Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Stefan Meißner

smart:shelf

Fakultät Technik und Informatik
Studiendepartment Informatik

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Dieses Projekt entstand aus Überlegungen zu meiner Ausarbeitung zum Seminar „Anwendungen 1“. Ein in Meißner (2007) dargestelltes Szenario behandelt das Auffinden von Objekten mittels RFID-Technologie. Das Ziel, Barrierefreiheit durch den Einsatz von Ambient Intelligence zu erreichen, soll durch teilweise Umsetzung des genannten Szenarios konkrete Formen annehmen. Im smart:shelf-Projekt wird ein Teil des Szenarios untersucht und umgesetzt (vgl. Abschnitt 1.2).
Sehbehinderte Menschen stellen grundsätzlich sicher, dass die ihnen bekannten Gegenstände immer an ihrem zugehörigen Ort aufzufinden sind. Dieses Vorgehen ermöglicht einer blinden Person, schnellen Zugriff auf Objekte und Vermeidung von Unfällen. Wenn jedoch ein Gegenstand falsch platziert ist, oder die blinde Person den Ort des Gegenstandes vergessen hat, kann das Objekt nur durch vorsichtiges, langwieriges Suchen (d. h. Abtasten, etc. . .), wiedergefunden werden. In Regale einsortierte CDs sollten im Optimalfall immer in der selben Reihenfolge stehen, Auffinden wird üblicherweise durch die Verwendung von Braille-Etiketten (Blindenschrift) erleichtert.

1.2 Zielsetzung

Ziel des smart:shelf-Projekts ist, das gegebene Szenario in abgewandelter Form prototypisch umzusetzen. Dabei wird das räumliche Umfeld, folglich der Suchbereich, auf ein einfaches Holzregal beschränkt. Durch das Anbringen von Sensoren sowie Aufstellen eines Rechners, soll die Möglichkeit geschaffen werden, die Gegenwart von Objekten zu ermitteln, d. h. ebenso nach Objekten suchen zu können.
Weiterhin wird eine lose Kommunikationsarchitektur angestrebt, die es ohne aufwändige Installation zulässt, weitere Teilnehmer (hier: Regale) hinzuzufügen. Dadurch wäre es vorstellbar, z. B. viele Verkaufsregale in einem Kaufhaus aufzustellen und ohne weiteres Zutun, Gegenstände in diesen zu suchen.
Die Daten zu den Gegenständen sollen in einer einfachen, relationalen Datenbank hinterlegt werden. Diese Datenbank sowie die Benutzerschnittstellen („Such-Terminals“) sollen

wiederum die oben beschriebene Kommunikationsschicht verwenden und dadurch ebenso leicht erweiterbar oder austauschbar sein.

1.3 Gliederung

Die Arbeit gliedert sich in fünf Kapitel. Neben einem Überblick über die Motivation und Zielsetzung wird in der Einleitung eine grobe Abgrenzung des Projekts geschildert. In Kapitel 2 wird eine Anforderungsanalyse durchgeführt, die zu entsprechender Systemkonzeption in Kapitel 3 führt. Nachfolgende Beschreibung der Umsetzung (Kapitel 4) zeigt die konkreten Ansätze, aber auch Schwierigkeiten in der Realisierung des smart:shelf-Projekts, mit dem Fokus auf das RFID-Regal. Kapitel 5 stellt den aktuellen Stand der Entwicklung vor, fasst die Vorgänge zusammen und gibt einen kurzen Ausblick.

1.4 Abgrenzung

Da das Projekt zeitlich stark begrenzt ist und eine Umsetzung des Szenarios sehr komplex wäre, werden die Teilaspekte nur stark eingegrenzt und teils vereinfacht umgesetzt. Die Erkenntnisse aus einem solchen Prototyp können in einer folgenden Iterationsstufe zu komplexeren Systemen führen. Eine Integration in ein Ambient Intelligence-Umfeld wäre dabei anzustreben.
Die Kernpunkte des Projekts sollen sein:

- Eingabe der Suche mittels Webanwendung
- Verteilung von Suchanfragen und -antworten
- Ausgabe der Suchergebnisse durch Webanwendung
- Ausgabe des Inventars auf Konsole oder grafischer Benutzeroberfläche

2 Anforderungsanalyse

Das gesamte smart:shelf-Projekt wird in drei Teilen speziell betrachtet. Diese Ausarbeitung behandelt den Teil „RFID-Regal“, d. h. die Konzeption und Umsetzung eines Regals mit RFID-Technologie. Die weiteren Teile behandeln zum einen die Kommunikationsarchitektur (Hollatz, 2008) und zum anderen die Datenbank sowie die Benutzerschnittstelle (Urich, 2008).

