Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

smart:shelf

Dennis Hollatz

Projektbericht

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Dennis Hollatz

Projektbericht

smart:shelf eingereicht im Rahmen des Master-Projekts
im Studiengang Informatik (Master of Science)
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 3. März 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Motivation

Die Idee zur Umsetzung dieses Projekts entspringt in ihrem Grundgedanken den Ideen, die von Stefan Meißner in Meißner (2007) entwickelt wurden. Er beschreibt hier u. a. eine Umgebung, die es körperlich eingeschränkten Personen ermöglicht, die Nachteile dieser Einschränkungen auszugleichen. Ein Teil dieser Idee ist die Unterstützung bei der Suche nach Gegenständen. Die Umgebung ist in der Lage speziell markierte Gegenstände ausfindig zu machen und dem Suchenden mitzuteilen, wie er den Gegenstand finden kann. Es wird somit beispielsweise einer blinden Person ermöglicht, verlegte Gegenstände effizient wiederzufinden.

In diesem Szenario sollen die Ideen aus Hollatz (2007) zu einer Kommnikationsarchitektur im kollaborativen Umfeld eingearbeitet werden. Es handelt sich hierbei insbesondere um den in Fox u. a. (2000) und Johanson und Fox (2002) vorgestellten *EventHeap*, bei der der Datenaustausch über eine modifizierte TupleSpaces-Architektur erfolgt[1].

Weitere Einflüsse ergaben sich durch die in Urich (2007) beschriebenen Aspekte der Context-Awareness, deren Eigenschaften sind u. a. Service-Discovery und die dadurch bedingte einfache Erweiterung der Umgebung. Neu hinzugefügte Geräte sollen ohne aufwendige Konfiguration schnell einsatzbereit sein und dem Benutzer in vollem Umfang zur Verfügung stehen.

1.2 Zielsetzung

Das Ziel des Projekts ist es, ein prototypisches System zu schaffen, das die generische Suche nach Gegenständen ermöglicht. Ein Nutzer soll die Suche über ein für ihn angepasstes Eingabesystem anstoßen können, die dann innerhalb des Suchbereichs ein Ergebnis ermittelt und dem Nutzer anschließend über ein für ihn passendes Ausgabesystem mitgeteilt werden, wo er den entsprechenden Gegenstand finden kann.

[1] vgl. hierzu auch die in Erman u. a. (1980) beschriebenen Blackboard-Architekturen

Um die Leistungsfähigkeit des *EventHeap* (Johanson und Fox, 2002) überprüfen zu können, soll die Kommunikation innerhalb des Systems ausschließlich über diesen geschehen. Ausnahmen sollen nur dann getroffen werden, wenn sich für die Übertragung bestimmter Daten ergibt, dass eine effiziente oder stabile Übertragung auf diesem Weg nicht möglich ist.

1.3 Gliederung

Diese Arbeit ist in vier Kapitel eingeteilt. Das vorliegende Kapitel 1 gibt einen kurzen Überblick über die Motivation, Zielsetzung, den Aufbau und den Schwerpunkt der Arbeit, sowie der mit dieser Arbeit verbundenen Arbeiten. Anschließend werden in Kapitel 2 das Szenario und die an es gestellten Anforderungen formuliert (Abschnitt 2.3) und in eine konzeptionelle Architektur überführt (Abschnitt 2.4). Kapitel 3 beschreibt die Umsetzung. Es wird hier auf die implementierungsspezifischen Feinheiten des Systems eingegangen. Das abschließende Kapitel 4 fasst die Ergebnisse des Projekts zusammen. Es wird der aktuelle Entwicklungsstand erläutert, eine Bewertung der Ergebnisse vorgenommen und ein Ausblick gegeben.

1.4 Schwerpunkt und verbundene Arbeiten

Das Projekt wurde in zusammenarbeit mit Stefan Meißner und Jaroslaw Urich durchgeführt. Weitere aus dieser Zusammenarbeit resultierende und mit dieser Arbeit verbundene Arbeiten sind Meißner (2008) und Urich (2008).

Meißner (2008) beschäftigt sich hierbei primär mit der der Konzeption und Realisierung der Integration von RFID-Technologie in ein Regal, sowie der Realisierung der Regal-Dienste und einer entsprechenden Testumgebung.

Urich (2008) legt den Fokus auf die Gesichtspunkte der Context-Awareness und hierbei insbesonderen auf die Kommunikation in einem Context-Aware System. Ziel war es, diese zu analysieren und zu testen.

Der Schwerpunkt dieser Arbeit liegt auf der Kommunikation der einzelnen Komponenten mit Hilfe einer auf TupleSpaces und Events basierenden Technologie. Es soll hierbei die Eignung und Leistungsfähigkeit des *EventHeap* (Johanson und Fox, 2002) getestet werden und konkrete Erfahrungen im Umggang mit der Architektur gesammelt werden. Diese Erfahrungen sollen u. a. als praktische Grundlage für die in Hollatz (2008a) gesammelten Vergleiche gelten.

