenstände befinden. Diese Regale verfügen über einen sogenannten *RFID-Reader*[2], der die *RFID-Tags*[3] der zugehörigen Gegenstände auslesen und somit identifizieren kann.

Um etwas mehr Komfort anzubieten, bietet das System eine Suche nach Gegenständen an. Mit dieser Suchfunktionalität kann der Anwender die Gegenstände nach ihren Eigenschaften suchen und lokalisieren. So können beispielsweise Bücher nach ihrem Autor oder Genre gefunden werden. Das System liefert in diesem Fall nicht nur die bestimmten Titeln oder ISBN-Nummern der Bücher, sondern weist auf u.a. den Bestimmungsort der Bücher hin (wie das Regal und das Fach).

Das System ist relativ einfach erweiterbar. So kann beispielsweise ein neues Regal mit einem minimalen Aufwand (bzw. völlig automatisch) integriert werden. Eine neuartige Systemerweiterung wie z.B. die Suche nach Gegenständen per Sprache ist nicht auszuschließen.

2.2 Anforderungsanalyse

Anhand der oben beschriebenen Systemvision wird hier die Anforderung an das System erläutert.
Das System soll so konzipiert werden, dass es die folgenden Eigenschaften aufweist:

---

[1] siehe auch verwandte Arbeiten unter (Hollatz, 2008) und (Meißner, 2008)

[2] *Radio Frequency Identification (RFID)* ist ein Verfahren zur automatischen Identifizierung von Gegenständen und Lebewesen (Wikipedia).

[3] Ein *RFID-Tag* ist ein Objekt, das in ein Produkt, ein Tier oder sogar in eine Person integriert werden kann, für den Zweck der Identifizierung mit Hilfe von Radiowellen.

- *Identifikation der Gegenstände im Regal mittels RFID-Technologie*
  Die RFID-Tags werden immer kleiner und billiger. Einige von denen sind bereits in der Größe einer handelsüblichen Postmarke und können somit relativ einfach an die Gegenstände angebracht werden. Der Kostenfaktor spielt hier eine nicht weniger wichtige Rolle. Die einzelnen Tags dürfen nur wenige Cents kosten, damit sie an jedem Gegenstand angebracht werden können. Ein höherer Preis für diese RFID-Tags würde die Benutzer hindern sie in Massen zu kaufen. Nur dann macht es Sinn das System zu verwenden, wenn alle Gegenstände mit diesen Tags versehen worden sind.
- *einfache Installation der Regale (bzw. ihre Integration in das System)*
  Ein Benutzer wird nur dann ein System nutzen, wenn die Bedienung einfach und verständlich ist.
- *Suche nach Gegenständen anhand deren Eigenschaften*
  Häufig sucht man z.B. eine CD mit einem bestimmten Lied sucht. Das System finden eine oder mehrere CDs mit dem gewünschten Lied. Der Benutzer kann dann entscheiden, welche er gesucht hat. Das System verrät ihm die Position dieser CD.
- *einfache Erweiterung des Systems*
  Das System kann relativ einfach auf die weiteren Regale erweitert werden. Das System ist in der Lage die neuen Regale zu registrieren. Die Erweiterung des Gesamtsystems auf die neue Funktionalität ist u.a. möglich. So kann beispielsweise das System auf eine sprachgesteuerte Suchfunktionalität erweitert werden[4].

Abbildung 2.1 stellt die Funktionalität des Systems aus der Benutzersicht graphisch dar.