2.1 Anforderungen an smart:shelf

Folgende globale Anforderungen werden detailliert in den Projektberichten behandelt:

- Auffinden von Objekten in einem oder mehreren Regalen
  Das System soll ermöglichen, Gegenstände, die in einem Regal liegen, auffinden zu können. Falls mehrere Regale verwendet werden, bzw. ein größeres Regal eingesetzt wird, sollen Informationen über den Ort (Regalnummer, Fach) des Objekts zurückgegeben werden
- einfache Erweiterung des Systems
  Weitere Regale, Datenbanken oder Benutzerschnittstellen sollen hinzugefügt werden können, ohne komplexe Konfigurationen durchführen zu müssen. Anzustreben ist hierbei eine Art „Autokonfiguration“. Bei einem marktreifen Produkt würde dies dazu führen, dass neue Geräte direkt nach Stromversorgung *ad-hoc* im System integriert sind.
- Unabhängigkeit von Plattform und Technologie
  Alle Teile des smart:shelf-Systems sollen untereinander unabhängig von Plattform und Technologie sein. Durch eine geeignete Koordinations-Infrastruktur kann diese Anforderung erfüllt werden.

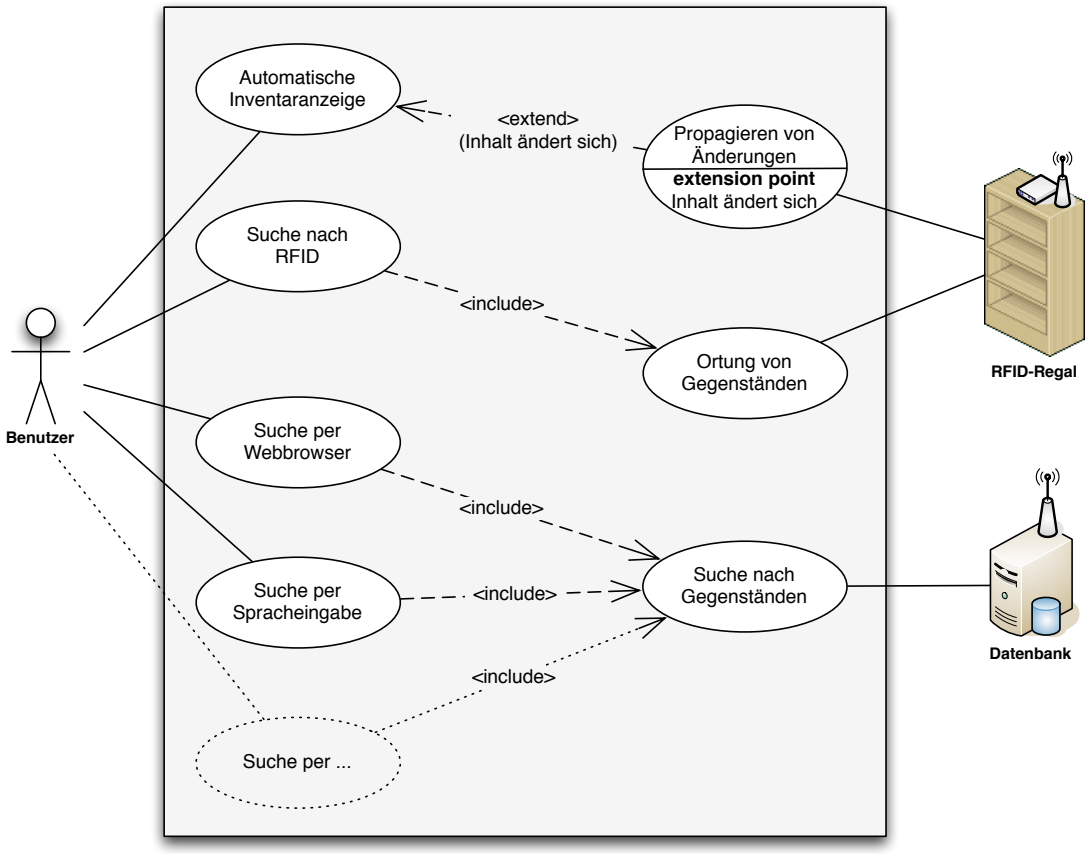

Abbildung 2.1: Anwendungsfälle für das Suchen und Auffinden von Objekten (Hollatz, 2008)

2.2 Anforderungen an die Komponente „RFID-Regal“

Die Anwendungsfälle in Abbildung 2.1 zeigen sehr klare Anforderungen an das gesamte smart:shelf-System. Neben diesen fachlichen Zielen ist für die RFID-Regale anzustreben, folgenden Anforderungen zu genügen:

- Auffinden von Objekten
  Die Objekte, die in das Regal gelegt werden, sollen im Optimalfall sehr genau geortet werden können. Dabei ist eine Einteilung des Systems in Regale, Regalböden und Regalfächer anzustreben.
- Möglichst unauffällige Technik
  Die RFID-Lesegeräte, Antennen und der Regal-Computer sollten versteckt im Regal untergebracht werden. Ziel ist es, kleine Computer unter dem Regal zu befestigen, Antennen an den Böden anzubringen und eine kabellose Vernetzung mit dem smart:shelf-System umzusetzen.