2 Szenario – smart:shelf

Dieses Kapitel beschreibt das Szenario *smart:shelf*. Es wird zunächst eine Abgrenzung zu dem in Meißner (2007) vorgestellten Szenario auf eine während des Projektzeitraums realisierbare Lösung vorgenommen. Aus der Abgrenzung wird dann im Folgenden eine Systemvision entwickelt, von der im anschließenden Abschnitt die Anforderungen abgeleitet werden. Abschließend mündet dieses Kapitel in einem konzeptionellen Entwurf, der die Grundlage für die anschließende Realisierung des Systems bildet.

2.1 Abgrenzung

Da über die relativ kurze Projektdauer das Szenario nicht vollständig umgesetzt werden konnte, wurde zunächst eine Eingrenzung auf die Kernpunkte der Idee vorgenommen:

- Eingabe von Anfragen
- Verteilung von Suchanfragen
- Ausgabe des Suchergebinsses

Hierrauf aufbauend sollte ein Suchsystem geschaffen werden, das es ermöglicht, Gegenstände in einer Menge von Regalen zu aufzufinden. Es sollte möglich sein, verschiedenste Ein- und Ausgabeformen verwenden zu können. Es ist hierbei denkbar, eine Suchabfrage sowohl über ein Web-Interface, einen Client auf einem Mobiltelefon oder auch per Spracheingabe zu stellen.

Die Verteilung der Suchanfragen an die jeweiligen Regale soll ohne besonderen Konfigurationsaufwand möglich sein. Das hinzufügen einer weiteren Komponente soll bestenfalls automatisch geschehen.

Die Antwort mit den Suchergebnissen soll auf einem für den Anfragenden geeigneten Kanal geschehen. Im Regelfall wird dies über den Client geschehen, der die Anfrage gestellt hat. Die Antwort kann allerdings auch auf einem anderen Wege geschehen (z. B. Suchanfrage über Spracheingabe, das Ergebnis der Suche wird über die Lautsprecher des Zimmers wiedergegeben).

2.2 Systemvision

Bei *smart:shelf* handelt es sich um ein System, dass das Orten von Gegenständen erlaubt. Es kann sich hierbei um einige Regale handeln, die beispielsweise mit RFID-Lesegeräten ausgestattet sind. Mit Hilfe der Lesegeräte ist es möglich mit RFID-Tags versehene Gegenstände zu identifizieren. Jedes Regal soll als eigenständige Instanz im System betrieben werden und auf Anfragen nach seinem Inhalt oder nach einem bestimmten Gegenstand reagieren können.

Es sind zwei Benutzerschnittstellen geplant:

- Der Nutzer des Systems soll seine Anfragen über ein Web-Interface stellen können und auch über dieses seine Antwort bekommen.
- Ein weiterer Client soll ständig den aktuellen Inhalt aller Regale anzeigen und so als virtuelle Repräsentation dieser Regale dienen.

Die Verteilung aller Nachrichten soll zentral über den *EventHeap* erfolgen.

Das System hält dabei viel Logik in den jeweiligen Clients. Es ist daher notwendig die Kommunikation durch die Definition klarer Schnittstellen stabil zu halten. Die Verlagerung nahezu aller Anwendungslogik auf die dienstanbietenden Clients schafft eine Umgebung in der sehr flexible Anwendungsmöglichkeiten denkbar sind. Sie erschwert allerdings auch die Verteilung und das Update einzelner Clients auf eine neue Softwareversion.

2.3 Anforderungen

Auf Basis der Abgrenzung und der Systemvision werden die folgenden funktionalen Anforderungen für das System formuliert:

Suche nach Gegenständen anhand ihrer Eigenschaften. Um den Nutzen möglichst groß zu halten soll es nicht nur möglich sein, über den Namen nach einem Gegenstand zu suchen, sondern auch nach bestimmten Eigenschaften. Beispielsweise ist man auf der Suche nach allen Filmen eines bestimmten Regisseurs. Oder man ist auf der Suche nach einer CD mit einem bestimmten Lied, weiß allerdings nicht, auf welcher CD sich das Stück befindet. Die Suche nach dem Titel kann dann die CD und ihre Position liefern. Es ist auch denkbar, dass das Ergebnis eine Liste mehrerer CDs umfasst, da sich der Titel mehr als einmal in der Sammlung befindet. Es soll dabei die Möglichkeit für beliebige Formen der Eingabe gegeben sein, so dass beispielsweise die Suche über einen Webbrowser genauso möglich ist, wie über Spracheingabe.

Auffinden eines eindeutigen Gegenstandes anhand seiner RFID. RFID-Tags sind inzwischen sehr klein und kostengünstig zu bekommen. Es ist zu erwarten, dass die Preise für diese Technologie weiter fallen werden. Jeder Gegenstand des Systems soll mit Hilfe eines RFID-Tags eindeutig registriert werden, um ihn später berührungslos in einer großen Menge von Gegenständen wieder identifizieren zu können. Nachdem ein Gegenstand über seinen Namen oder anhand anderer Eigenschaften gefunden wurde, soll dieser Gegenstand anhand seiner eindeutigen RFID im System aufgefunden werden können.

