CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Masterstudiengang Informatik

Masterarbeit

Mobile und Echtzeit-fähige Kontext-Modellierung, -Erfassung und -Verwaltung in der Domäne des (Smart) Home Sharings

vorgelegt von

Cosmin Pitu

Gutachter:

Jun.-Prof. Dr. Daniela Nicklas
Prof. Dr. Susanne Boll

Oldenburg, 19. Oktober 2012

Zusammenfassung

Im Laufe dieser Arbeit wird ein Konzept vorgestellt, welches als Erweiterung der “Home Sharing” Domäne (die eigene Wohnung zur Verfügung zu stellen) angesehen wird. Eine Analyse der aktuellen Anbieter von “Home Sharing” Lösungen zeigte, dass die Betrachtung der Domäne aus technologischer Sicht schon bei der Vermittlung der Buchung zwischen Wohnungseigentümer und Wohnungsmieter aufhört.

Es ist der Denkansatz dieser Arbeit, dass das Nutzungserlebnis von “Home Sharing” von einer technologischen Erweiterung profitieren könnte. Zu diesem Zweck wurde eine Forschungsfrage abgeleitet, die die definierten Konzepte der Lösung zusammenfasst. Die Forschungsfrage begleitete damit die Entwicklung der Arbeit und lautete folgendermaßen:

Wie muss ein Kontext-Modell gestaltet sein, dass es Erweiterungen und einen Umgang mit Echtzeit-Daten erlaubt, unter Beibehaltung gewisser Eigenschaften (wie die Allgemeinheit, Flexibilität oder ein erhöhtes Sicherheitsgrad) ?

Zwecks die Beantwortung der Forschungsfrage wurden mögliche verwandte Lösungsansätze anhand einer Literaturrecherche analysiert.

Zu der genannten Analyse der Anbieter von “Home Sharing” Lösungen wurden User Stories definiert, die den möglichen Umgang von Benutzern mit der vorgestellten technischen Lösung beschreiben können. Aufgrund dessen wurden Anforderungen erhoben, die eine technische Architektur für die Lösung des Konzepts bereitstellen. Ein erheblicher Teil der Architektur wird von der Definition eines Kontext-Modells ausgeprägt.

Weiterhin wurden die Konzepte in einer prototypischen Anwendung, bestehend aus zwei Systemen (Backend/Server und Frontend/Client) umgesetzt. Die vorbereitete technische Lösung wurde am Ende der Bearbeitungszeit durch eine Evaluation auf die Erfüllung der Forschungsfrage überprüft. Schließlich zeigte die Analyse der Evaluations-Ergebnisse, dass die angebotene Lösung als hilfreich empfunden wurde, was das Potential der Softwarelösung zeigt und damit weitere Arbeiten daran nicht ausschließt.

Inhalt

1 Einleitung

In diesem Kapitel wird das benötigte Hintergrundwissen vorgestellt, der allgemeine Kontext, sowie die Domäne der zu leistenden Arbeit. Daraus wird eine Forschungsfrage etabliert, welche im Laufe der Arbeit beantwortet wird.

1.1 Übersicht

Die Informationsgesellschaft entwickelt sich in eine neue Richtung, in der sich die Computing-Technologie von dem etablierten Paradigma des festen Arbeitsplatz entfernt [BA09]. Die Tendenz zum mobilen, ubiquitären Computing wird deutlicher. Im Fall des ubiquitären Computing werden verteilte Systeme gebildet, die hauptsächlich aus vernetzten und interaktiven Geräten bestehen und auch verschiedene Apparaturen und deren Sensorik umfassen.

Solche verteilten Systeme ermöglichen einen Zugang zu einer breiten Palette von Diensten aus verschiedenen Domänen und werden normalerweise in unterschiedlichen physische Standorten eingesetzt wie z.B. Häuser, Büros, Autos oder im öffentlichen Raum. Solche Anwendungen zählen zu den Bausteine einer Technologie namens "Ambient Intelligence". Darüber hinaus enthalten sie weiterhin die Idee, dass Applikationen so entwickelt sollen, dass sie für bestimmte Nutzer, deren Umgebung sowie Ausgangssituation während der Benutzung "zugeschnitten" sind oder sich zur Laufzeit adaptieren.

Die Adaptation zu der aktuellen Situation des Benutzers spielt eine große Rolle, wenn man sich in einem unbekannten Kontext befindet. Durch Technik kann dieser Prozess erleichtert werden, damit lassen sich eventuell unentdeckte Vorteile der Umgebung ausnutzen. Allgemein ordnet sich dieses Szenario zu den Paradigmen der "Ambient Assisted Living" Domäne, wo die Unterstützung des Benutzers während seiner täglichen Zielen und Aufgaben im Vordergrund steht [WX08].

Diese Aspekte werden in dieser Arbeit in das sich ständig entwickelnde globale ökonomische Konzept namens *Sharing Economy* eingebettet, auch unter dem Begriff "Gemeinschaftlicher Konsum" (engl. Collaborative Consumption) bekannt. Dieses Konzept basiert auf der Idee, dass Zugang wichtiger als Besitz ist. Dadurch werden diejenige Personen, die Eigentümer eines Produkts oder einer Dienstleistung sind, einer anderen Personen zugeordnet, die dieses Produkt oder Dienstleistung braucht[1]. Durch dieses Modell entstehen sogenannte "Peer-to-Peer-Marktplätze", welche auf sozialen Prinzipien wie gegenseitiges Vertrauen, Ansehen und einem Gemeinschaftsgedanken basieren. Weitere Details sind im Absatz 2.4.1 zu finden.

Ein Teil der Sharing Economy sind virtuelle Marktplätze für Wohnungen (engl. Home Sharing). "Home Sharing" bedeutet das Teilen bzw. Vermieten der eigenen Wohnung. Die genannte Problematik des Einlebens ist in dieser Domäne aus Sicht der Mieter auch vertreten: die neue Wohnung und deren Bestandteile wie Geräte und Hausregeln müssen zuerst entdeckt oder gelernt werden. Eine mobile Anwendung könnte die Mieter dabei unterstützen, sich im neuen Kontext zurecht zu finden und würde es erlauben, mögliche Unklarheiten mit dem Wohnungseigentümer zu fast jeder Zeit klären zu können.

[1] Artikel "Silicon Valley Technology helps power new sharing economy", Dana Hull, San Jose Mercury News, 9. Mai 2012, `http://scienceandtechnologypopular.com/silicon-valley-technology-helps-power-new-sharing-economy-san-jose-mercury-news`

Die Zielsetzung dieser Arbeit ist die Bereitstellung einer technischen Architektur und Anwendungen, die eine mögliche Implementierung des “Home Sharing”-Anwendungsfalles realisiert, während bestimmte Kriterien wie die Allgemeinheit und Sicherheit der Lösung bestehen bleiben.

1.2 Hintergrund

Ein intelligentes Haus (engl. “Smart Home”) wendet die von Mark Weiser im Jahr 1991 definierten Konzepte des “ubiquitären Computings” an: Rechensysteme, die zu jeder Zeit erreichbar sind sowie “intelligent” und möglichst unsichtbar agieren [Wei91]. Weiterhin sind diese Systeme in der Lage, die Bedürfnisse der Benutzer zu erkennen und diese im Alltagsleben zu unterstützen [RQBA09].

Technologien und Anwendungsgebiete wie “Smart Homes”, “Ambient Assisted Living” sowie “Smart Products” stellen Konzepte vor, die diese Vorstellung von Weiser umsetzen können und dadurch neue Wege für eine Welt der vernetzten Geräte öffnen. Trotzdem hat sich in mehr als 20 Jahren keine vollständige, “aus einer Hand” Lösung für ein intelligentes Haus eröffnen. Mögliche Gründe werden in [RQBA09] dargestellt: im Wesentlichen liegt eine fehlende gemeinsame dynamische Architektur für die Installation zugrunde. Des Weiteren lässt sich der Zuwachs des Gerätespektrum nennen, sowie eine fehlende gemeinsame (im Sinne von “normaler” und “Smart” Wohnung) Benutzeroberfläche.

Aktuelle Richtungen zeigen, dass mehr und mehr Geräte vernetzt werden – als Oberbegriff wird das sog. “Internet of Things” verwendet [RQBA09]. Wie genannt hat diese Entwicklung das Internet, das Web als grundlegende Infrastruktur und Übertragungskanal. Diese Initiative malt sich eine Zukunft aus, in der eine nahtlose Integration und Erreichbarkeit von “Smart Objects” im Web ubiquitär ist. Leider ist diese Vision auch in aktuellen Zeiten noch nicht realisiert worden, aufgrund Inkompatibilitäten auf der Geräte-Ebene und fehlende Übertragung- und Integrationsstandards [Dav10].

Diese Entwicklungen haben den globalen Zuwachs von Internet-Benutzern als Hintergrund: schon im Jahr 2005 benutzten 2 Milliarden Menschen das Internet[2], ausschließlich von Desktops oder Notebooks[3]. Anfang 2011 verdoppelte sich schon diese Menge, der nächste Schritt wird voraussichtlich 2017 erreicht. Bis dahin wird erwartet, dass eine weitere Milliarde von Benutzer *Internet-fähige* Geräte wie Smartphones, Tablets oder e-Readern sowie Spielkonsolen oder Mediaplayers benutzen werden. Der Oberbegriff für diese Art von Geräten wäre “nicht-Rechner” (engl. non-computer) oder verallgemeinert “mobile Geräte”. Die Anwendung solcher “nicht-Rechner” Geräte ist ein globales Phänomen, das im August 2011 schon in fünf globalen Märkte (Singapur, Großbritannien, Vereinigten Staaten, Japan und Australien) vertreten war. Bereits 5 Prozent vom Browser-basierten Traffic kam in den benannten Märkten nicht von üblichen Rechnern.