- RFID-Technik muss austauschbar sein
  RFID selbst beschreibt nur ein Verfahren zur Identifizierung mithilfe von elektromagnetischen Wellen. Da verschiedene Standards existieren, muss darauf geachtet werden, dass die Regale verschiedene RFID-Lesegeräte unterstützen sollten. Optimal wäre die Möglichkeit zur gleichzeitigen Identifikation von RFID-Tags verschiedener RFID-Standards.

3 Konzeption

Nachfolgend soll die Konzeption des smart:shelf-Projekts vorgestellt werden. Dabei wird das Design der verschiedenen Komponenten kurz angerissen und genauer auf das Design für das RFID-Regal eingegangen. Ein Überblick über die verschiedenen Komponenten ist in der Systemübersicht (Abbildung 3.1) gegeben.

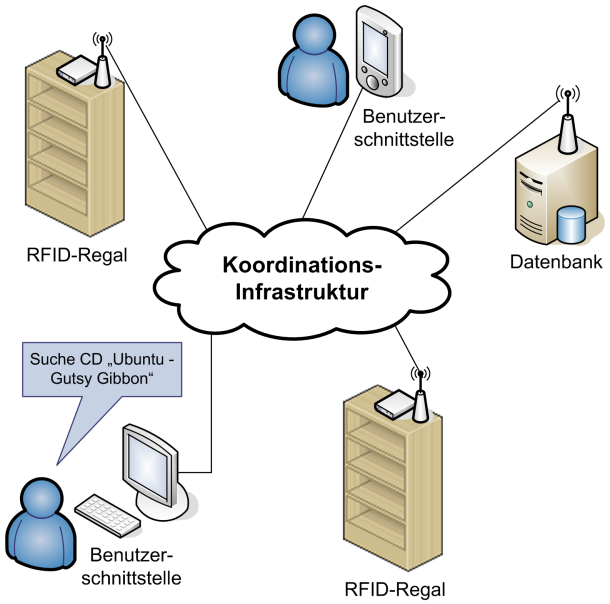

Abbildung 3.1: Systemübersicht smart:shelf

Nachfolgend werden Szenarien des Gesamtsystems beschrieben, um die Aufgaben der Komponenten zu verdeutlichen.

3.1 Szenario: Suchen eines Gegenstandes

- der Benutzer benutzt eine grafische Oberfläche zur Eingabe von Suchkriterien (z. B. „CD-Titel: „Ubuntu - Gutsy Gibbon“)

- die Benutzerschnittstelle sendet diese Anfrage an die Koordinations-Infrastruktur
- die Datenbank bekommt diese Anfrage und kann den Bestand durchsuchen
- die CD wurde in der Datenbank gefunden und besitzt die ID *XYZ123*
- die Datenbank sendet einen „Suche Objekt“-Auftrag mit der ID *XYZ123* an die Koordinations-Infrastruktur
- ein RFID-Regal (A) bekommt den „Suche Objekt“-Auftrag und startet die interne Suche nach Objekt *XYZ123*
- ein weiteres RFID-Regal (B) bekommt ebenso diesen Auftrag und beginnt auch nach *XYZ123* zu suchen
- das Regal (A) findet die CD und sendet das Ereignis „Objekt gefunden“ mit weiteren Informationen (Regal ID, Fach, etc. . .) an die Koordinations-Infrastruktur
- das Regal (B) sucht unter Umständen noch nach der CD und empfängt das Ereignis „Objekt gefunden“ und bricht - je nach Konfiguration - die eigene Suche ab
- die Datenbank empfängt auch das Ereignis über den Fund und kann im Datenbestand den aktuellen Ort des Objekts vermerken
- die Benutzerschnittstelle bekommt nun die Antwort, eben das „Objekt gefunden“-Ereignis, und zeigt die Informationen über den Ort an

Ein leicht abweichendes Szenarien kann z. B. entstehen, wenn Suchbegriffe zu einer Liste von möglichen Objekten aus der Datenbank führen. Diese Liste würde dann zuerst durch die Benutzerschnittstelle angezeigt werden müssen. Der konkrete Suchauftrag kann dann direkt, ohne Datenbank, versendet werden.