Popagation von Inhaltsänderungen Jedes Regal soll – je nach Konfiguration – Änderungen seines Inhalts öffentlich propagieren. Jeder Client, den diese Änderungen interessieren, soll über diese Zustandsänderungen informiert werden. Die Bereitstellung dieser Informationen soll es ermöglichen, eine fortlaufende Inventur aller Regalinhalte durchführen zu können.

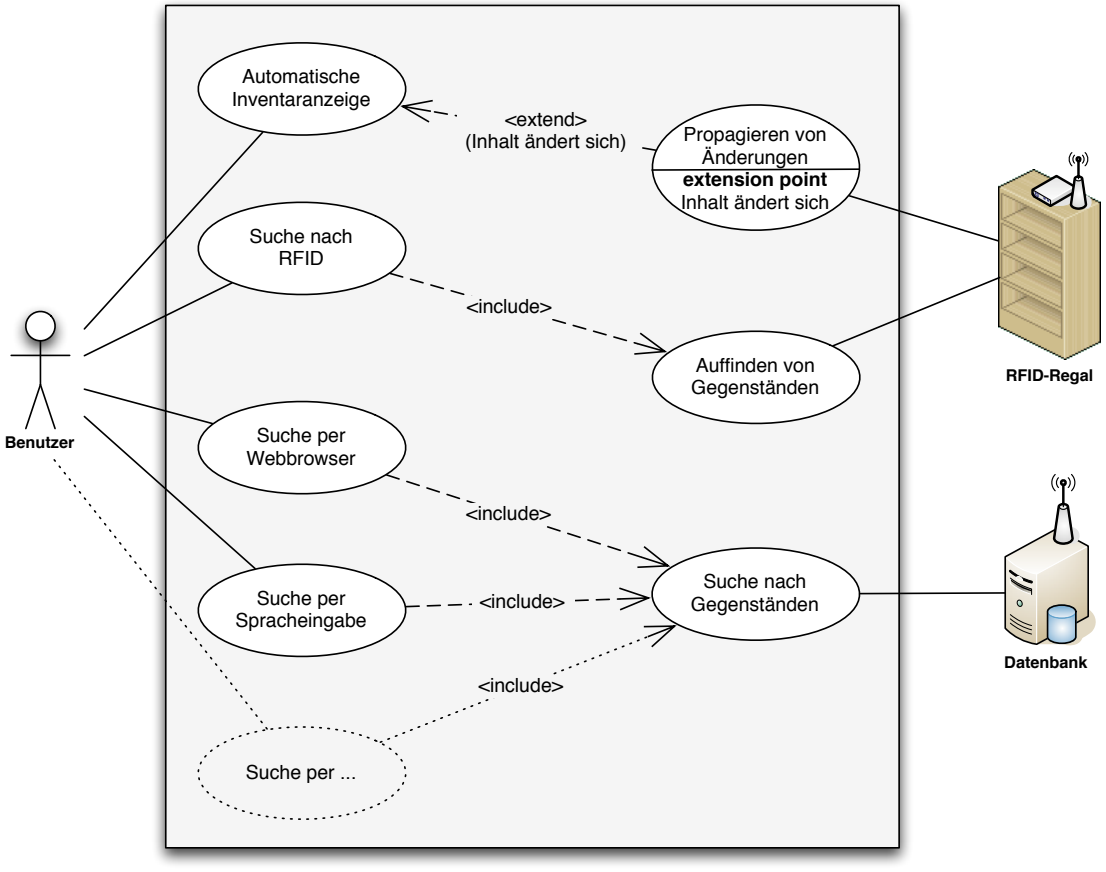

Abbildung 2.1: Usecases für das Suchen und Auffinden von Gegenständen

Abbildung 2.1 zeigt die oben aufgeführten Anwendungsfälle in Relation zueinander.

Ergänzend zu den funktionalen Anforderungen werden noch die folgenden nichtfunktionalen Anforderungen definiert:

Einfache Installation der Regale (bzw. ihre Integration in das System). Die Installation der Regale und der Infrastruktur soll möglichst einfach sein. Insbesondere das Hinzufügen weiterer Regale soll im Idealfall nicht komplizierter sein, als der Aufbau eines herkömmlichen Regals. Dies ist ein wichtiges Akzeptanzkriterium, da ein zu kompliziertes System nicht genutzt werden würde.

Einfache Erweiterung des Systems. Das System soll sowohl einfach um weitere Regale erweitert werden, als auch um weitere Formen der Ein- und Ausgabe. Auch soll die Datenhaltung flexibel genug sein, um mit verschiedensten Formen von Elementen umgehen zu können, wie z. B. Schuhe, Flaschenöffner, Elektrogeräten oder Schallplatten.