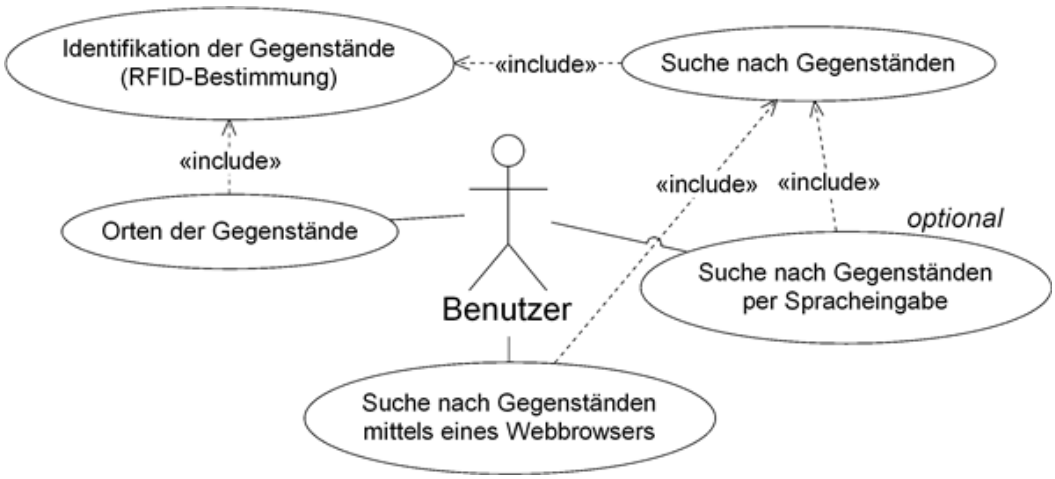

Abbildung 2.1: Anwendungsfalldiagramm für die Suche und das Orten von Gegenständen

[4]vgl. auch mit den Eigenschaften eines Context-Aware Systems aus Kapitel 1.1

3 Konzeption

Dieses Kapitel beschäftigt sich mit der Konzeption von smart:shelf. Hier wird die Architektur des Systems präsentiert und die Arbeitsweisen der einzelnen Systemkomponenten erläutert.

3.1 Systemarchitektur von smart:shelf

Die Abbildung 3.1 zeigt die Architektur von smart:shelf. Das gesamte System besteht aus vier Hauptkomponenten: *Event-Manager*, *Shelf*, *ShelfDB* und *WebClient*.

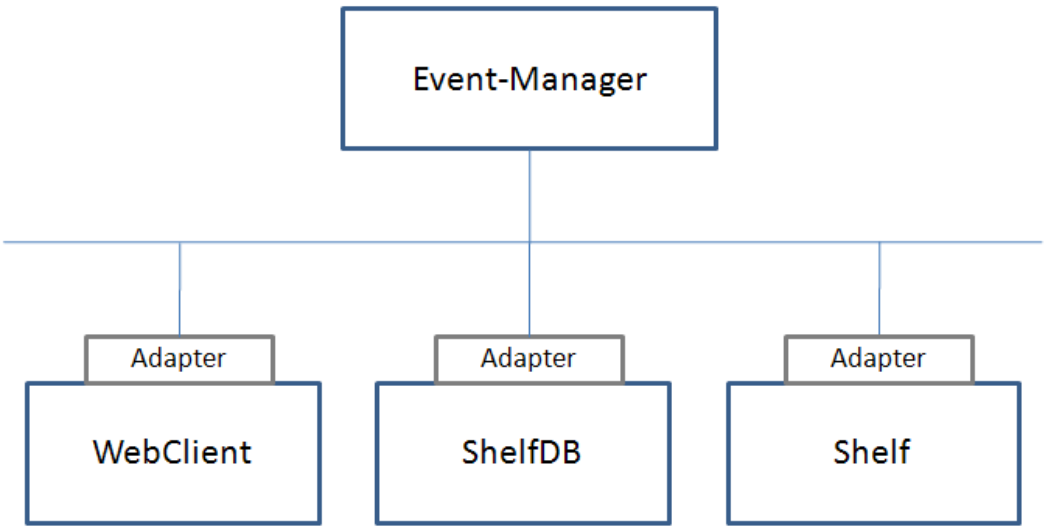

Abbildung 3.1: System-Architektur von smart:shelf

Der Event-Manager ist für die Kommunikation zwischen den Systemkomponenten zuständig. Seine Aufgabe besteht darin, die Nachrichten von einer Komponente auf die andere zu übertragen. Der Nachrichtentransport wird mittels einer Event-basierten Kommunikation realisiert[1].