Darüber hinaus werden Smartphones immer wieder mehr aus dem benannten Anteil von “nicht-Rechner” gewinnen: laut International Data Company (IDC), einem international tätigen Marktforschungsunternehmen auf dem Gebiet der Informations- und Kommunikations-Technologien (IKT), hat der globale Telefon-Markt um 11 Prozent im 2. Quartal des Jahres 2011 zugenommen. Damit haben Mobiltelefone, welche nicht zu den Smartphones gehören, zum ersten Mal in fast zwei Jahren an Marktanteil verloren. Das heißt, dass die Smartphone-Industrie ständig an neue Nutzern gewinnt:

[2] Artikel “One Billion Internet Users”, Jakob Nielsen, Erscheinungsdatum: 19. Dezember 2005, `http://www.useit.com/alertbox/internet_growth.html`

[3] vlg. die kommerzielle IT-Publikation “Digital Omnivores: How Tablets , Smartphones and Connected Devices are changing U.S. Digital Media Consumption Habits”, 2011, `http://www.ipmark.com/pdf/Omnivoros.pdf`

wie vom globalen Marktforschungsunternehmen Nielsen bekanntgegeben, sind 40 Prozent aller US-Bürger Smartphone-Anwender, wovon wiederum 40 Prozent das Android Betriebssystem verwenden. Eine sehr aktuelle Studie [4] des Forschungsunternehmen Strategy Analytics zeigt, dass die Anzahl von global benutzten Smartphones zum ersten Mal seit der historische Einführung des Segments im Jahr 1996 mehr als eine Milliarde beträgt.

Weiterhin stehen als Basis dieser Trends die schon ab 2006 erkannten Fortschritte in mobilen und ubiquitären Technologien, wie z.B. die Paradigmen von "Pervasive Computing", eingebettete Sensor-Technologien oder die Vielfalt von drahtgebundenen und drahtlosen Protokollen [ATH06] erkenntlich. Natürlich baut die Vision von "The Internet of Things" auf etablierten Technologien auf und erzielt, wie betont, die Art und Weise in dem die Geräten selbständig mit anderen Geräten aus demselben (lokalen) Netz interagieren und sogar mit den Benutzer kommunizieren.

So bleibt die Entwicklung einer Lösung für die "Smart Home" Domäne fast nur eine Frage der benötigten Infrastruktur: wie muss ein System definiert werden, um das tägliche Leben des Wohnungsbesitzer und -Gäste wesentlich zu vereinfachen? In den nächsten Kapitel wird ein Konzept einer Anwendung vorgestellt, die in einem mobilen Kontext die Anpassung ans Neue (z.B. neue, temporäre Wohnung, Umgebung, Stadtviertel etc.) der Wohnungsmieter begleitet und eventuell auch verbessert.

1.3 Problemstellung

In diesem Abschnitt wird die Problemstellung formuliert, zunächst aus der Sicht der Anwendungsdomäne und darauffolgend unter Berücksichtigung der dadurch entstehenden und durch diese Arbeit zu lösenden technologischen, informatischen Probleme.

1.3.1 Aus Sicht der Anwendungsdomäne

Die Anwendungs-Domäne dieser Arbeit ist als "Home Sharing" bekannt. Wie genannt ist diese Domäne ein Teil des Sharing Economy Konzeptes, welcher im Folgenden Kapitel 2.4.1 detaillierter vorgestellt wird.

Mit dem Begriff "Home Sharing" beschreibt man das freiwillige Teilen bzw. Vermieten der eigenen Wohnung oder einzelner Zimmer in einer Wohnung für normalerweise begrenzte Zeiträume. Eine mögliche, abstrahierte Unterteilung der individuellen Phasen des hauptsächlichen Vorgang Home Sharings ist die folgende:

- **Kontaktaufnahme zwischen Wohnungseigentümer und Wohnungsmieter**: Eine direkte Kontaktaufnahme oder eine, die durch ausgewählte Dienstleister wie z.B. Airbnb[5], Wimdu[6] oder 9flats[7] entsteht. Sie kann zu einer festen Buchung der Wohnung führen.
- **Vorbereitungsphase**: An dieser Stelle könnte der Wohnungseigentümer den vermieteten Raum mit Informationen anreichern, mit z.B. Tipps relativ zu der Nutzung verschiedener Gegenstände

[4] Artikel "Worldwide Smartphone Users Cross 1 Billion Mark", Erscheinungsdatum: 17. Oktober 2012, `http://www.ibtimes.com/worldwide-smartphone-users-cross-1-billion-mark-report-847769`

[5] `http://www.airbnb.com`

[6] `http://www.wimdu.com`

[7] `http://www.9flats.com`

wie z.B. die Waschmaschine oder das “Home Entertainment” System aus der Wohnung oder die Bereitstellung einer Landkarte mit den umgebenden Sehenswürdigkeiten. Diese Phase ist freiwillig und hat als Ziel, den Mietern während des Mietzeitraums zu helfen.

- **(Virtuelle) Einführungsphase**: Der/Die Wohnungsmieter kommen und bekommen eventuell vom Eigentümer eine Tour durch die Wohnung, in der die wesentlichen Informationen vermittelt werden, die nicht aus der Beschreibung der Wohnung bzw. die “Exposé” oder Anzeige ersichtlich sind. Aus sozialer Perspektive ist dieser erste Tour ein Minimum von Hilfe, die den Mieter geleistet werden kann; alle anderen Interaktionen ergeben sich auf einer “erforderlichenfalls”-Basis. Des Weiteren könnte man sich vorstellen, dass eine virtuelle Vorführung der Wohnung auch im Voraus während der Vorbereitungsphase stattfinden kann, mithilfe einer technischen Infrastruktur wie das Web.

- **Mietphase**: Diese Phase beschreibt die eigentliche Zeitspanne in der die Wohnung vermietet wird. Ein auftretender Vorgang an dieser Stelle ist das Einleben der Mieter, zur Erreichung des “wie zu Hause” Gefühls.

- **Endphase**: Mit dem Auszug der Wohnungsmieter wird das System erneut in den anfänglichen Zustand gebracht. Eventuelles Feedback vom Wohnungsmieter kann eingebaut werden und Vorbereitungen für die nächste Vermietung können anfangen.

Die von dieser Arbeit betrachteten Phasen sind die Vorbereitungs-, Einführungs und Mietphase, mit der zusätzlichen Nennung erweiterter Perspektiven der Endphase. Problematik an der Vorbereitungsphase stellt sich die Mangel an (technischen) Architektur, die die “analogen” Pläne des Wohnungseigentümers (z.B. bezüglich der virtuellen, zu jeder Zeit aufrufbaren Tour der Wohnung) in einer digitalen Art und Weise unterstützten und verwirklichen kann. Davon abgesehen, dass die Dienstleister keine solche Lösung anbieten, da sie normalerweise nur auf die Phase der Kontaktaufnahme und Buchung spezialisiert sind, stellt sich diese Ebene als den Einstiegspunkt des Konzeptes.

Der Hintergrund für die Einsetzung einer technologischen Ebene, die die Wohnungsmieter während des Mietintervalls unterstützen kann, lässt sich rückverfolgen bis zu einem der zentralen Prinzipien aus der Domäne “Sharing Economy”: die Zusammengehörigkeit und die Etablierung einer Gemeinschaft für geteilte Nutzung. Sicherlich gehört zu den Gründen auch die Gastfreundlichkeit des Eigentümers, welcher sich freiwillig für die Vermietung angemeldet hat und auf die Zufriedenheit der Gäste achten muss, sodass er/sie künftig mehrere Mieter willkommen kann.

Die Absicht des Wohnungseigentümers ist dann die Erleichterung des Anpassungsprozesses in der Mietphase: dem Wohnungsmieter wird ein neuer Kontext (die Wohnung, Umgebung und anderen Aspekte), neuen Situationen begegnen und dafür ist zu erwarten, dass eine Eingewöhnung nötig wird, da der Mieter versucht, die Wohnung kennenzulernen. Eine technologische Unterstützung dieser Versuches wäre sinnvoll – dabei zu beachten ist, dass die einfache Bereitstellung bzw. Darstellung aller Informationen der Wohnung überlastend wäre: beispielsweise würden Tipps zur Vorbereitung einer Tasse Kaffee im Fall von nicht-Kaffee Trinker überflüssig, während eine Benachrichtigung um 20 Uhr, in der auf eine wichtige Hausregel zurückgegriffen wird, die erst ab 23 Uhr gilt, unnötig ist.

In dieser Hinsicht wird es versucht, eine Situations- bzw. Kontext-abhängige Anpassung zu gewährleisten: z.B. anhand der aktuellen Lokation des Mieters werden nur Informationen aus seiner Nähe dargestellt. Diese Anpassung am Kontext (engl. Context Adaptation) ist zu einer alternativen

Definition von Kontext aus [GL08] zusammengefasst, wo Kontext als "die Fähigkeit, Änderungen der äußeren Umgebung verstehen und darauf reagieren zu können" beschrieben wird.

Die Übertragung dieser Sichtweise in der Domäne von "Home Sharing" bedeutet dass die Wohnungsmieter sich mit der Umgebung (die neue Wohnung) vertrauen werden müssen. Es gibt keine eindeutig anwendbaren Strategie zur Kontext-Anpassung, da der Umfang der möglich eintretenden Situationen die Mächtigkeit jedes Kontext-Modells übersteigen würde. Aus diesem Grund müssen Techniken zur Anpassung entwickelt werden, wie weiter in Kapitel 6 beschrieben.

1.3.2 Aus Sicht der technologischen Herausforderungen

Eine technische Lösung wäre, eine erweiterbare Menge von *möglichen* Bestandteilen des Kontext-Modells zur Verfügung zu stellen, wie weiter in Kapitel 1.5 beschrieben. Dadurch können gewisse Gegenstände (engl. "assets") in der Wohnung modelliert werden, die zu einer bestimmten (vorgegebenen oder erweiterten) Kategorie gehören. Die dadurch erfassten Gegenstände können die Wohnungsmieter bei der Anpassung helfen. Zum Beispiel können Fragen wie "Wo finde ich das Bügeleisen?" eventuell automatisch beantwortet (es sei denn, der Wohnungseigentümer nützlich empfunden hatte, dieser Element in der Modellierung mit einzubeziehen).

Kritisch an dieser Stelle ist die Bereitstellung (durch z.B. eine Darstellung auf dem mobilen Gerät) der so gesehenen "richtigen" Gegenstände (oder "Vorzüge") in einer möglichst *automatischen* Art und Weise. Ein erstes Kriterium für die Auswahl des richtigen Objektes wäre beispielsweise die aktuelle Lokation des anfragenden Benutzers. Solche Informationen bauen zu einer Entscheidung auf und müssen verfolgt und gespeichert werden können – eine Lösung ist die Definition eines Kontext-Modells, welches ein schneller Zugriff auf die vorher modellierten sowie auf neu eintretenden Instanzen dieser Informationen ermöglichen kann. Die Automatisierung ist auch vorstellbar: anhand von mehrerer Informationen *aus der Umgebung* können die Entscheidungen (z.B. über die Aufsicht oder Aktivität des Benutzers, die Situation in der er/sie sich gerade befindet) automatisch vorgeschlagen werden.