2.4 Konzeptioneller Entwurf

Das System *smart:shelf* besteht aus vier verschienen Komponenten: der Benutzerschnittstelle, den Regal-Clients, einer Datenbank und der Kommunikationsinfrastruktur. Abbildung 2.2 zeigt einen Überblick des Systems und des Zusammenspiels der einzelnen Komponenten.

Abbildung 2.3 zeigt die konzeptionelle Architekur des Systems. Ein Dienst verarbeitet die Ein- und Ausgaben der *realen Welt*. Zur Kommunikation mit anderen Diensten werden die Eingaben durch einen Adapter in Events aufbereitet, die über den Event-Manager verteilt werden. Andere Services registrieren sich für bestimmte Services und erhalten diese daraufhin vom Event-Manager, wenn ein Event des Typs gesendet wird.

Jeder Event enthält eine ID, anhand derer der Absender und die zugehörige Anfrage eindeutig identifiziert werden können. Der Empfänger versieht seine Antwort ebenfalls mit dieser ID, so dass der Empfänger die Antwort auf seine Anfrage erkennen kann.

2.4.1 Benutzerschnittstelle

Die Benutzerstelle soll zunächst als Web-Client realisiert werden. Sie soll sowohl als Eingabe, als auch als Ausgabeschnittstelle dienen. Der Web-Client ist der Prototyp eines anfragenden Clients. Der zweite Client soll permanent den Inhalt der angemeldeten Regale anzeigen. Er soll als Prototyp für eine Anwendung gelten, die permanent über Änderungen des Regalinhalts informiert wird.

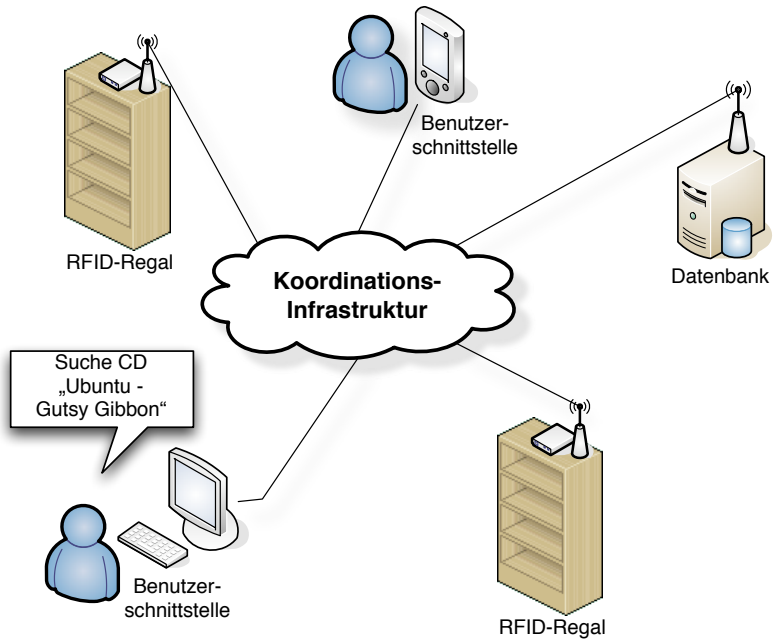

Abbildung 2.2: Systemüberblick
(aus Meißner (2008))

2.4.2 Regaldienst

Der Regaldienst stellt fest, welche IDs sich durch seine Sensoren erfassen lassen. Er führt regelmäßig eine Inventur durch und sendet bei Änderung des Inhalts eine Liste der gefundenen IDs. Er kann auf Anfrage nach einer speziellen ID suchen und auf diese Anfrage Antworten.

2.4.3 Datenbankdienst

Der Datenbankdienst beantwortet Anfragen nach Objekteigenschaften und stellt die Übersetzung in IDs her. Er liefert zu einer Suchanfrage eine Liste von IDs, die dieser entsprechen. Er empfängt Anfragen, bearbeitet diese und schickt seine Antwort zurück an den Event-Manager.

In der Regel wird es nur einen Datenbankdienst pro Systemaufbau geben, es ist allerdings möglich, dass z. B. mehrere spezialisierte Datenbanken für verschiedene Objekttypen existieren. Auch ist dieser Dienst unabhängig von seiner Implementierung als Datenbank denkbar.