[1] siehe (Urich:Ring, 2008)

Bei dieser Kommunikationsart kennt der Sender die Empfänger nicht, bzw. nicht direkt. Der Sender schickt eine Nachricht, die in ein Event verpackt und an den Event-Manager übergeben wird. Anhand des Event-Typs stellt der Event-Manager diese Nachricht nur an die Komponente zu, die sich zuvor für diesen Event-Typ registriert haben. Der Vorteil bei dieser Kommunikationsart ist, dass weitere Komponente relativ einfach in das System integriert werden können. Sie müssen sich lediglich für die Events bestimmter Art registrieren und sofort können sie die Nachrichten empfangen.
Alle Systemkomponenten, die in das System integriert sind, besitzen einen Adapter, der die verpackten Nachrichten an den Event-Manager weiterleiten.

Shelf ist eine Komponente, die die Position der Gegenstände ermittelt. Sie hat einen RFID-Reader, mit dessen Hilfe die Ermittlung stattfindet. Der RFID-Reader stellt fest, welche Gegenstände sich in der Nähe befinden. Hierfür liest er alle erreichbaren RFID-Tags, die an die Gegenstände angebracht wurden. Somit kann der gesamte Bestand von den sich in der Nähe befindlichen Gegenständen erfasst werden.
Die Komponente Shelf ist in einem Regal installiert um den Inhalt des Regales erfassen zu können.

ShelfDB hält die Informationen über die einzelnen Artikel. Die verwendeten RFID-Tags haben eine nicht ausreichende Speicherkapazität um die Informationen für den entsprechenden Gegenstand zu speichern. Die Eigenschaften der einzelnen Gegenstände werden in einer Datenbank gesammelt. Somit können die Gegenstände (bzw. ihre RFID-Nummer) anhand deren Eigenschaften gefunden werden. Dies ermöglicht eine etwas bequemere Suche nach Gegenständen, die sich in im Regal befinden.

WebClient ist eine Webanwendung. Sie bietet ein graphisches Interface für die Interaktion mit dem Anwender an. So kann der Benutzer mit Hilfe eines Webbrowsers nach Gegenständen suchen, bzw. ihre aktuelle Position ermitteln.

3.2 Konzeptuelle Architektur von smart:shelf

Abbildung 3.2 zeigt die konzeptuelle Architektur von smart:shelf.

Die End-Komponenten bieten Dienste an, mit deren Hilfe unterschiedliche Aufgaben ausgeführt werden können. Jede End-Komponente besitzt einen Adapter, der für die Registrierung beim Event-Manager und für die Event-Erzeugung zuständig ist. Der Event-Manager verwaltet die End-Komponenten und stellt die Events zu (vgl. Abbildung 3.2).

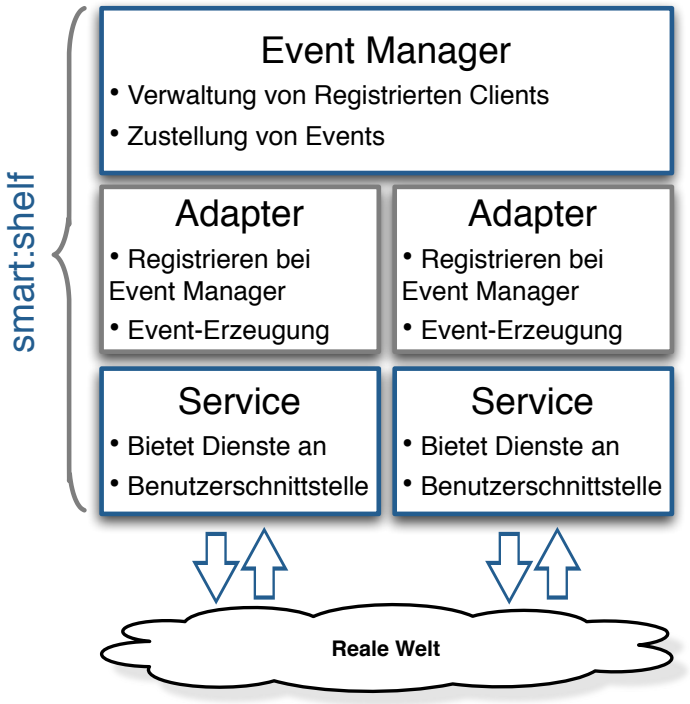

Abbildung 3.2: konzeptuelle Architektur von smart:shelf

Die Arbeitsweise des Systems wird anhand eines Beispiels erläutert. Das Beispiel zeigt eine Suche nach Gegenständen.