Andere mögliche Unklarheiten oder allgemeine Probleme, die während der Mietphase eintreten können, sind Fragen wie beispielsweise "Warum soll nach 23 Uhr keine laute Musik abgespielt werden" oder "Wie funktioniert diese hoch automatisierte Waschmaschine". Konzepte, die diesbezüglich abgeleitet werden können, sind Hausregeln und Anwendungshinweise, die in Kapitel 1.5 eingeführt werden. Diese Informationen sollen auch möglichst automatisch abgerufen werden (wenn z.B. sich der Benutzer das mobile Gerät in der Nähe des Gegenstandes befindet) – hiermit ist eine Technik der Lokalisierung gefragt.

Im Laufe der Mietphase könnte man sich auch Situationen vorstellen, die aus dem Rahmen des Üblichen fallen können – hier können beispielhafte Grenzfälle wie der Defekt eines wichtigen Haushaltsgeräts oder eine außergewöhnliche Anfrage von der Seite der Mieter gehören. In diesen Situationen ist eine möglichst direkte aber trotzdem nicht zu störende Kontaktaufnahme gefragt; darüber hinaus können gravierende Ereignisse dem Dienstvermittler oder dem Wohnungseigentümer direkt kommuniziert werden. Diese Konzepte sind als das "Kommunikationskanal" zusammengefasst und in Kapitel 5.2.3 weiter vorgestellt. Technisch schwierig an dieser Stelle ist die Realisierung der Kommunikation in einer Echtzeit-mäßigen, möglichst dezentralisierten Art und Weise, zusammen verknüpft mit dem *nicht störenden* Aspekt (eine Frage der Präsenz).

Zusammenfassend setzt die zu beschriebene Softwarelösung Konzepte um, die das Einleben der Wohnungsmieter erleichtern können. Anhand eines Kontext-Modells wird die Umgebung der Wohnung gestaltet und mit Gegenstände befüllt, sodass diese später während der Mietphase entdeckt (bzw. lokalisiert) und entwertet werden können. Darüber hinaus wird auch ein Konzept vergegenwärtigt, welches eine zu sozialen Normen konforme, zu beliebigen Zeitpunkten ausführbare Kommunikation zwischen Eigentümer und Mieter ermöglichen könnte. Diese Themen bieten der Ausgangspunkt zur Formulierung einer zentralen Forschungsfrage, welche als nächstes behandelt wird.

1.4 Forschungsfrage

Im Zuge dieses Kapitels wurden die wesentlichen, mit der Hauptthematik dieser Arbeit verwandten Forschungsfelder kurz genannt, zusammen mit denjenigen Problemen, die diese Arbeit versucht anzusprechen. Dadurch entsteht eine Hauptfrage bezüglich der Forschungsrichtung, mit Hilfe derer Konzepte definiert werden, die die betrachteten Anwendungsfälle der Domäne abdecken können.

Die genannten und in Kapitel 2 ausführlich eingegangenen Arbeiten stellen viele Lösungen zu partiellen Problemen vor, die in einer möglichen Lösung der “Home Sharing” Problematik angewandt werden könnten.

Weiterhin lässt sich die Problemstellung des “Home Sharing” Szenarios auch nicht anhand der Funktionalitäten vorhandener Anbieter von “Home Sharing” Systemen vollständig lösen, wie im Absatz 4.1 motiviert. Zu diesem Zweck muss eine generische und erweiterbare Softwarelösung gefunden werden, die bestimmte Kriterien erfüllt. Ein erster Schritt, womit die vorgestellte Angelegenheit einzugrenzen ist, wäre die Formulierung einer Forschungsfrage.

In dieser Arbeit wird der Fokus darauf gesetzt, ein Kontext-Modell zu definieren, welches eine technische Architektur bereitstellt. Die Architektur muss das vorgestellte Home Sharing Szenario sowie weitere “Smart Home”-bezogene Szenarien unterstützen. Die abgeleitete, grundlegende Forschungsfrage lautet folgendermaßen:

Wie muss ein Kontext-Modell gestaltet sein, dass es Erweiterungen und einen Umgang mit Echtzeit-Daten erlaubt, unter Beibehaltung gewisser Eigenschaften (wie die Allgemeinheit, Flexibilität oder ein erhöhtes Sicherheitsgrad) ?

Um eine Antwort auf die gestellte Frage nennen zu können, müssen grobe Konzepte eingeführt werden, die auf den betrachteten User-Stories angewendet werden. Diese führen zu eine Erhebung von Anforderungen an die zu entwickelnde Softwarelösung. Diese breite Thematik bereitet alle notwendigen Bausteine für die letztendliche Gestaltung, Umsetzung und Evaluation der in der Forschungsfrage genannten Aspekte.

1.5 Motivation

In diesem Abschnitt werden die zentralen Konzepte dieser Arbeit vorgestellt, welche aus Sicht der Benutzer, deren Stories und Use Cases, des Entwurfs und Umsetzung weiter benutzt werden.

1.5.1 Unterstützung des Home Sharing

Ein erster Motivationsfaktor dieser Arbeit wird von der Domäne repräsentiert, in der die vorzustellenden Konzepte umgesetzt werden: Home Sharing, welche einer der erfolgreichsten "Branchen" des Sharing Economy (siehe auch 2.4.1) ist. Unter "Home Sharing" wird ein Marktplatz für Wohnungen verstanden. Diese Arbeit schlägt eine mögliche Erweiterung des grundlegenden Szenarios dieser Branche vor – der genauere Konzept dazu wird im Laufe dieses Abschnittes vorgestellt.

Das Kernkonzept der Arbeit umschließt wie erläutert die Vorgänge von Home Sharing, legend den Fokus auf die Mietphase. Dadurch wird das Einleben der neuen Mieter technologisch unterstützt, indem eine mobile Softwarelösung zur Verfügung gestellt wird, die zusammen mit einem Server-System (ein Backend) arbeitet.

Eine mögliche "Definition" des Home Sharing "Erlebnisses" könnte die Folgende sein: die Möglichkeit, eine Wohnung zu suchen und auszuwählen, mit der Zusage des Wohnungseigentümers anreisen, unter Beachtung der vereinbarten Hausregeln und allgemeinen Geschäftsbedingungen die Wohnung vermieten, schließend auschecken und eine Bewertung abgeben. Zusammengefasst wird die Idee hinter dem Home Sharing Erlebnis auch auf der Webseite von Wimdu (ein Home Sharing Dienst): "Wohne wie ein Local, hol Dir Insider Tipps von deinem Gastgeber und erlebe eine Stadt aus einer ganz neuen, individuellen Perspektive". Weitere Details über Home Sharing und die umfassende Domäne namens "Sharing Economy" können aus dem kommenden Absatz 2.4.1 entnommen werden.

Darüber hinaus wird im Paragraph 4.1 eine Analyse durchgeführt, die die Nützlichkeit einer weiteren Unterstützung bzw. Vertiefung oder Erweiterung dieses "Erlebnisses" motivieren kann. Grundlegend für die durchgeführte Analyse war die Etablierung verschiedener Kriterien, die zu untersuchen waren. Die Kriterien werden an dieser Stelle kurz genannt, da in den kommenden Abschnitte Bezug darauf genommen wird:

- Erfassung von tiefgreifender Datenbestände über die Wohnung und deren Umgebung
- Förderung des spontanen Kontakts während des Mietintervalls
- Unterstützung der Mobilität

In den nächsten Abschnitten werden einige Konzepte vorgestellt, die der Konzept des Home Sharing unterstützen und auch erweitern. Ein erster Schritt hierfür ist die Definition eines Kontext-Modells, womit feingranulare Informationen über die vermietete Wohnung erfasst und verwertet werden können.

1.5.2 Definition eines Kontext-Modells

Der in der Analyse verschiedener Anbieter von "Home Sharing" zuerst angewendete Kriterium, *Erfassung von tiefgreifender Datenbestände über die Wohnung und deren Umgebung* braucht zuerst eine fertigzustellene Struktur, die von den eigentlichen Instanzen einzuhalten ist und eventuell auch erweitert werden kann.

Die genannte "Struktur" wird als ein "Modell' in der Terminologie der Softwaretechnik bezeichnet. Weiterhin werden in der Softwaretechnik Techniken der Modellgetriebenen Software-Entwicklung

eingesetzt, um das Wissen einer bestimmten Domäne zu erfassen und in Anwendungen zu benutzen. “Wissen” ist in diesem Fall eine abstrahierte Darstellung von Prozessen und Sachkenntnissen, die die beschriebene Domäne bestimmen. Der Fokus wird in der Modellgetriebenen Entwicklung nicht auf algorithmischen, rechnerischen Sachverhalte gelegt, sondern auf das Erzeugen von Abstraktionen, die die Absicht des Entwurfs auf der konzeptionellen Ebene und nicht mehr auf der technologischen Ebene (z.B. Hardware-Eigenschaften der Ziel Plattform) [Sch06] erfassen.

Aus diesem Grund lässt sich Kontext-Modellierung als eine spezifische Anwendung von Techniken der Modell-getriebenen Software Entwicklung ansehen. In der Domäne der kontextbewussten Anwendungen (weitere Informationen sind in Kapitel 6 zu finden), sind Informationen über die aktuelle Situation oder Aktivität gefragt, die die Anpassung (hinsichtlich der Darstellung oder Operationen) der Anwendung steuern. Um die Entwicklung von kontextbewussten Anwendungen zu erleichtern, ist die Verwaltung solcher Informationen (auch als Kontext-Informationen bekannt) nicht auf der Ebene der Anwendung zu führen, sondern in spezialisierten Software-Komponenten namens Kontext-Modelle [CWGN11]. Der Akzent wird dadurch auf die Wiederverwendbarkeit des Kontext-Modells gelegt – damit verbunden ist eine Kostensenkung im Fall der Entwicklung weiterer kontextbewussten Anwendungen, die dasselbe Modell verwenden.