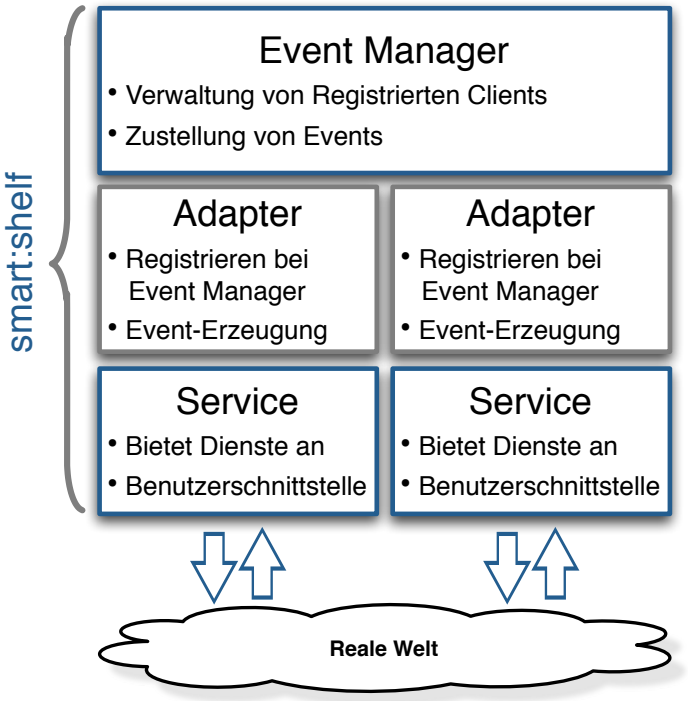

Abbildung 2.3: Konzeptionelle Architektur von smart:shelf
(nach Urich (2008))

2.4.4 Kommunikationsinfrastruktur

Die kommunikationsinfrastruktur verarbeitet die Anfragen der Dienste. Er verwaltet Event-Registrierungen und leitet Events an die entsprechenden Dienste weiter. Er hält zudem einzelne Event-Tupel bis zum Ablauf eines durch den Event vorgegebenen Timeouts vor, um neu registrierten Clients die Möglichkeit zu geben, auch ältere Events zu abzurufen.

2.5 Analyse des Kommunikationsaufkommens

Innerhalb des Systems wird über eine Reihe verschiedenartiger Nachrichten kommuniziert, die sich vor allem in der Art und der Anzahl der Empfänger unterscheiden. Es handelt sich hierbei um Anfragenachrichten, die an alle in Frage kommenden Empfänger gesendet werden und um Antworten auf diese Anfragen, die an einen (oder wenige) Empfänger geschickt werden. Die Abbildung 2.4 zeigt eine Übersicht der häufigsten im System versendeten Nachrichten. Jede Nachricht wird indirekt über den *Event Manager* weitergeleitet. Dieser Aspekt wurde in der Grafik nicht abgebildet, um die Übersichtlichkeit nicht einzuschränken.

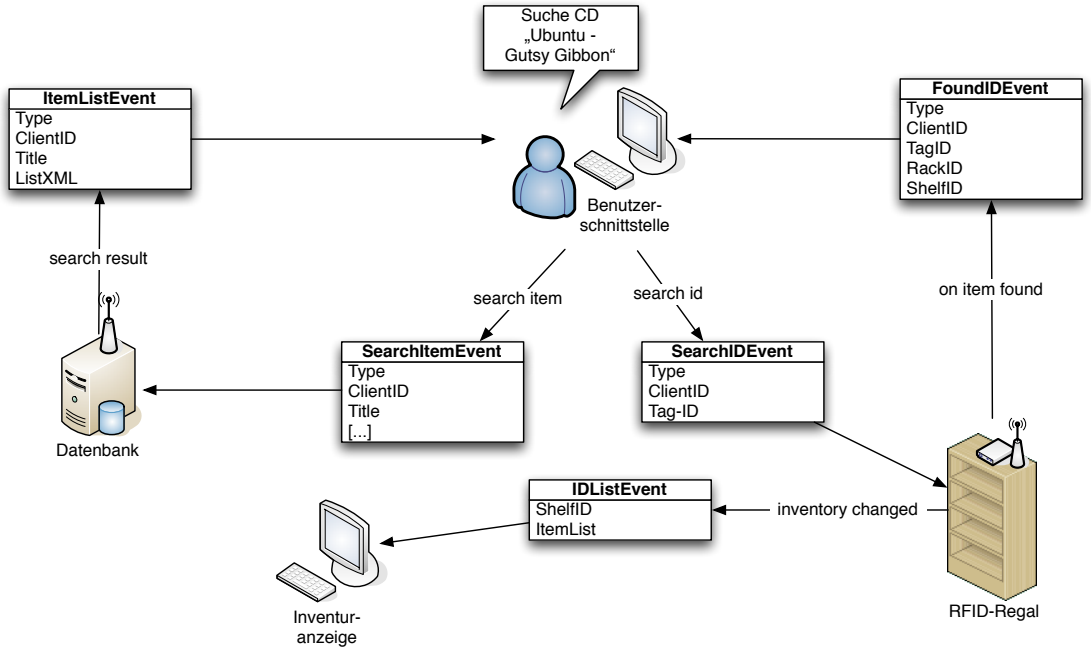

Abbildung 2.4: Schema des Nachrichtenaufkommens

2.5.1 Anfrage nach Gegenstand/ID

Es handelt sich hierbei um die Anfrage eines Dienstes an den Datenbankdienst. Sie enthält eine Liste aller in der Suchanfrage genannten Felder und eine eindeutige Kommunikations-ID. Der anfragende Dienst registriert sich vor dem Absenden der Nachricht für Antwortnachrichten mit einer von ihm generierten eindeutigen Session-ID. Diese teilt er den Empfängern des Events mit dem gesendeten Event mit, so dass diese ihre Antworten so erstellen können, dass lediglich der Absender direkt über Antworten zu seiner Anfrage informiert wird.