3.2 Konzeptionelle Architektur

Die Konzeptionelle Architektur (Abbildung 3.2) zeigt die angebotenen Dienste als Endpunkte und die Koordinations-Infrastruktur als *Event Manager*. Konkrete Dienste wären in diesem Fall z. B. ein RFID-Regal mit Suchfunktion oder ein Datenbankdienst. Die Besonderheit liegt in den Adaptern, die jede Komponente besitzen muss, um über die Koordinations-Infrastruktur Ereignisse empfangen und senden zu können. Diese ereignis-basierte Kommunikation ermöglicht lose Kopplung und dadurch ein *ad-hoc*-Hinzufügen weiterer Dienste.

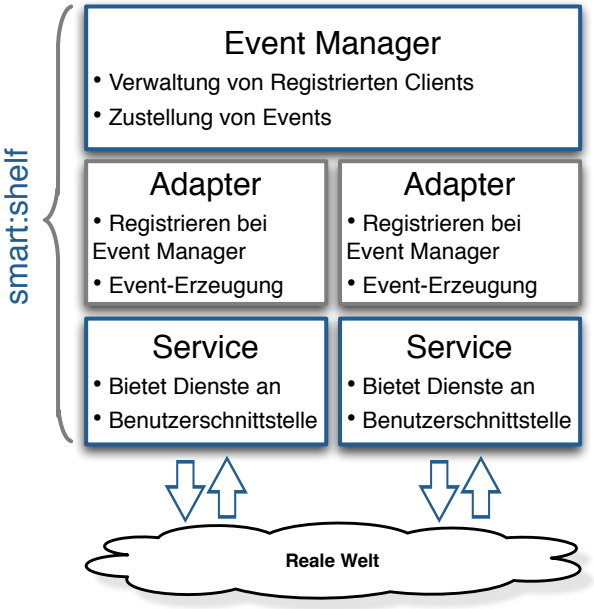

Abbildung 3.2: Konzeptionelle Architektur smart:shelf (Urich, 2008)

3.3 Design des RFID-Regals

Das RFID-Regal bietet den Dienst zum Auffinden von Objekten an. Wie alle Dienste im Gesamtsystem, verfügt dieser Dienst über einen Adapter zur Koordinations-Infrastruktur Wenn ein neues Regal dem System hinzugefügt wird, meldet sich dieses bei dem *Event Manager* an und bekommt ab diesem Zeitpunkt alle relevanten Ereignisse. Die Anfragen müssen nun an die Anwendungslogik des Regals weitergeleitet werden.
Die smart:shelf-Anwendung im RFID-Regal reagiert auf Suchanfragen und spricht die angeschlossenen RFID-Lesegeräte an.
Das Komponentendiagramm (Abbildung 3.3) zeigt die Komponenten eines RFID-Regals.

Event Manager Adapter Dieser Adapter registriert sich bei dem *Event Manager* und ist somit die Schnittstelle zur Koordinations-Infrastruktur. Nach Registrierung für die relevanten Ereignisse eines RFID-Regals, erhält dieser Adapter Informationen über das Auftreten aller entsprechenden Ereignisse. Ereignisse, die das RFID-Regal selbst erzeugt werden über den *Event Manager Adapter* versendet.

Shelf Manager Durch den *Shelf Manager* sind alle Dienste eines RFID-Regals in einer Komponente geschachtelt. Neben der Konfiguration aller *Shelf-Reader* ist der *Shelf Manager* für die ständige Aktualisierung des Inventars zuständig und ermöglicht das Durchsuchen des