Jedoch sind mit der Etablierung bzw. Definition eines Kontext-Modells erhebliche Herausforderungen zu beachten. Beispielsweise muss das Modell flexibel genug sein, um statische (vorher modellierte Informationen) mit dynamischen (zur der Ziel-Anwendung entstehenden) Informationen erweitern lassen zu können. Weiterhin sollte das Modell generisch ausgeprägt sein, sodass die modellierten Sachverhalte abstrakt genug sind, um sich für weitere Anwendungsfälle anpassen zu können, aber jedoch auch konkret genug, um eine bestimmte Anwendungsdomäne wie z.B. Home Sharing erfassen zu können. Darüber hinaus muss das Kontext-Modell auch als gemeinsames Kommunikationsschema zwischen die zu entwickelnden Anwendungen (des Wohnungseigentümers sowie die der Mieter) agieren. Die Nützlichkeit des Kontext-Modells stellt sich als ein weiter wichtiges Thema heraus – die Benutzer sollen in der Lage sein, die vorhandenen Informationen abzurufen, zu bearbeiten bzw. zu erweitern und letztendlich auch zu benutzen.

Diese Herausforderungen sind auch zusammen mit den im Laufe des Paragraphs 4.3.4 vorgestellten Anforderungen zu berücksichtigen. Damit wird im Rahmen dieser Arbeit ein Kontext-Modell für die Umsetzung des genannten Kriteriums *Erfassung von tiefgreifender Datenbestände über die Wohnung und deren Umgebung* entwickelt. Dieses zentrale Konzept der Arbeit wird in Kapitel 6 detailliert.

1.5.3 Unterstützung der Navigation durch Lokalisation

Verknüpft mit dem Kriterium *Unterstützung der Mobilität* ist die Entwicklung einer Indoor-Lokalisierung zur Bestimmung der aktuellen Lokation des Benutzers innerhalb eines räumlich (un)begrenzten Raum. Erzielt an dieser Stelle wird eine Anpassung der Anwendung – in diesem Fall, können Elemente des Kontext-Modells vorgeschlagen werden oder automatisch abgerufen bei höheren Wahrscheinlichkeit einer festen Lokalisierung.

Die Motivation dieses Konzeptes wird in Arbeiten wie [SBG99] gegeben – die Lokation, der Ort des Benutzers ist einer der hauptsächlichen Komponente des Kontextes im Rahmen der Mensch-Maschine-Interaktion. Die Verortung dient aus Sicht dieser Arbeit zu der Selektion der Wohnungsinformationen, die zu einem gewissen Zeitpunkt relevant sein könnten.

Das Konzept, wofür eine Indoor-Lokalisierung von Vorteil wäre, ist wie genannt die Zusammenarbeit mit dem Kontext-Modell: zudem können auch räumlich-bezogene Fragen beantwortet werden wie etwa die Frage “Welche Haushaltsgegenstände befinden sich in der Garage?”, sowie “Welche modellierten Geräte sind in der Küche zu finden?” oder beispielsweise “Wo finde ich die nächste U-Bahn Station?”. Die Unterstützung solcher räumlichen Anfragen ergibt sich aus der Möglichkeit, räumliche Daten und Beziehungen in dem Kontext-Modell zu speichern, wie weiter im Abschnitt 6.2.2 beschrieben.

Aus der verschiedener Möglichkeiten zur Indoor-Lokalisierung verwendet die Softwarelösung die vereinfachte Variante der Lokalisierung, mittels ausgedruckter und eingescannter QR-Codes, wie weiter in Kapitel 7 vorgestellt wird. Die Möglichkeit, eine andere Methode zur Lokalisierung zu verwenden, ist vorhanden bzw. unterstützt, wie es im Abschnitt 7.4.3 erläutert wird. Diese Lösung basierend auf QR-Codes wurde aufgrund der vorhersehbar hohen Implementierungs- und Evaluationsaufwand ausgewählt. Eine Möglichkeit zur Outdoor-Lokalisierung lässt sich jedoch im Fall der Kontext-Elementen außerhalb der Wohnung benutzen und verwendet als technologische Grundlage die übliche GPS- Lokalisierung.

1.5.4 Ein Kommunikationskanal für Normal- und Notfälle

Das letzte Kriterium, *Förderung des spontanen Kontakts während des Mietintervalls* wird gelöst durch die Entwicklung eines Kommunikationskanals. Dieser Konzept kapselt die allgemeinen Begriffe aus der Kommunikationstheorie [SW98]: es werden Monologe durch “Offline” Mitteilungen unterstützt, Dialoge in Form eines nah-Echtzeitigen Chats oder eine Konferenz im Fall von mehreren involvierten Parteien (mehrere Wohnungseigentümer oder Wohnungsmieter). Als Sender und Empfänger können sich sowohl der Wohnungseigentümer als auch der Wohnungsmieter beteiligen.

Auf den Kommunikationskanal können beidseitige Interaktionen entstehen wie z.B. eine Echtzeit-Kommunikation zwischen Mieter und Wohnungsbesitzer als auch einseitige Handlungen wie z.B. die Entdeckung neuer Kontext-Informationen von den Mietern und die eventuell damit verbundene Überwachung der Aktivität vom Wohnungsbesitzer. Schon auf dieser Ebene lassen sich Aspekte der Privatsphäre und berechtigte Datenmanagement erkennen, welche eine weitere Herausforderung des Szenarios darstellen und im kommenden Unterabschnitt behandelt werden.

Zugleich mit dem “normalen” Betriebsmodus des Kommunikationskanals, indem allgemeine Mitteilungen verschickt werden können, stellt den Konzept des Kommunikationskanals auch eine weitere Möglichkeit zur Verfügung – eine Art Notfall-Mitteilung, die direkt zu dem Dienstvermittler ankommen können. Eine mögliche Erweiterung an dieser Stelle von Notfall-Mitteilungen wäre die Realisierung einer Verbindung zu einem Notfalldienst. Dieser Konzept übersteigt aber den Aufwand dieser Arbeit und wird nur als eine mögliche Erweiterung angesehen.

Aus der Entwurf- und Entwicklungsseite lässt sich der Kommunikationskanal mithilfe einer Middleware (dt. Diensteschicht), die die Verwaltung der angekommenen, versandten und zu persisitierenden Mitteilungen übernimmt und ausführt. Weitere Informationen über die Umsetzung werden im Paragraph 7.3.4 eingegangen.

1.5.5 Sicherheitskonzept

Datensicherheit spielt im Konzept eine wichtige Rolle, weil persönliche Daten gespeichert werden, wie z.B. die Adresse oder separate Merkmale der Wohnung, sowie eventuelle Daten über Buchungen oder die Profile der Wohnungsmieter. Darüber hinaus erfolgt die Kommunikation zwischen die Client(s) und dem Server in der Wohnung über einen möglich unsicheren Netzwerk (normalerweise eine WLAN Verbindung). Aus diesem Gründen lassen sich folgende Ziele hinsichtlich der Sicherheit der Softwarelösung aus [Sib] identifizieren:

- **Vertraulichkeit**: Übertragene und gespeicherten Daten sollen nur berechtigten Instanzen zugänglich sein. In erster Linie zu bedenken ist an dieser Stelle die sichere Persistierung der Daten.
- **Teilnehmerauthentizität**: Dieses Ziel beschreibt die Gewissheit, dass der Kommunikationspartner tatsächlich derjenige sein sollte, für den er sich ausgibt.
- **Nachrichtenauthentizität (engl. Message Authenticity)**, auch als Datenintegrität bekannt, repräsentiert die Gewissheit, dass es nicht möglich ist, die übertragenen Daten unautorisiert und unbemerkt zu manipulieren.
- **Datenintegrität**: Ein weiteres Sicherheitsziel, der von MACs unterstützt wird, ist die Datenintegrität. Dieses Prinzip beschreibt die erwünschte Situation, in der die übertragenen Daten unautorisiert und unbemerkt nicht manipulierbar sind.

Zusätzlich zu den oben genannten Sicherheitszielen ist ein anderer Konzept der Datensicherheit in dieser Arbeit zu erwähnen: der sogenannte “Data Takeout”-Vorgang. Dieser Vorgang ist eine Initiative von Google und Teil des “Data Liberation”- Projektes[8]. Wie der Name es andeutet, wird durch diesen Prozess der allgemeine Vorgang beschrieben, womit man seine von einem Anbieter gespeicherten Daten “mitnimmt” im Fall eines Anbieterwechsels.

Der Prozess vermeidet die Situation, in der eine “Vorratsdatenspeicherung” entsteht und sich Daten nicht mehr exportieren lassen. Diese Art von Speicherung ist auch als ein “(Vendor) lock-in” bekannt[9]. Mit der Hilfe es Data Liberation Projektes lassen sich allgemein Daten (beispielsweise sind im Fall von Google eine Mehrzahl von Produkten für einen Datenexport bereit) womöglich in einem öffentlichen Format exportieren[10].

Zusammenfassend lässt sich erwähnen, dass die Sicherheitspolitik jedes Produktes zum großenteils den Erfolg eines Konzeptes bestimmen kann. In der Regel besteht die beste Praxis daraus, die wichtigsten Merkmale des Sicherheitskonzeptes in Form einer Datenschützerklärung oder in Form von AGBs zu veröffentlichen. Die Umsetzung der vorgetragenen Sicherheitskonzepte wird in Kapitel 7 in Detail eingegangen.

1.6 Struktur der Arbeit

Diese Arbeit ist wie folgt organisiert: in Kapitel 2 werden die Arbeiten vorgestellt, die zur Entwicklung der vorgestellten Konzepte und Szenarien eine unterstützende Rolle spielen. Weiter werden

[8] `http://www.dataliberation.org`
[9] Vgl. “Linux Information” Projekt: `http://www.linfo.org/vendor_lockin.html`
[10] `http://www.dataliberation.org/home/faq`

in Kapitel 3 mögliche User-Stories vorgestellt, die im Freitext die zu untersuchenden Anwendungsfälle beschreiben. Darauffolgend werden im Laufe des Kapitels 4 die auf User-Stories aufbauenden Anforderungen erhoben bzw. definiert.