Die Anfrage nach einer ID läuft nach dem gleichen Muster ab, lediglich die Typen der erzeugten Events unterscheiden sich von der Anfrage nach Gegenstandsanfragen.

2.5.2 Inventurnachrichten

Der Regaldienst sendet eine Liste der von ihm erfassten IDs an den *Event Manager*, wenn es eine Änderung seines Inhalts feststellt. Die Aktualisierung kann beispielsweise durch periodisches Scannen von RFID-Tags erfolgen. Interessierte Clients können sich beim *Event Manager* für dieses Event registrieren.

Der Regaldienst erwartet auf dieses Status-Event keine Antwort, daher ist es auch nicht erforderlich, diesem Event eine Kommunikations-ID anzufügen.

3 Umsetzung

In diesem Kapitel wird die konkrete Umsetzung des im Kapitel 2 beschriebenen Szenarios erläuert. Es werden zunächst die Technologieentscheidungen dargelegt und die hiermit erfolgte Implementierung beschrieben. Im Anschluss erfolgt eine Beschreibung des aktuellen Entwicklungsstands.

3.1 Konkrete Implementierung

Das Zusammenspiel der einzelnen Komponenten mit dem Regaldienst ist in Abbildung 3.1 dargestellt. Der *Shelf-Manager*-Dienst verwendet mindestens einen Reader, der den Inhalt des Regals ermitteln kann. Hierbei wurde bewusst eine Architektur gewählt, die einen leichten Austausch der konkreten Implementierung des Lesegeräts erlaubt. So kann das System einfach über einen Mock-Reader getestet werden und ist nicht von der konkreten Anwesenheit spezieller RFID-Tags abhängig. Durch die Kombination verschiedener Readertypen ist es möglich auch Gegenstände zu erfassen, bei denen die RFID-Technologie nicht funktioniert[1]. Der *Shelf-Manager* wird zudem von einem User-Interface benutzt. Dieses dient der Anzeige des derzeitigen Regalinhalts.

Die kommunikation mit anderen Diensten erfolgt über die Komponente des *Event-Manager-Adapters*. Dieser registriert sich beim Event-Manager und kann daraufhin über diesen mit anderen Diensten Nachrichten austauschen. Die konkrete Implementierung der Kommunikation bleibt somit fast vollständig vor dem Client verborgen und wird somit leicht austauschbar – ein wichtiges Transparenzkriterium verteilter Systeme (s. Coulouris u. a. (2005)).

Für die Infrastruktur wurde auf die unveränderte Version des an der Stanford University entwickelten *EventHeap* zurückgegriffen. Dieser wurde dort ebenfalls als zentraler Bestandtteil des *interactive Workspaces*-Projekts zur Koordination der Kommuniktaion verschiedenster Geräte eingesetzt (s. Stanford HCI Group (2008)).

Die Realisierung des Systems erfolgt zu großen Teilen in Java. Für den entwickelten Web-Client wurden der *Jetty*-Servlet-Container und das *Wicket*-Framework verwendet. Für die Kommunikationsinfrastruktur wurde der in Johanson und Fox (2002) beschriebene