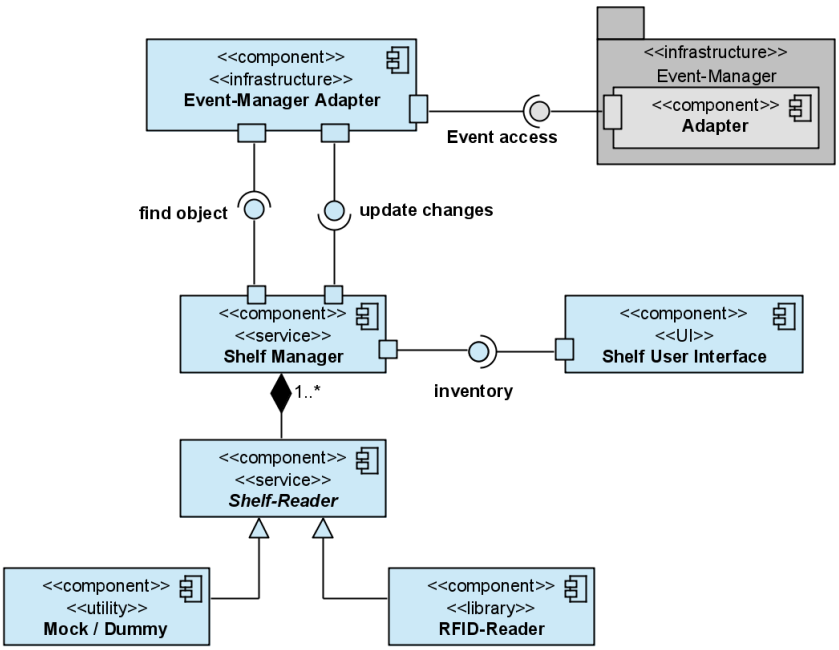

Abbildung 3.3: Softwarekomponenten eines smart:shelf-Regals

Bestands. Je nach Konfiguration meldet der *Shelf Manager* Bestandsveränderungen indem der *Event Manager Adapter* benachrichtigt wird.

Shelf-Reader *Shelf-Reader* sind abstrakte Komponenten, die alle Typen von RFID-Readern aber auch potenziell andere, zur Identifikation von Objekten geeignete Geräte, abstrahieren. Ein *Shelf-Reader* bietet also grundsätzliche Funktionen, wie das Erfassen aller Objekte oder das Auffinden eines einzelnen Objekts anhand einer ID. Die Ergebnisse sind dabei durch den jeweiligen Wirkungsbereich (z. B. Reichweite) eines *Shelf-Readers* begrenzt. Jedes RFID-Regal kann mehrere *Shelf-Reader* verwenden.

RFID-Reader Die *RFID-Reader* Komponenten sind im Regelfall gekapselte Bibliotheken der RFID-Hardware. Durch diese Hardwareabstraktion kann ein RFID-Regal unterschiedliche RFID-Endgeräte gleichzeitig verwenden. Für neue Hardwaretypen wird jeweils eine neue Instanz der *RFID-Reader* Komponente benötigt.

Mock / Dummy Da für das Projekt nicht unverzüglich unterschiedliche RFID-Hardware vorhanden ist, musste in der Konzeption und Umsetzung des RFID-Regals auf Dummies zurückgegriffen werden. Spezielle Mock-Objekte bieten dabei vorgetäuschtes Verhalten eines *Shelf-Readers*, wie z. B. ein leeres Inventar oder Ausgaben von vorgegebenen Testdaten.

Shelf User Interface Zu jedem RFID-Regal kann nicht zuletzt für Funktionstests eine Benutzerschnittstelle gestartet werden. Diese Schnittstelle soll vorerst nur das Inventar, in bestimmten Zeitintervallen aktualisiert, darstellen. Geeignete Benutzerschnittstellen können z. B. Konsolenausgaben oder eine grafische Darstellung der Objekte sein.

3.4 Testkonzeption

Um während der Umsetzung des RFID-Regals schon Testen zu können und dabei nicht von Hardware abhängig zu sein, sollen so genannte Mock-Objekte verwendet werden. Diese Mock-Objekte werden, wie in Abschnitt 3.3 beschrieben, als Dummies für heterogene RFID-Hardware eingesetzt.
Der Einsatz von Mock-Objekten weist folgende Vorteile auf:

- das System kann mit vielen RFID-Readern getestet werden, ohne dass teure RFID-Hardware angeschafft werden muss - Simulation anderer RFID-Standards möglich
- bei nicht vorhandener Hardware (z. B. durch Lieferverzögerungen), kann schon getestet und weiterentwickelt werden
- Tests deterministischer Ereignisse sind möglich

Funktionales Testen soll außerdem durch Ausgabe des aktuellen Inventars auf einer Test-GUI ermöglicht werden. Die Test-GUI kann als *Shelf User Interface*-Komponente auf den *Shelf Manager* zugreifen.