In der Folge lässt sich der zugrundeliegende Entwurf zur Durchführung der Anforderungen in Kapitel 5 vorstellen, aus Sicht der visuellen und softwaretechnischen Architektur. Der zentrale Konzept der Arbeit wird dementsprechend näher in Kapitel 6 und mit einbezogen in der Vorstellung der prototypischen Umsetzung aus Kapitel 7. Die Evaluation des entwickelten Prototyps erfolgt in Kapitel 8, während in Kapitel 9 das Fazit gezogen wird und einen Ausblick gegeben wird.

2 Verwandte Arbeiten

Im Rahmen dieses Kapitels werden die verwandten und verwendeten Arbeiten, die zu der Erstellung dieser Arbeit beigetragen haben, beschrieben. Der Bezug zu jeder genannten Domäne wird kurz erläutert und danach werden einzelne Werke ausgezeichnet, um den Zusammenhang oder Beitrag bzw. Inspiration zu dieser Arbeit erklären zu können.

2.1 Ubiquitous Computing

Mark Weiser hat den Begriff "Ubiquitous Computing" (UbiComp) zum ersten Mal im Jahr 1991 geprägt durch seine einflussreiche Arbeit "The Computer of the 21^{st} Century" [Wei91]. Das Konzept beschreibt eine mögliche Zukunft, in der allgegenwärtigen Rechnersysteme den Alltag der Menschen unterstützen, sei es in der eigenen Wohnung oder bei der Arbeit. Es entsteht dadurch eine kontinuierliche Interaktion des Benutzers mit vielen, drahtlos vernetzten Rechnersystemen [HCH+11]. Eine andere Perspektive wird in [DRAPZ04] gegeben: die hauptsächliche Zielsetzung von Ubiquitous Computing ist die Untersützung oder "Erweiterung" (engl. augmentation) einer vom Benutzer geführten Aktivität in einer möglichst unsichtbare, unauffällige Art und Weise.

Diese Rechnersysteme würden im Hintergrund in einer unsichtbaren, unauffällige Weise arbeiten. Die Menschen würden dadurch von der Ausführung verschiedener mühsamen Routinearbeiten befreit. Die in [KK06] zitierte und verbreitete Definition lautet wie folgt: unter ubiquitären Computing versteht man jede Computing Technologie, die eine menschliche Interaktion außerhalb einem einzigen Arbeitsplatz bzw. Workstation erlaubt. Beispiele für Technologien, die für ubiquitär gehalten werden können, wären: Stift-basierte Technologie, Handgeräte oder mobile Geräten, zusammen mit großangelegten, interaktiven Displays, sowie Stimme- und Vision-basierte Technologien.

Folglich lässt sich an dieser Stelle anmerken, dass Ubiquitous Computing auch die Problematik der Informationsüberflutung anspricht [HCH+11]: die oben genannten Rechnersysteme müssen sich zu vorhandenen menschlichen Umgebungen anpassen, anstatt die Menschen zu erzwingen, die "rechnerische Welt" anzutreten. Damit müssen im Wesentlichen anpassungsfähige Interaktionen bzw. Darstellungen entwickelt werden, die sachkundig dem Benutzer im Alltag unterstützen.

In ihrer "ultimativen" Form malt die Vision von ubiquitären Computing jedes Gerät aus, das Rechenarbeiten durchführen kann, währenddessen der Nutzer mobil ist. Eine weitere wichtige Eigenschaft dieser Vision ist die Tatsache, dass das mobile Gerät schrittweise ein dynamisches Modell der ständig verändernden Umwelt aufbauen kann. Darüber hinaus mithilfe vorhandener "Intelligenz" und Inferenz-Fähigkeiten kann es sein eigenes Leistungsangebot bezüglich der aktuellen Situation anpassen. Damit würden Geräte in der Lage sein, vorherig modellierte Umgebungen zu erkennen oder proaktiv vorhandene Dienste auf neuen Umgebungen einstellen zu können.

Zusammenfassend repräsentiert ubiquitäres Computing annähernd den Gegenteil der virtuellen Realität. Wohingegen die virtuelle Realität Menschen in einer neu erschaffenen Welt integriert, beschreibt Ubiquitous Computing das Szenario, in dem die Rechensystemen zur Aufbesserung in die wahrhafte Welt eingebettet werden. Diese Vision hat sich in den Grundlagen verwandter Domänen wie Ambient Intelligence oder Internet of Things ausgeprägt. Zugleich bestimmen die Ideen dieser zentralen Domäne in hohem Maße die grundlegenden Konzepte dieser Arbeit.

2.2 Ambient Intelligence (AmI)

Ambient Intelligence stellt sich vor als ein Konvergenzpunkt mehrerer Ideen aus dem Ubiquitous Computing und repräsentiert zugleich der Basis für Ambient Assisted Living. In [KK06] wird der Zusammenhang der verschiedenen Bereichen oder Teilkonzepte aus Ambient Intelligence (AmI) erläutert: der Konzept von AmI wird durch die Bereitstellung von ubiquitären Computing (s.u.), ubiquitären Kommunikation (ein universeller, standortsunabhängiger Zugang zu einer Computing- und Netzwerkinfrastruktur) sowie der Einsatz intelligenter, anpassungsfähiger Benutzersoberflächen (engl. intelligent, user-adaptive Interfaces) realisiert.

Die häufig verwendeten Begriffe “ambient” sowie “ubiquitär” lassen sich anhand von [KK06] wie folgt erklären: im Fall von “ambient” oder dt. “umgebend” gilt die folgende Wörterbuch-Definition – “aus allen Seiten vorhanden” (engl. “existing or present on all sides”). Die Idee hinter “Ubiquität” oder “Allgegenwart” beschreibt die Existenz eines oder derselben Gegenstandes überall innerhalb eines definiertes Raumes, wie z.B. die Existenz verschiedener Sensoren in einer Smart Home Umgebung.

In [Cur11] wird motiviert, dass die Konvergenz von Informationen, Kommunikationen und Netzwerktechniken ein gemeinsames Framework etabliert haben, die die Einführung von umgebenden (engl. ambient), ubiquitären Computing ermöglicht. Die Vision von Ambient Intelligence hat das Potential, eine Revolution anzutreiben bezüglich der Art und Weise, indem diese Computing Dienstleistungen angeboten werden. Dieses Angebot ist umfassend und basiert auf die idee von intelligenten Umgebungen: alltägliche Wohnflächen, in denen Unterstützungstechnologie eingebaut wird. Die Zielsetzung ist es, diejenige Technologien einzusetzen, die für die Benutzerzielgruppe relevant und möglichst sehr hilfreich im Alltag sind.

Eine wichtige Anforderung an diesem Szenario ist der dauernde Zugriff auf präzise Informationen, ohne Berücksichtigung des Standortes innerhalb der intelligenten Umgebung. Deshalb sind Systeme der Ambient Intelligence anpassungsfähig und kontextbewusst – um die Kontext-abhängig relevanten Informationen in einer personalisierten Form liefern zu können. Damit wird es den Benutzer ermöglicht, gut informierte Entscheidungen in einer dringenden Zeitspanne zu treffen. Die Individualisierung der Lösung bestimmt die genannte Eigenschaft der Relevanz: die “intelligente” Technologie erkennt die Präsenz der Menschen und sollte die individuellen Präferenzen ihrer Benutzer erkennen und ausnutzen im Laufe der Anpassung. Weitere intelligente Merkmale wären die Unterstützung der natürlichen Sprache und die damit verbundene Formulierung von intelligenten Antworten und damit eine Dialogfähigkeit.

Die “intelligente” Facette der Ambient Intelligence Systeme stammt aus dem Forschungsfeld der Intelligenten Systeme [Sha03], wo fortgeschrittene Algorithmen rund um maschinelles Lernen und Muster-Abgleich, Stimme-Erkennung oder Gesten-Klassifizierung sowie die Bewertung von Situationen (Situation-Bewusstsein) entwickelt werden.

Weiterhin werden spezifische Eigenschaften der Mensch-Maschine -Interaktion durch Ambient Intelligence erweitert: die Wechselwirkung und Benutzung der Technologie sollte mit minimalen Aufwand erfolgen; weiterhin muss die Interaktion leicht verständlich sein und damit zu einer insgesamt angenehmen Erfahrung führen.

Zusammenfassend bietet Ambient Intelligence eine Vielfalt von einfallsreichen Möglichkeiten an, Bequemlichkeit für die Benutzer des Systems zu erzeugen. Es werden Lösungen vergegenwärtigt, die über den aktuellen Stand der Technik hinausgehen – wie z.B. die Verknüpfung des Cloud Computing

Paradigmas in einen Ambient Intelligence Kontext, wie in [LLCS10] beschrieben. Weitere Szenarien sind in der ersten Arbeit über Ambient Intelligence [DBS+01] enthalten und in [Sha03] kurz zusammengefasst.

Aus der erkundeten Werke ist der Konzept von Ambient Intelligence zu entnehmen, insbesondere die erwünschte Merkmale wie leicht bedienbar, "intelligent" sowie das insgesamt angenehme User Experience, die im Mittelpunkt der Gestaltung aller Benutzungoberflächen stehen sollte. Diese Prinzipien wurden beachtet bei der Definition des gesamten User Experience Konzeptes, die in 5.4 vorgestellt wird.

2.2.1 Smart Objects

Das im Rahmen der "Ambient Intelligence" eingeführten Paradigma, nämlich die Einbettung von minimalen Rechnersystemen in der Umgebung kennt eine Ausprägung durch den Forschungsfeld namens "Smart Objects". Gemäß dieser Vision werden alltägliche Objekte wie z.B. ein Spiegel, welcher personalisierte Nachrichten oder den aktuell gemessenen Energieverbrauch des Hauses darstellt mit "Intelligenz" anreichert [KLN08].

Dieses Konzept nähert die Prinzipien der verwandten Forschungsdomäne namens "Internet of Things" (siehe auch 2.6), jedenfalls mit dem Unterschied, dass die "Intelligenz" normalerweise nicht direkt im Objekt vorhanden ist – die meiste Programm-Logik, die Entscheidungstreffen sowie die Wahrnehmung der Umgebung ist auf einem zentralen System (die Infrastruktur) gelagert [KLN08]. Das Objekt selber dient zur Identifikation, Verfolgung und Teilung (s.u.): ein Beispiel eines Smart Objects sind einzelne Produktbestandteile im Kontext der allgemeinen Fertigung, die mit RFID (Radio-Frequency Identification. Ein Tag ermöglicht die automatische Identifizierung und Lokalisierung von Gegenständen und erleichtert damit die Erfassung von Daten) Tags vorgesehen werden. Weiterhin lassen sich diese Smart Objects entlang des Lieferantenkettenmanagements (eng. Supply Chain Management, SCM) benutzen bzw. anhand dem kosteneffizienten RFID Tag verfolgen [KLN08].