Beim Starten des Systems registrieren sich der WebClient und die ShelfDB bei dem Event-Manager. Die ShelfDB registriert sich für Such-Anfragen und der WebClient für Such-Ergebnisse. Der Benutzer startet einen Webbrowser mit der Webanwendung. Er gibt in die Suchmaske ein, dass er nach einer CD der Gruppe „The Doors“ sucht.

Diese Anfrage wird von dem Adapter in ein Such-Event verpackt und an den Event-Manager weiter übergeben. Der Event-Manager stellt diese Anfrage an die ShelfDB zu, da diese Komponente sich für diese Art von Events bereits registriert hat. Die ShelfDB empfängt diese Abfrage, führt sie aus und generiert eine Antwort-Nachricht, die wiederum von dem Adapter in ein Event verpackt und an den Event-Manager überreicht wird. Damit der Empfänger erkennen kann, dass diese Nachricht die Antwort auf seine Anfrage ist[2], werden das Anfrage- und das Entwort-Event mit einer eindeutigen ID versehen. Der WebClient empfängt diese Antwort und stellt die gefundenen CDs graphisch dar.

[2] Es können unterschiedliche WebClients existieren, die sich für die gleichen Event-Typs registrieren. In diesem Fall wird die Antwort von allen WebClients empfangen.

4 Realisierung

Dieses Kapitel beschäftigt sich mi der Implementierung von smart:shelf. Hier werden die wichtigsten verwendeten Technologien genannt.

Das System wurde in der Programmiersprache Java 1.5 realisiert. Für diese Technologieentscheidung sprachen mehrere Gründe. Wichtig war, dass Java plattformübergreifend ist. So können beispielsweise Java-Anwendungen sowohl unter Windows als auch unter Unix laufen. Ein weiterer Grund für den Java-Einsatz war die relativ große Auswahl an Frameworks, die bei der Realisierung arbeitserleichternd wirken.

Im Folgenden wird die Implementierung der einzelnen Komponenten näher erläutert.

4.1 Event-Manager

Für den Event-Manager wurde das Framework *iROS* verwendet, das an der Stanford University entwickelt wurde. *iROS (Interactive Room Operating System)* ist eine Middleware, die den Komponenten der Raumumgebung eine Plattform zur Kommunikation untereinander anbietet[1].

Eine der Hauptkomponenten von iROS stellt der *Event Heap* dar. Er ist zugleich die zentrale Komponente, über den die Kommunikation zwischen den einzelnen Komponenten stattfindet. Der Event Heap ist in Java implementiert und kann somit relativ einfach in Java-basierte Systeme integriert werden.

In smart:shelf wird der Event Heap als Event-Manager verwendet (siehe vorheriges Kapitel). Der Adapter, den jede End-Komponente besitzt, stellt eine Schnittstelle zu dem Event-Manager und ist ebenfalls in Java realisiert.

[1] vgl. (iROS, 2003)

4.2 Shelf

Für die Implementierung der Komponente Shelf wurde ein RFID-Reader von der Firma NXP verwendet. Der Zugriff auf die Hardware erfolgt über ein in C geschriebenes Tool. Dieses Tool wurde von NXP bereit gestellt. Da diese Komponente ebenfalls[2] in Java realisiert ist, wird das Tool mittels eines speziell dafür entwickelten JNI-Wrappers[3] angesprochen.

4.3 ShelfDB

Die Komponente ShelfDB verwaltet die Informationen von Gegenständen. Diese Informationen werden in einer Datenbank gehalten. Für die Realisierung wurde eine MySQL[4] Datenbank verwendet.