4 Umsetzung

Dieses Kapitel legt die praktische Umsetzung von smart:shelf, sowie Arbeiten, die im Projektverlauf bewältigt wurden, dar. Weiterhin wird auf Schwierigkeiten und auf Kompromisse bezüglich der Realisierung eingegangen.
Die konkrete Umsetzung des smart:shelf-Projekts teilt sich analog der Anforderungsanalyse und Konzeption in die jeweiligen Teile RFID-Regal, Koordinations-Infrastruktur und Datenbank sowie Benutzerschnittstelle. Dieses Kapitel beschäftigt sich primär mit der Realisierung des Dienstes *RFID-Regal*. Die Technologien und Umsetzungsdetails der weiteren Teilprojekte werden in Hollatz (2008) und Urich (2008) detaillierter behandelt.
Die Technologieentscheidungen konnten, nicht zuletzt aufgrund von Vorarbeiten in den Seminaren „Anwendungen 1“ und „Anwendungen 2“, frühzeitig getroffen werden. Grundsätzlich wurde eine plattformunabhängige Lösung angestrebt, so dass Java (Version 1.5) als Programmiersprache eingesetzt wurde. Als Entwicklungsumgebung diente Eclipse 3.3 (Europa).
Die Koordinations-Infrastruktur verwendet das Framework *iROS*[1] der Stanford University, Kalifornien. Der *Event Manager* wird dabei konkret durch die *iROS*-Komponente *EventHeap* umgesetzt. Für weitere Informationen zu *iROS* siehe Johanson und Fox (2004), sowie Johanson und Fox (2002) für Informationen zum *EventHeap*.
Als Datenbanksystem wird eine *MySQL*-Datenbank eingesetzt. Das Persistenz-Framework *Hibernate* bietet hierzu die objektrelationale Abbildung. Die Benutzerschnittstelle für den Anwendungsfall der Objektsuche wurde als Webanwendung realisiert. *Jetty* als Web-Server und das Java-Web-Framework *Wicket* bieten hierfür passende Unterstützung.

4.1 Realisierung des RFID-Regals

Die Bestandteile des RFID-Regals entsprechen den in der Konzeption (Abbildung 3.3) geplanten Komponenten. Während der Entwicklung konnte ein RFID-Reader mit *I-Code*-Technologie (SLRM900) eingesetzt werden (SLRM900, 2002). Dieses Gerät arbeitet mit einer Frequenz von 13,56 MHz im *I-Code*-Betrieb und kann in dieser Betriebsart so genannte „Smart Tags“, d. h. sehr flache Transponder, die auf einer Folie aufgebracht sind,

[1] Interactive Room Operating System

lesen. Mit dieser Frequenz wird eine kurze bis mittlere Reichweite erreicht, im Test konnten ca. 0,5m festgestellt werden. Die „Smart Tags“ (genauer I-CODE1 Label IC) können ohne Stromzufuhr (d. h. passiv) ausgelesen werden und liefern als wichtigste Information eine 64-Bit-Seriennummer (ID). Die Tags können neben der ID noch weitere Daten liefern und speichern, vgl. hierzu I-CODE1 (2002). Bei der Realisierung des Regals wurde darauf geachtet, die Kopplung der Komponenten durch entsprechender Schnittstellen sehr niedrig zu halten. Durch den Einsatz geeigneter Entwurfsmuster und Paradigmen (Observer, ereignisbasierte Kommunikation) ließen sich die Verbindungen zwischen dem *Shelf User Interface*, *Event Manager Adapter* und dem *Event Manager* lose koppeln.
In den folgenden Abschnitten werden die Hauptaufgaben der Realisierung dargestellt.

4.1.1 Kapselung der RFID-Hardware

Die vom Hersteller der RFID-Geräte mitgelieferte Bibliothek (DLL[2]) sowie Testprogramme sind in C respektive C++ realisiert. Eine direkte Nutzung dieser Bibliothek in Java konnte wegen verschiedener, sehr spezieller Implementationsdetails dieser Bibliothek, nicht erreicht werden. Diese Tatsache und das Ziel eine übersichtliche Schnittstelle zu schaffen, führte zur Kapselung der Bibliothek mittels einer weiteren DLL, die in C++ geschrieben ist. Diese Fassade bietet alle benötigten Operationen des RFID-Readers mit sehr klaren Schnittstellen an. Für die Verwendung dieser Fassade mit Java wurde auf JNA[3] zurückgegriffen. Diese Java-Bibliothek erlaubt das direkte Verwenden von DLLs mit einfachen Schnittstellen. Ein gekapselter RFID-Reader entspricht in der konzeptionellen Sicht der *RFID-Reader*-Komponente, d. h. einem *Shelf-Reader*.

4.1.2 Kommunikation per Events - Der Event Manager Adapter

Bei Initialisierung des RFID-Regals versucht der *Event Manager Adapter* den *Event Manager* zu erreichen und registriert sich bei Erfolg für die relevanten Ereignisse des RFID-Regals. Konkret registriert sich der Adapter bei dem *EventHeap* für das „Suche nach ID“-Ereignis. Hiernach bekommt der Adapter alle Ereignisse diesen Typs, erzeugt daraus eine entsprechende Aktion und informiert, mittels Observer-Pattern, den *Shelf Manager*, der diese Aktion abarbeitet.
In umgekehrter Richtung werden Veränderungen durch den *Shelf Manager* dem *Event Manager Adapter* bekannt gemacht, welcher dieses in neue Ereignisse verpackt und an den *EventHeap* sendet.