Weiterhin können Smart Objects sich in Systeme organisieren, aufbauend ein "Smart Object System". In diesem Szenario kommuniziert und arbeitet ein Smart Object zusammen mit anderen "Agenten" aus der räumlich begrenzten Umgebung, um eine spezifische Aufgabe zu lösen. Anhand dieser Organisation der Objekte in einem Netz können Aufgaben auch zentral von einem Server-System gelöst bzw. koordiniert werden, ohne dass die einzelnen Objekte von ihrer benachbarten Gleichartigen (engl. peers) wissen müssen.

Zusammenfassend stellen Smart Objects eine weitere Möglichkeit, eine Umgebung mit digitalen Informationen in einer meist kostengünstige Weise anzureichern, um ein bestimmtes Ziel zu erreichen. Mithilfe bestehender Technologien wie das Web, die Nachwuchs-Technologie zur Near-Field Communication (NFC) sowie Techniken der Indoor-Lokalisierung kommen Visionen wie Smart Objects näher zum verbreiteten kommerziellen Einführung. Bis zu dem Meilenstein bleiben jedoch verschiedene Fragen, die adressiert werden sollten, wie in [KLN08] ausführlich beschrieben – z.B. die Erstellung von Smart Objects, die ihre "Intelligenz" (engl. Smart Features) für mehrere Gegenstände anwenden können und nicht nur diese Fähigkeiten in einem Objekt einbetten. Darüber hinaus bleibt besteht eine Herausforderung darin, die Smart Features eines angereichertes Objekt mittels Anwendungen (Apps) zu Nutze zu machen.

Die aus der “Smart Home” Welt bekannte Problematik der fehlenden standardisierten Schnittstellen (siehe auch 2.4) tritt in diesem Fall auch auf: eine Interoperabilität zwischen Smart Products Lösungen wird dadurch aufgehalten, da es keine gemeinsame Infrastruktur gibt, die Produkte vom verschiedenen Anbieter und verschiedene Wirtschaftssektoren zusammenbringt [RQBA09]. Damit kann die Vision eines einheitlichen “Ökosystemes”, wo jede entwickelte Technologie zusätzlichen Nutzen für existierende Geräte und Dienste mit sich bringt, nicht erfüllt werden.

Dieses Konzept eines Objektes, das zusammen mit “Smart Features” und eine entsprechende technische Architektur für die Bereitstellung von “Intelligenz” als ein “Smart Object” agieren zu können, wurde in dieser Arbeit konzeptuell umgesetzt. Der entsprechende Beitrag befindet sich in dem erstellten Datenmodell für Home Sharing, im Paragraph 7.2.2.

2.3 Ambient Assisted Living (AAL)

Die Informationsgesellschaft entwickelt sich in eine neue Richtung, in der sich die Computing Technologie von dem etablierten Paradigma des festen Arbeitsplatz entfernt [BA09]. Das neue “Post-PC” Paradigma umfasst verteilte Systeme, die mit einem breiten Spektrum von Geräten, Vorrichtungen und Sensoren vernetzt sind und dadurch Zugang zu mehreren Diensten anbieten. Die betroffenen Domänen sind nahezu universell, da die neuen Systeme in verschiedenen physischen Orte eingesetzt werden können, inklusive Häuser, Büros, Autos oder öffentliche Bereiche. Durch die verschiedenen Möglichkeiten der Aufstellung wird den Konzept von Ambient Intelligence (s.u.) verknüpft mit der Idee, Anwendungen zu entwickeln, die zu der Umgebung, den Benutzer und deren Interaktionsmodi maßgeschneidert ist [BA09].

Eine mögliche Definition von Ambient Assisted Living wird ebenso in [BA09] gegeben: ein Smart Home Umgebung, die die Unterstützung alltäglicher Aktivitäten der Benutzer-Zielgruppe durch den Einsatz von Technologie. In Linie zu der Smart Home Grundidee, wird die Umgebungsintelligenz (engl. Ambient Intelligence) in der Umgebung eingebettet. Die Zielsetzung von Ambient Assisted Living ist, die Benutzer bzw. Bewohner bei deren Aktionen und Pläne mehr zu helfen mithilfe von Technologie – dadurch werden alle Zielgruppe von Benutzer angesprochen. Trotzdem wird normalerweise im Umfeld von Ambient Assisted Living (AAL) den aktuellen Fokus darauf gelegt, die Technologie für die Hilfe von Senioren und Menschen mit Behinderung einzusetzen.

Ein letzter wichtiger Punkt aus [BA09] ist der folgende: in den Feldern von AAL, Ambient Intelligence und Smart Home Umgebungen ist ein Thema immer noch von Aktualität – zwar wird versucht, eine gemeinsame Architektur und Framework für AAL Umgebungen zu entwickeln. Die zentrale Herausforderung dabei stellt die Integration von Wohnungssteuerung, Sensoreninformationen sowie Modalitäten der Interaktion dar. Dieser Aspekt der mangelhaften vollständigen Interoperabilität zwischen (AAL) Lösungen wird in [LLCS10] auch bestätigt.

Weiterhin kann die Entwicklung und Laufzeitmanagement der multimodalen Interaktionen in den oben genannten Felder erheblich erleichtert werden durch die Benutzung von modellbasierten Entwicklungsmethoden [BA09]. Im Fall dieser Anwendung wird eine kleinmaßstäbliche Integration der oben aufgeführten Komponenten erzielt, deren Entwicklung modellbasiert ist, wie vorher vorgeschlagen.

Eine weiter inspirierende Arbeit ist in [ALGMVM10] zu finden: hier wird eine Möglichkeit der technischen Infrastruktur einer Anwendung im AAL / Ambient Intelligence Felder vorgestellt. In der

Arbeit wird eine technologische Plattform beschrieben, die Service-gesteuert ist. Ein Service in der Arbeit wird durch ein RESTful Web Service realisiert, das eine Beschreibung der von ihm ausimplementierten Schnittstellen bereitstellt (eine Art Service Description) und sich ins "Verzeichnis" der verwendeten Service Discovery einträgt.

Diese Services lassen sich zusammensetzen, indem eine Definition der anderen beteiligten Services vorgeschrieben wird, zusammen mit einer Beschreibung des Datenablaufs zwischen den individuellen Bestandteilen. Weiterhin wird in der Arbeit der subtiler Aspekt der Privatsphäre betont: die meisten Kritiker des AAL-Umfeldes betonen die Wichtigkeit dieses Themas und weisen Vorwürfe der Mangelhaft auf. In der Arbeit wird die feine Abwägung zwischen die Erfüllung der Zielsetzung (Hilfe und Unterstützung leisten) und die Achtung auf der Privatsphäre. Auf diesem Aspekt wird auch in dieser Arbeit durch die im Absatz 5.2.2 vorgestellten Sicherheitskonzepte geachtet.

2.4 Smart Home

Der Domäne von "Smart Homes" repräsentiert eine Ausprägung, eine Anwendung der Konzepte, die im Paragraph 2.2 beschrieben worden sind. Grob gesehen ist eine "Smart Home" Umgebung eine Umgebung, die mit umgebender Intelligenz (Ambient Intelligence) bereichert worden ist. Die "Intelligenz" in diesem Fall besteht beispielsweise aus Microcontroller-Einheiten und drahtlose Sende-Empfängsgeräten (engl. transceiver), die zusammen mit Netzwerktechnik zu einer lokalen, drahtlosen Netzwerk aufbauen. Die Netzwerk wird stets als ein Wireless Sensor Network (WSN) bezeichnet. Eine äquivalente Definition wird in [LRBA10] gegeben, wobei die alternative Bezeichnung für Smart Home Technologien auch erläutert ist – Smart Environments oder Smart Spaces [PGS+11].

Der Fokus einer Anwendung, die in einer Smart Home Umgebung eingesetzt ist, spiegelt, wie im Fall von Ambient Assisted Living (siehe auch 2.3 die Unterstützung der Bewohner im Alltag wieder. Die Unterstützung erfolgt anhand mehrerer wahrgenommenen Umgebungsparameter und "findet einen Ausprägung" z.B. in einer graphischen Darstellung einer vorgeschlagenen Lösung zu der ständig ändernden Aktivität bzw. Situation des Benutzers [LRBA10].

Ein annähernder Blickwinkel der Funktionalität wird in [PGS+11] dargestellt: ubiquitäre Anwendungen im Rahmen innerhalb von "Smart Spaces" benutzen die lokal vorhandenen Verbindungen und die Entdeckung (engl. discovery) anderer lokalen Geräten, um Inhalte und Dienste zwischen den vernetzten Geräte auszutauschen.

Diese vereinfachte Beschreibung von "Smart Home" Technologien, die als eine Umsetzung der Ambient Intelligence Konzepte angesehen werden kann, malt jedoch einen idealisierten Anwendungsfall aus. In der Tat stellt sich die Umsetzung bzw. die Installation und Bereitstellung einer Smart Home Technologie in eine Wohnung als eine wesentlich schwierigere Aufgabe heraus. In [RQBA09] wird motiviert, dass eine der zentralen Herausforderungen von Smart Home Technologien war und ist immer noch der Mangel an einer einheitlichen Entwicklung von Produkten und Dienstleistungen, die auf gemeinsame Standards und Protokolle basieren.

Darüber hinaus wird dasselbe Argument auch von [PGS+11] geliefert: während die Anzahl von Forschungsprojekte, die das Potential von Smart Home Technologien vorstellen, hoch ist, sind die Ergebnisse dieser Recherchen nicht weitgehend eingesetzt worden. Der wesentliche Grund dafür ist die mangelhafte Zusammenarbeitsfähigkeit: die Anwendungen können einwandfrei in einer kontrollierten Smart Home Umgebung laufen, wo der Typ und Anzahl der verschiedenen Geräte begrenzt ist.

Wenn aber in der Praxis hunderte von verschiedenen Geräte verwendet werden, die auf verschiedenen Betriebs- und Laufzeitumgebungen ausgeführt werden, verbleibt das entwickelte System größenteils nicht völlig funktionsfähig.