[1] wie z. B. metallbeschichtete Gegenstände oder Flüssigkeiten

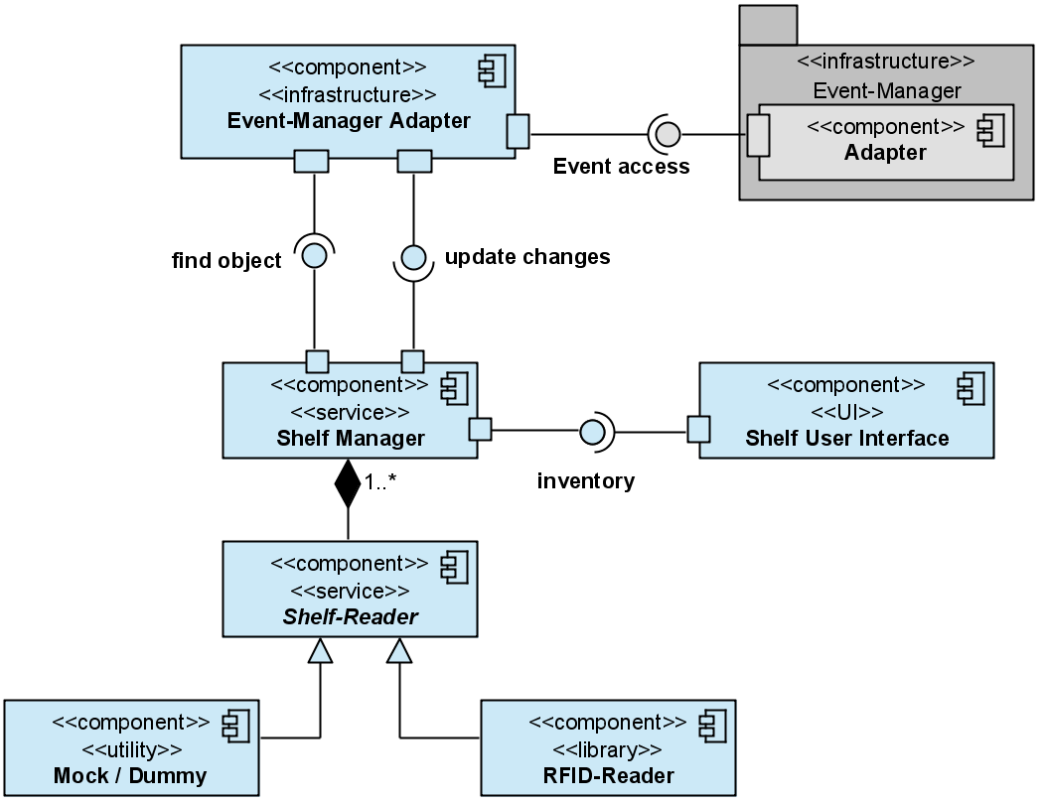

Abbildung 3.1: Komponentendiagramm
(aus Meißner (2008))

EventHeap eingesetzt. Der Zugriff auf die RFID-Reader erfolgt über eine vom Hersteller gelieferte DLL und einer in C++ entwickelten Fassade. Die Schnittstelle zwischen der nativen Schnittstelle und Java wurde mit JNA realisiert, welches einen vereinfachten Zugriff auf die JNI ermöglicht.

3.2 Stand der Entwicklung

Es wurde über die Projektdauer ein lauffähiges und stabiles System entwickelt, das es ermöglicht alle Eckpunkte des Szenarios abzudecken. Alle Teilkomponenten sind lauffähig und kommunizieren über den *EventHeap* miteinander.

Der Regaldienst ist in der Lage Anfragen zu verarbeiten und Inventaränderungen zu propagieren. Abbildung 3.2 zeigt die prototypische GUI des Regaldienstes. Die Anfragen können von einem Web-Client aus gestellt werden, der ebenfalls die Antworten verarbeitet. Die Verarbeitung der Inventur-Events geschieht über einen Dienst mit Java-Swing-Oberfläche.

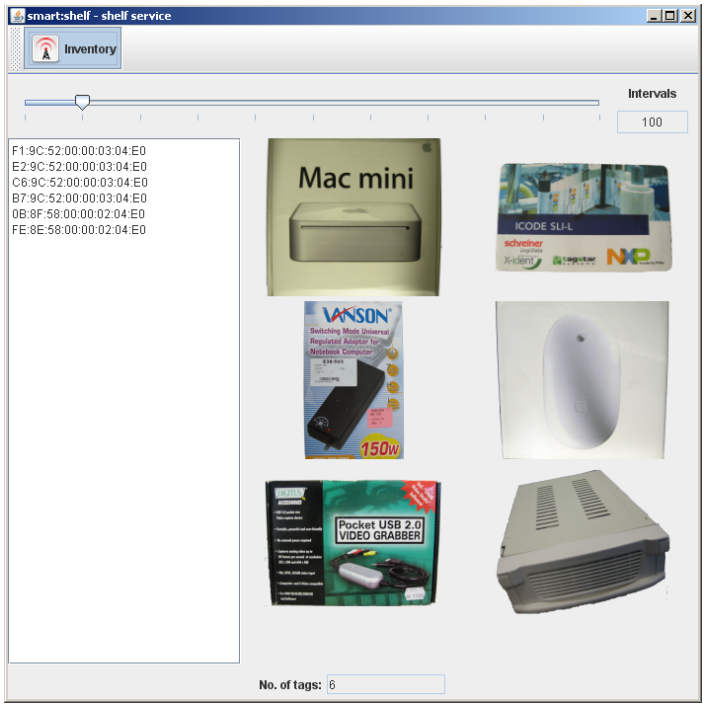

Abbildung 3.2: Shelf Service – Screenshot

Auf die Entwicklung weiterer Ein- oder Ausgabekomponenten wurde aus Zeitgründen verzichtet. Durch die generische Bereitstellung der Schnittstellen zwischen den einzelnen Komponenten und die klare Definition der Events ist eine Erweiterung um weitere Komponenten zur Ein- und Ausgabe möglich.

4 Resümee

4.1 Berwertung des Projektergebnisses

Das Projektergebnis ist ein stabiles System, das alle gewünschten Eigenschaften aufweist. Es wurde gezeigt, dass die kommunikation in mit Hilfe des *EventHeaps* möglich ist. Gemessen an den in Kapitel 2 beschriebenen Anforderungen wurde das Projektziel erreicht.