[2] Dynamic Link Library
[3] Java Native Access: https://jna.dev.java.net/

4.1.3 Eine einfache Benutzerschnittstelle

Um während der Umsetzung des smart:shelf-Systems das Regal selbständig testen zu können, wurden Komponenten zur Ausgabe des aktuellen Bestands implementiert. Neben einer Konsolenausgabe, wurde eine einfache Oberfläche umgesetzt, die auf die Dienste (vorerst den Inventardienst) des *Shelf Managers* zugreift. Für die Oberfläche wurde Swing verwendet. Das RFID-Regal läuft grundsätzlich ohne Benutzerschnittstelle, kann aber durch Verwendung einer GUI überwacht oder administriert werden.

4.1.4 Schwierigkeiten

Während der Umsetzung des RFID-Regals traten Schwierigkeiten verschiedener Art auf, wobei hier auf die hauptsächlichen Komplikationen eingegangen wird.
Grundsätzlich war es schwierig, konkrete Fortschritte zu machen, ohne auf genaue Hardwarespezifikationen respektive RFID-Hardware selbst zugreifen zu können. Durch Lieferverzögerungen und der Heterogenität der RFID-Standards und Geräte, musste für die Hardwareschnittstellen abstrahiert werden. Wie schon in der Konzeption erwähnt (vgl. Abschnitt 3.3) war ohnehin der Einsatz von Mock-Objekten geplant, so dass das RFID-Regal bereits weiterentwickelt werden konnte. Die Mock-Objekte lassen sich mittels Konfiguration hinzuschalten, wodurch es weiterhin möglich ist, verschiedene *Shelf-Reader* einzusetzen, d. h. simultane Verwendung von echter Hardware und den Mock-Objekten.
Nachdem der erste RFID-Reader (SLRM900) geliefert wurde, musste sehr viel Zeit verwendet werden, die mitgelieferte Bibliothek des Herstellers zu abstrahieren. Genauer mussten die hardwarenahen Zugriffe auf das Gerät so gekapselt werden, dass eine passende Schnittstelle für die Verwendung in Java entsteht. Das Vorgehen war aufgrund wenig dokumentierter, unklarer Parameter und Funktionen sehr mühselig. Teile der spezifischen Kommandos und Strukturen konnten dem I-CODE1 Design Guide (2002) entnommen werden.

5 Schluss

Der vorliegende Projektbericht beschreibt die Anforderungsanalyse, Konzeption und Umsetzung eines Systems zur Identifikation, bzw. Suche von Objekten mittels RFID. Das so entstandene smart:shelf-Projekt wurde von drei Projektmitgliedern konzipiert und umgesetzt. Diese Arbeit befasst sich mit dem Teil „RFID-Regal". Die Konzeption und Umsetzung der weiteren zwei Teile kann in den Berichten Hollatz (2008) und Urich (2008) eingesehen werden.

5.1 Stand der Entwicklung

Der aktuelle Stand des smart:shelf-Projekts kann im Grunde die Anforderungen aus Kapitel 2 abdecken. Der Versuchsaufbau besteht derzeit aus:

- Apple Mac mini für ein RFID-Regal
- 19-Zoll-Rack-Server für ein weiteres RFID-Regal
- RFID-Reader für jedes Regal
- Apple Mac mini für die Koordinations-Infrastruktur
- Desktop-PC als Datenbank und Web-Server
- Desktop-PCs als Clients

Zu Demonstrations- und Testzwecken lässt sich auf jedem Regal, sowie auf den Clients jeweils eine grafische Benutzerschnittstelle starten. Die grafische Oberfläche eines Regals zeigt dabei das aktuelle Inventar des Regals (siehe Abbildung 5.1). Die Oberflächen der Clients können noch dazu das gesamte Inventar und dadurch die „Objektbewegungen“ in allen Regalen anzeigen.
Weiterhin wird das in Abschnitt 3.1 beschriebene Szenario vollständig unterstützt.