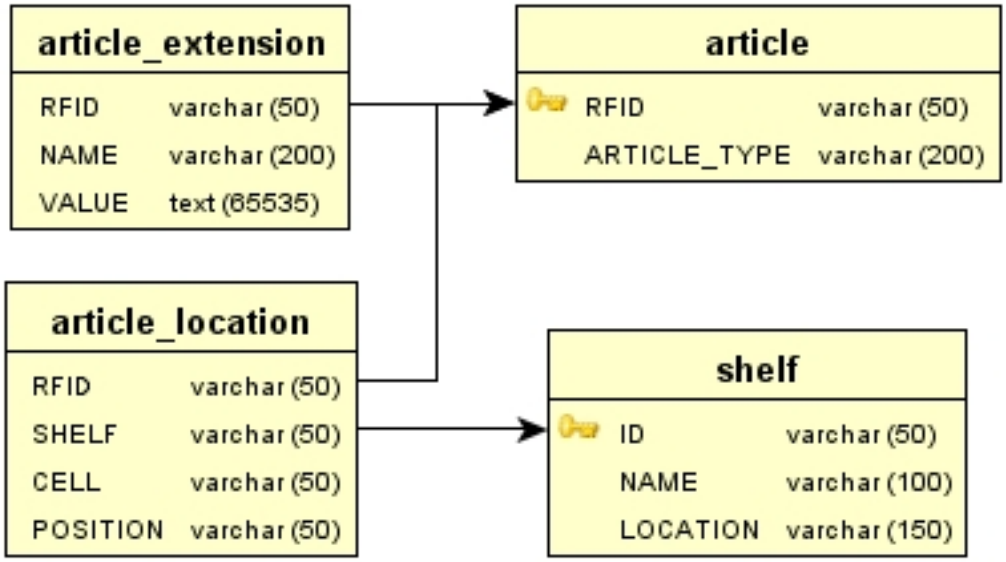

Abbildung 4.1: Datenbank-Schema von smart:shelf

[2] wie das gesamte System

[3] *Java Native Interface (JNI)* ist eine standardisierte Anwendungsprogrammierschnittstelle (API) zum Aufruf plattformspezifischer Funktionen bzw. Methoden aus der betriebssystemunabhängigen Programmiersprache Java heraus (Wikipedia). So kann beispielsweise ein in C geschriebenes Programm von einem Java-Programm verwendet werden.

[4] Die Informationen zu dieser Datenbank können unter (MySQL, 2007) entnommen werden.

Abbildung 4.1 stellt das verwendete Datenbankschema dar. Der Vorteil dieses Schemas ist, dass beliebige Gegenstände in der Datenbank gespeichert werden können. In der Tabelle *ARTICLE_EXTENSION* können alle möglichen Eigenschaften gehalten werden, da u.a. die Information über die Eigenschaft selbst ebenfalls in der Tabelle gespeichert (vgl. Spalten *NAME* und *VALUE*).

Für den Zugriff auf die Datenbank wurde das Framework *Hibernate* verwendet. Hibernate ist ein Open-Source-Persistenz-Framework für Java. Mit Hilfe dieses Frameworks werden Java-Objekte in die Datenbank geschrieben und aus der Datenbank wieder gewonnen[5].

4.4 WebClient

Die Komponente WebClient ist eine Webanwendung. Somit wird ein Web-Server verwendet, auf dem diese Anwendung läuft. Für diese Komponente wurde *Jetty*[6] als Server verwendet. Die Vorteile von Jetty sind seine relativ gute Performance und die Möglichkeit den Server in eine Java-Anwendung einzubetten.

Das User-Interface wurde mit Hilfe von *Wicket* realisiert. Wicket ist ein Java-Web-Framework, das eine Entwicklung der Java-basierten Webanwendungen unterstützt. Die Eigenschaften und zugleich die Vorteile des Frameworks sind die folgenden:

- *Einfache Verwendung*
  Um das Framework nutzen zu können sind lediglich Java- und HTML-Kenntnisse notwendig. Keine zusätzliche Konfigurations-Dateien sind notwendig (wie z.B. dies bei *Struts*[7] der Fall ist).
- *Wiederverwendbarkeit der Komponenten*
  Die Komponenten, die der Entwickler implementiert hat, können wieder verwendet werden, so wie einfache Java-Objekte. So kann ein bereits implementiertes Eingabe-Formular in andere HTML-Seiten benutzt werden.
- *Ajax-Unterstützung*
  Mit der Einführung des Web 2.0-Paradigmas ist es heutzutage notwendig in der Lage