Diese Problematik ist auch in der übergeordneten Domäne von Ubiquitous Computing bekannt – eine zitierte Studie aus derselben Arbeit [PGS+11] benennt im Jahr 2004 ebensoviel wie 29 unterschiedliche Plattformen für die Entwicklung von Ubiquitous Computing Anwendungen. Dieser Phänomen wurde in [DRAPZ04] beschrieben als nur die Handhabung verschiedener "Inseln" der Funktionalität aber noch keine richtige, integrierte Lösung.

Trotz dieser Schwierigkeiten werden in der letzten Zeit vorhandene Prinzipien der Domäne "Web of Things" für Smart Home Technologien wiederverwendet: in [KTP11] und [PGS+11] wird über die Entwicklung von Web-orientierten Smart Home Technologien berichtet.

Die Einführung Web-basierter Technologien in die "Smart Home" Domäne kommt als eine mögliche Lösung zu der vorher genannten Problematik der Interoperabilität: durch einheitliche, standardisierte Schnittstellen wie z.B. HTTP, XML können viele heterogene Systeme miteinander Informationen austauschen. Durch den Einsatz von Sende-Empfangsgeräte können über Web Technologien komplexere Systemen aufgebaut werden [PGS+11] – auch wenn nicht alle Geräte ein Dienst anbieten, können sie die anderen Dienste aus der Smart Home bedienen, indem sie Inhalt und Kontext aus der lokalen Perspektive abliefern.

Weiterhin sind in der genannten Arbeit andere Ideen der internetfähigen Smart Home vorhanden: die Vereinigung lokaler bzw. "Intranet zugänglicher" Diensten mit "Extranet" bzw. Internet / Web verfügbaren Diensten. Ein Beispiel aus [PGS+11] nennt zum einen die Sammlung verschiedener lokal vorhandenen, mit geographischen Koordinaten angemerkten Fotos und zum anderen die Darstellung von damit verbundenen Landkarten über eine Internet-Verbindung.

Darüber hinaus wird in [KTP11] belegt, dass eine Evolution möglicher Web-basierten Lösungen im Rahmen von Smart Homes zu bemerken ist: während die ersten Lösungskonzepte SOAP-basierte Web Services benutzten, steht für die aktuelle Entwicklung die Benutzung von RESTful Web Services (siehe auch 2.7) im Mittelpunkt. Die Einbindung von REST Technologien wird in der oben genannten Arbeit weitergeführt, indem separate Geräten mit IP Technologie eingebettet werden – mittels IEEE Standards wie 6LoWPAN ("IPv6 over Low power Wireless Personal Area Networks") und HTTP Zwischenspeicherung (engl. caching) wird es ermöglicht, HTTP Web Servers auf jeden Gerät laufen zu lassen.

Damit wird die Verbindung zu den Konzepten dieser Arbeit hergestellt – die Anwendung von Web-basierten Technologien wie REST im Rahmen des "(Smart) Home Sharing" Szenarios kann auch *direkt* über die individuellen im Heimnetzwerk vorhanden Geräten erfolgen (jeweils mit einer zusätzlichen Aufwand). Dem Extrapolationsverfahren zufolge, lassen sich dann beide Szenarien behandeln – sowohl der Umgang mit der Sensorik eines Smartphones als auch mit der eingebetteten Sensorik. Das zu entwickelnde Kontext-Modell muss daher diese Anforderung unterstützen. Weiteres hinsichtlich der Anforderungsdefinition lassen sich aus dem nächsten Kapitel entnehmen.

Ein weiterer Aspekt der Domäne "Smart Home" ist die im Rahmen dieser Arbeit "eingeführte" Sub-Domäne namens "(Smart) Home Sharing", welche im Folgenden behandelt wird.

2.4.1 (Smart) Home Sharing als zentrale Ausprägung der Sharing Economy Konzepte

Die "Sharing Economy" oder "Collaborative Consumption" Bewegung verwendet das Diktum "Nutzen statt Besitzen", wie aus der Studie [SSbO10] bekanntgegeben und hat sich in den letzten Jahren als ein neues, sehr lukratives Geschäftsmodell entwickelt. Der Unterschied zu einem traditionellen Modell wird in der Einleitung des viel beachteten Buches "The Mesh" [Gan10] erläutert: während im Fall eines "herkömmlichen" Geschäfts ein Produkt oder Dienstleistung erstellt, vermarktet bzw. verkauft und damit hoffentlich auch ein Gewinn sammelt, steht das Sharing Economy den Kunden mehrere Alternativen, Werkzeuge sowie mehr Peer-to-Peer Kraft, um die Vermarktung und letztendlich Nutzung des angebotenen Produktes zu optimieren.

Die Optimierung in dieser Sinne kommt aus zwei grundlegenden Betrachtungen: verfügbare Peer-to-Peer Plattformen wie Social Media, Soziale Netzwerken werden ausgenutzt, um die Vermarktung anzutreiben, um Waren und Dienste perspektivischer Kunden präzis anzubieten und zwar zu *dem* Zeitpunkt, in dem sie es auch brauchen, ohne die Belastung und Kosten die durch ein Besitz der Waren oder Diensten verursacht würde. Eine Umformulierung dieser Idee ist auch in einer Studie beantragt von Airbnb[1] zu finden:

"Sei es der Kauf von privat zu privat, die Buchung einer Privatunterkunft für den Urlaub oder die gemeinsame Nutzung von selten gebrauchten Gegenständen... Insbesondere die jüngere Generation hat die Vorteile einer Ökonomie des Teilens wiederentdeckt und belebt sie dank Internettechnologie neu. Hier liegt großes Potenzial für eine neue Nachhaltigkeit, die auch politisch und gesellschaftlich unterstützt werden sollte"

Der Aspekt der Nachhaltigkeit wird in [SSbO10] auch notiert:

"Wenn sich mehrere Personen ein Auto oder einen Rasenmäher teilen und dieses bzw. diesen nicht mehr selbst besitzen, braucht man weniger Autos bzw. Rasenmäher und damit weniger natürliche Ressourcen , um dieselbe Menge an Personenkilometern bzw. dieselbe Fläche an gemähtem Rasen "herzustellen" "

Ohne weiter eingehen zu müssen, lassen sich die Vorteile des Sharing Economy zusammenfassen: es wird gestrebt, eine neue ökonomische Bewegung zu erstellen, die eine "nachhaltigere Verbraucherherrschaft" erschafft. Dafür entstehen die sogenannten "Peer-to-Peer-Marktplätze", welche auf soziale Prinzipien wie gegenseitiges Vertrauen, Ansehen und einen Gemeinschaftsgedank basieren, indem die Kunden sich gegenseitig helfen, weniger zu kaufen und mehr zu benutzen [Gan10].

Büro- und Veranstaltungsräume[2], Autos[3] sind zu mieten, sogar Versicherte können sich zu einem Netzwerk schließen und gemeinsam bei Versicherungen sparen[4] – diese Beispiele zeigen nur einen kleinen Teil der aktuellen Breite des Konzeptes. Eine systematisierte Behandlung verschiedener Richtungen, in denen sich Sharing Economy bewegt (z.B. "Product Service Systems" oder "Communal Economies") wird in [BR10] gegeben.

[1] Pressemitteilung "Deutschland teilt! Auf dem Weg in eine neue Konsumkultur", Erscheinungsdatum: 19. Juni 2012, http://www.enorm-magazin.de/blog/2012/06/19/studie-deutschland-teilt-bestatigt-trend-des-teilens-in-deutschland/

[2] http://www.deskwanted.com/

[3] http://www.car2go.com/

[4] http://www.friendsurance.de

2.5 Indoor-Lokalisierung

Der im Paragraph 1.5.3 genannte Vorgang der Lokation-Bestimmung wird in der Fachsprache als Lokalisierung bekannt. Lokalisierung stammt aus der ersten Arbeiten in der Domäne von ubiquitären Computing [POM12]: somit nennen die ersten Definitionen von Kontext [DA00] die Lokalisierung bzw. die Lokation eines Benutzers als einer der grundlegenden Bestandteile des (Benutzer-)Kontextes, neben Zeit, Aktivität und Identität. Eine Lokalisierung kann außen / Outdoor oder innen / Indoor ausgeführt werden.

Im Raum der Außen-Lokalisierung gilt die GPS-Technologie als ein de facto Standard und eine fast universell einsetzbare Technik. An dieser Stelle lässt sich sagen, dass das Problem von Outdoor-Lokalisierung fast keinen Forschungsaufwand mehr bräuchte [POM12], da die Ergebnisse der Technik, in Zusammenarbeit mit Wi-Fi oder Mobilfunknetze, hohen Qualitätsansprüchen entsprechen. Trotzdem ermöglicht die Forschung in der Outdoor-Lokalisierung den Denkansatz, die Ergebnisse aus der Outdoor zu der Indoor-Welt zu übertragen. Jeweils kann die Übertragung nicht automatisch erfolgen, weil die Indoor-Umgebungen erhebliche Unterschiede im Fall der Signalausbreitung präsentieren.