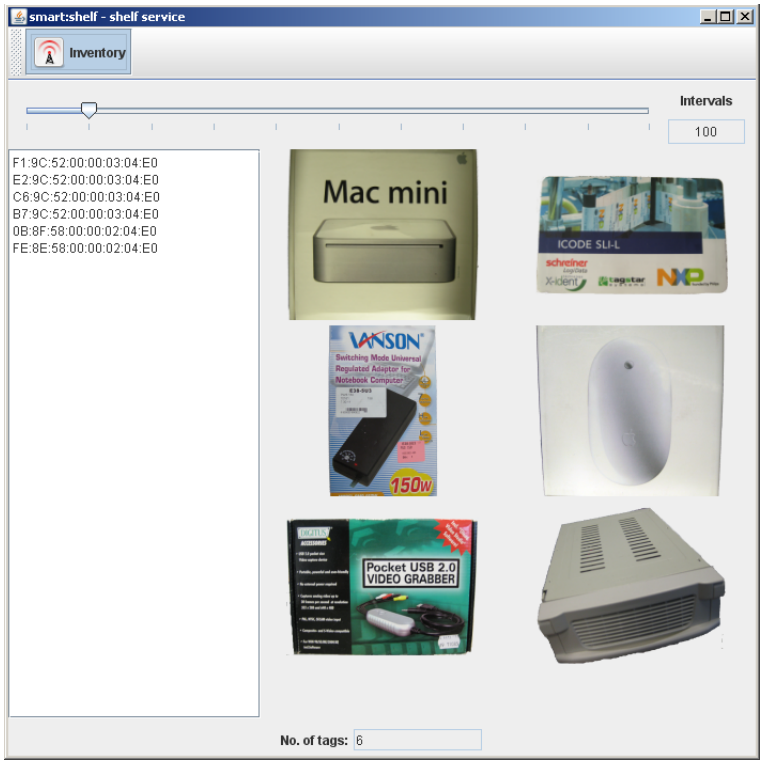

Abbildung 5.1: Screenshot der Benutzeroberfläche des smart:shelf-Dienstes

5.2 Ausblick

Das in dem Projekt entwickelte smart:shelf ist als prototypisches System fähig, die wichtigsten Anforderungen zu erfüllen, weshalb ein Einsatz in dem intelligenten Raum *iLoft* angestrebt wird. *iLoft* ist ein sich in Planung befindliches Projekt der Hochschule für Angewandte Wissenschaften Hamburg und stellt einen Prototyp einer intelligenten Wohnung dar. Da der *EventHeap* für die Kommunikation in diesem Umfeld geplant ist, kann dieser als Koordinations-Infrastruktur des smart:shelfs verwendet werden.
Die RFID-Regale benutzen derzeit ausschließlich die RFID-Hardware eines Herstellers. Da weitere Hardware eines anderen Herstellers angeschafft wurde, ist als nächster Schritt die Kapselung dieser Geräte anzuvisieren. Aufgrund der geringeren Abmaße der neuen Geräte, sollte der Einsatz mehrerer RFID-Reader und Antennen pro Regal untersucht werden.
Für den Einsatz im *iLoft* wäre es angebracht, das smart:shelf nicht ausschließlich als Regal zu nutzen. Der Einbau von RFID-Readern in Schränken, unter Tischen, usw. wäre ebenso möglich. Der Großteil der im *iLoft* verwendeten Objekte sollte für bestmöglichen Nutzen mit „Smart Tags“ versehen werden.

Literaturverzeichnis

[Hollatz 2008] HOLLATZ, Dennis: Projektbericht smart:shelf / Department of Computer Science, Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/projects.html, Feb 2008. – Forschungsbericht

[I-CODE1 2002] PHILIPS SEMICONDUCTORS: *Datasheet: SL1ICS3001 I-CODE1 Label IC*, 2002. – URL http://www.nxp.com/acrobat_download/datasheets/SL1ICS3001_2.pdf. – Zugriffsdatum: 28. Februar 2008

[I-CODE1 Design Guide 2002] PHILIPS SEMICONDUCTORS: *ICODE1 System Design Guide*, 2002. – URL http://www.nxp.com/acrobat_download/other/identification/SL048611.pdf. – Zugriffsdatum: 28. Februar 2008

[Johanson und Fox 2002] JOHANSON, Brad ; FOX, Armando: The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In: *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA : IEEE Computer Society, 2002, S. 83. – ISBN 0-7695-1647-5

[Johanson und Fox 2004] JOHANSON, Brad ; FOX, Armando: Extending tuplespaces for coordination in interactive workspaces. In: *J. Syst. Softw.* 69 (2004), Nr. 3, S. 243–266. – ISSN 0164-1212

[Meißner 2007] MEISSNER, Stefan: Barrierefreiheit mithilfe von Ambient Intelligence / Department of Computer Science, Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/meissner/bericht.pdf. – Zugriffsdatum: 28. Februar 2008, Jul 2007. – Forschungsbericht

[SLRM900 2002] SEMICONDUCTORS), Philips: *SLRM900 I-CODE Long range reader module hardware*, 2002. – URL http://www.nxp.com/acrobat/datasheets/SLRM900_2.pdf. – Zugriffsdatum: 28. Februar 2008

[Urich 2008] URICH, Jaroslaw: Projektbericht smart:shelf / Department of Computer Science, Hamburg University of Applied Sciences. URL http://users.informatik.haw-hamburg.de/~ubicomp/projects.html, Feb 2008. – Forschungsbericht