Ein Kritikpunkt wird allerdings in der Kommunikation zwischen zwei Partnern deutlich. Der *EventHeap* erlaubt zwar das setzen von Timeouts für Events, jedoch nicht für Event-Registrierungen. Wenn sich ein Client für die Antworten auf eine Anfrage anmeldet, so ist der Client auch dafür verantwortlich, diese Registrierung wieder zu löschen, wenn sie nicht mehr benötigt wird. Dieser Aspekt war bei der Implementierung des *EventHeap* nicht wichtig, gewinnt allerdings im Szenario *smart:shelf* an Bedeutung, da es hier eine gesteigerte Anzahl von Kommunikation zwischen zwei Partnern gibt. Der *EventHeap*-Server läuft somit Gefahr, von schlecht implementierten Clients mit Registrierungen überladen zu werden, was schlimmstenfalls zu einem Ausfall des Servers führen kann. Da es sich bei iROS (einschließlich des EventHeap) um Open-Source-Software handelt, ist eine Anpassung an dieser Stelle möglich. Eine andere Alternative wäre, den *EventHeap* nur als Vermittler zu nutzen und den eigentlichen Nachrichtenaustausch, bzw. die Antworten auf Anfragen, direkt zwischen den Clients auszutauschen.

4.2 Ausblick

Das smart:shelf ist als Bestandteil des für 2008 an der Hochschule für Angewandte Wissenschaften Hamburg geplanten *Ambient Intelligence Labors* geplant. Es handelt sich dabei um den Prototypen einer intelligenten Wohnumgebung, der beständig Erweitert werden soll. Als Basis für die gesamte Kommunikation in dieser Umgebung soll der *EventHeap* dienen. Die Erfahrungen aus dem Projekt haben gezeigt, dass dieses Vorhaben prototypisch bereits möglich ist. Für einen produktiven Dauerbetrieb seien hier allerdings die genannten Anpassungen empfohlen.

Es ist somit sinnvoll eine auf den Betrieb im *Ambient Intelligence Labor* angepasste Eigenentwicklung eines Event-Managers zu schaffen. Dieser könnte dann eine direkte Unterstützung aller geforderten Anwendungsfälle bieten, insbesondere eine verbesserte Unterstützung für die bidirektionale Kommunikation zwischen mehreren Diensten bieten. Hollatz (2008b) beschreibt, wie sich ein solches Vorhaben praktisch umsetzen lässt.

Literaturverzeichnis

[Coulouris u. a. 2005] COULOURIS ; DOLLIMORE, Jean ; KINDBERG, Tim: *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science)*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2005. – URL http://www.cdk4.net/. – Zugriffsdatum: 28.02.2008. – ISBN 0321263545

[Erman u. a. 1980] ERMAN, Lee D. ; HAYES-ROTH, Frederick ; LESSER, Victor R. ; REDDY, D. R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *ACM Comput. Surv.* 12 (1980), Nr. 2, S. 213–253. – URL http://doi.acm.org/10.1145/356810.356816. – Zugriffsdatum: 28.02.2008. – ISSN 0360-0300

[Fox u. a. 2000] FOX, Armando ; JOHANSON, Brad ; HANRAHAN, Pat ; WINOGRAD, Terry: Integrating Information Appliances into an Interactive Workspace. In: *IEEE Computer Graphics and Applications* 20 (2000), /, Nr. 3, S. 54–65. – URL http://citeseer.ist.psu.edu/fox00integrating.html. – Zugriffsdatum: 28.02.2008

[Hollatz 2007] HOLLATZ, Dennis: *Konzepte für interaktive Räume*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/vortraege.html. – Zugriffsdatum: 28.02.2008

[Hollatz 2008a] HOLLATZ, Dennis: *Managing Information - Infrastructures for Ambient Intelligence*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-aw/vortraege.html. – Zugriffsdatum: 28.02.2008

[Hollatz 2008b] HOLLATZ, Dennis: *Managing Information - Personal Information Environments auf der Basis von iROS*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/vortraege.html. – Zugriffsdatum: 28.02.2008

[Johanson und Fox 2002] JOHANSON, B. ; FOX, A.: The Event Heap: a coordination infrastructure for interactive workspaces. In: *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on* (2002), S. 83–93. – URL http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1017488. – Zugriffsdatum: 28.02.2008

[Meißner 2007] MEISSNER, Stefan: *Barrierefreiheit mithilfe von Ambient Intelligence*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/vortraege.html. – Zugriffsdatum: 28.02.2008

[Meißner 2008] MEISSNER, Stefan: *Projektbericht – smart:shelf*, Hochschule für Angewandte Wissenschaften Hamburg, Projektbericht, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp. – Zugriffsdatum: 28.02.2008. – in Vorberitung

[Stanford HCI Group 2008] STANFORD HCI GROUP: *Stanford HCI Group – Projects*. 2008. – URL http://hci.stanford.edu/research/index.html. – Zugriffsdatum: 28.02.2008

[Urich 2007] URICH, Jaroslaw: *Context-Aware Services: Multimedia-Unterstützung im Flugzeug*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/vortraege.html. – Zugriffsdatum: 28.02.2008

[Urich 2008] URICH, Jaroslaw: *Projektbericht – smart:shelf*, Hochschule für Angewandte Wissenschaften Hamburg, Projektbericht, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp. – Zugriffsdatum: 28.02.2008. – in Vorberitung