[5] Weitere Informationen können unter (Hibernate, 2007) gefunden werden
[6] Informationen über diesen Webserver können unter (Jetty, 2007) gefunden werden.
[7] Struts ist ein Open-Source-Framework für die Präsentations- und Steuerungsschicht von Java-Webanwendungen (Wikipedia).

zu sein, Webanwendungen basierend auf der Ajax-Technologie[8] zu entwickeln. Mit dieser Unterstützung können dynamische Webanwendungen realisiert werden.

Weitere Informationen zu diesem Framework können unter (WICKET, 2008) entnommen werden.

4.5 Implementierungsstand

Die Einarbeitung in die neuen Technologien hat die Entwicklung des Systems in der ersten Phase verzögert. Aufgrund der verspäteten Lieferung konnten die RFID-Reader erst in der zweiten Hälfte des Projektes eingesetzt werden. Trotz aller Schwierigkeiten kann das Projekt als erfolgreich bezeichnet werden.

Die Hauptfunktionalitäten der einzelnen Systemkomponenten wurden vollständig implementiert. Die ersten Tests des Gesamtsystems zeigten positive Ergebnisse. So konnte beispielsweise ein bestimmtes Buch anhand des Titel gefunden und seine Position ermittelt werden. Das System hat sich dabei u.a. als relativ performant erwiesen[9].

[8] Ajax ist ein Apronym für die Wortfolge *Asynchronous JavaScript and XML*. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen (Wikipedia).

[9] Der oben beschriebene Vorgang dauerte eine bis wenige Sekunden.

5 Fazit und Ausblick

In dieser Ausarbeitung wurde ein System namens smart:shelf vorgestellt, das im Rahmen der Veranstaltung *Projekt* entwickelt wurde. Das Ziel dieses Projektes bestand darin die Infrastruktur, die Kommunikation zwischen den einzelnen Systemkomponenten und das Service Discovery in einem Context-Aware System in der Praxis zu erproben[1].
Hierbei wurde eine Infrastruktur erschaffen, die die Realisierung eines Context-Aware Systems unterstützt bzw. vereinfacht. smart:shelf besteht aus mehreren Systemkomponenten, die auch für andere Systeme interessant sein könnten[2].

Im Rahmen der anstehenden Masterarbeit soll ein Context-Aware System entwickelt werden, das in (Urich:AW1, 2007) beschrieben wurde. Mit der Realisierung von smart:shelf wurden Erkenntnisse gewonnen und wertvolle Erfahrungen gesammelt, die bei der Entwicklung des Context-Aware Systems einfließen werden.

[1] vgl. Kapitel 1.1
[2] vgl. Event-Manager oder Shelf (siehe Kapitel 3 und 4)

Literaturverzeichnis

[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

[Hibernate 2007] HIBERNATE (Hrsg.): *Hibernate Official Website*. 2007. – URL http://www.hibernate.org/. – Zugriffsdatum: 17.12.2007

[Hollatz 2008] HOLLATZ, D. (Hrsg.): *smart:shelf, Projektbericht*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-proj/berichte.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

[Jetty 2007] JETTY (Hrsg.): *Jetty Official Website*. 2007. – URL http://jetty.mortbay.com/. – Zugriffsdatum: 21.12.2007

[Meißner 2008] MEISSNER, S. (Hrsg.): *smart:shelf, Projektbericht*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-proj/berichte.html

[MySQL 2007] MYSQL (Hrsg.): *MySQL Official Website*. 2007. – URL http://www.mysql.com/. – Zugriffsdatum: 19.12.2007

[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. – Zugriffsdatum: 05.11.2007

[Urich:Ring 2008] URICH, J. (Hrsg.): *Context-Aware Services: Multimedia-Dienste im Flugzeug*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/bericht.pdf. – Zugriffsdatum: 13.01.2008

[WICKET 2008] DASHORST, M. ; HILLENIUS, E.: *Wicket in Action*. Manning Publications, 2008. – ISBN 978-1932394986