Eine Übersicht der verschiedenen Indoor- und Outdoor- Lokalisierungstechnologien, bezüglich ihrer Reichweite und ihrer Genauigkeit wird in [Rai11] gegeben:

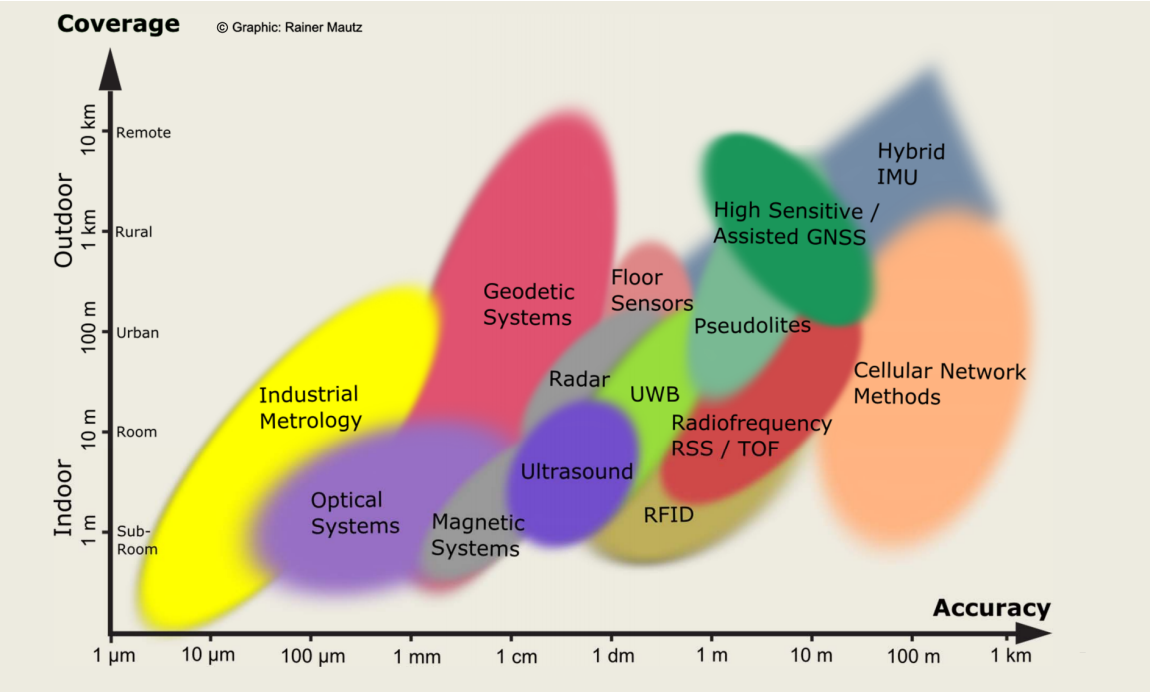


Abbildung 2.1: Überblick der verschiedenen Technologien der Indoor- und Outdoor-Lokalisierung. Quelle: [Rai11]

Aus der Abbildung 2.1 ist die Vielfalt der einsetzbaren Technologien zu entnehmen. Jedoch verknüpft mit der Auswahl einer bestimmten Umsetzung sind dazu Kriterien wie Kosten, benötigte Infrastruktur, Verfügbarkeit und Marktreife oder Geschwindigkeit (Anzahl und eventuell auch Format der

Ausgaben) zu beachten [Rai11]. Weitere Aussagen aus [Rai11] deuten darauf hin, dass ein systematische Evaluationsschema zur Zeit nicht vorhanden sei – die Gründe dafür sind die unterschiedlichen benutzten Technologien sowie die hohe Anzahl von Metriken oder Kriterien, die eine Technologie beschreiben können. Trotzdem geben neue in [Nor12] vorgestellten Ergebnisse Hoffnung, dass eine Evaluationsschema in der Tat erzielbar wäre.

Ein letzter wichtiger Punkt aus [Rai11] ist, dass die Einführung von Standards in dem Gebiet der Lokalisierung sehr hilfreich wäre: ein IS (International Organization for Standardization) oder IEC (International Electrotechnical Commission) Standard würde dazu führen, dass die verschiedenen genannten Methoden einheitliche Konzepte benutzen würden, sowie einheitliche Ausgaben zurückliefern würden. Zum Beispiel könnte das GUM (The Guide to the Expression of Uncertainty in Measurement, ISO/IEC Guide 98-3:2008) Dokument benutzt werden, um die Angaben der Messunsicherheit normen zu können. Initiativen in diese Richtung sind jedoch zu bemerken: ein Wettbewerb[5] namens "Evaluating AAL Systems through Competitive Benchmarking" hat als Zielsetzung die Evaluation von AAL (s.o.) Lösungen und ein Kriterium für die Evaluation ist die Performanz der benutzten Technik zur Indoor-Lokalisierung.

Der möglicherweise letzte Stand der Methoden zur Indoor-Lokalisierung befindet sich in Arbeiten wie in [POM12] oder [MPOMI10], wo die benutzte Techniken FM-Funkstationen als technische Basis haben. Weiter sind in [KSY12] und [Gei11] WLAN-Techniken eingesetzt, während in [WHZM12] verschiedene Strategien zur Lokalisierung für die nah am Körper Anwendung von RFID Tags detailliert wird.

Verknüpft mit der Verwendung der aktuellen Infrastruktur (WLAN, RFID, FM. etc) ist die Erstellung spezialisierten Sensorik zur Indoor-Lokalisierung, wie in [EKLW11] beschrieben. In diesen Aspekten lassen sich die Ergebnisse mehrerer Sensoren für verbesserte Präzision zusammensetzen. Weitere Ideen aus der genannten Arbeit gehen um Pfad-Prognose Algorithmen (die Feststellung mit gewisser Wahrscheinlichkeit, dass sich die künftige Lokation des Benutzers in einem bestimmten Wertebereich befinden wird). Eine sehr ausführliche Beschreibung vorhandener Technologien, inklusive kommerzielle Lösungen ist auch in [CFL+09] zu finden.

Weiterhin wird eine mögliche Revolution in dieser Feld wird durch die Einführung des "Indoor Atlas" Systems [Ind12] geschafft. Indoor Atlas verwendet eine Magnetfeld-basierte Technologie und eine mobile Anwendung für sehr präzise Messungen und Bestimmungen der Indoor Lokation in voraussichtlich Industrie-bezogene Szenarien. Nennenswert an dieser Stelle ist auch der Mangel an open-source Projekte bezüglich Indoor-Lokalisierung – jedoch wird durch Redpin[6] eine Lösung vorgeschlagen, die zumindest eine gute Präzision an der Zimmer-Ebene liefern kann.

Aus diesem Abschnitt zu entnehmen ist die technische Komplexität, die mit der Entwicklung einer Indoor-Lokalisierung verbunden ist. Aus dieser Perspektive wurde im Rahmen dieser Arbeit auf die Verwendung einer aufwendigen bzw. hochpräzisen Technik zur Indoor-Lokalisierung verzichtet, wie im Abschnitt 1.5.3 erläutert.

[5] `http://evaal.aaloa.org/`
[6] "Indoor Positioning for the Rest of Us", `http://www.redpin.org/`

2.6 Internet of Things

Laut [UHM11] ist das “Internet of Things” (IoT) ein Konzept, in dem die virtuelle Welt der Informations- und Kommunikations-Technologie nahtlos mit der realen Welt der “Sachen” eingebunden wird. Dadurch wird die reale Welt zugänglicher für Rechnersysteme sowie vernetzten Geräte, aus der Perspektive verschiedener Betriebs- und Alltagsszenarien.

An sich lässt sich den Begriff “Internet of Things” nur schwierig definieren, da es in verschiedenen Vermarktung und Vertrieb-bezogenen Publikationen als ein Modewort missbraucht worden ist [UHM11]. Außerdem wird in [KCA+11] motiviert, dass die Einführung zahlreichen Arbeiten aus der IoT Domäne normalerweise mit einer Umformulierung verschiedener weit angenommenen Definitionen beginnt. Trotzdem wurde 2009 vom europäischen IoT Forschungsprojekt (bzw. “Strategic Research Agenda of the Cluster of European Research Projects on the Internet of Things”, CERP-IoT 2009) eine genauere Begriffserklärung bekannt gemacht (hier angepasst):

A dynamic global network infrastructure with self configuring capabilities based on Standard and interoperable Communication Protocols, where Physical and Virtual “things” have identities, physical attributes, virtual personalities and use intelligent interfaces, whilst being seamlessly integrated into the information network.

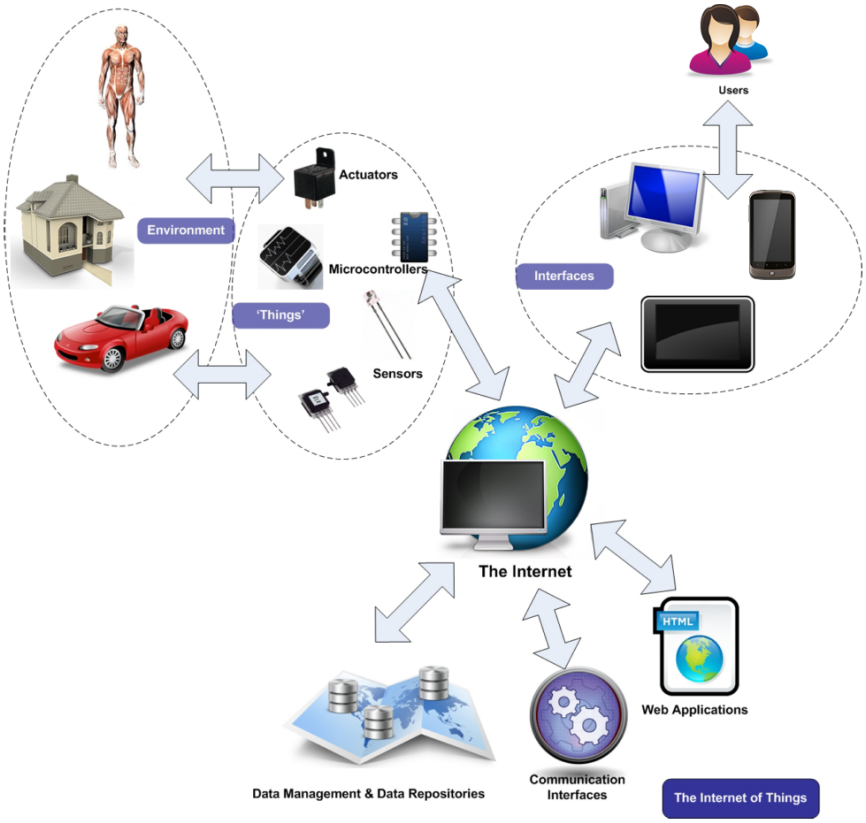


Abbildung 2.2: Überblick der Architektur von “The Internet of Things” (IoT). Quelle: Angelehnt an [Dou12]

Während in [UHM11] eine ausführliche Beschreibung verschiedener Strategien für die Architektur von IoT Lösungen gegeben wird, ist es an dieser Stelle ausreichend wenn die Hauptfunktionalitäten des kleinsten Bestandteils (bzw. eine "Sache", ein Gerät) aufgelistet wird. Eine Zusammenfassung ist in [Dou12] zu finden – die Funktionalitäten, die von einem Gerät zu bieten sind, um sich als ein Mitglied der "Internet of Things" zu "schätzen" wären die folgende:

- **Daten sammeln und übertragen**: Das Gerät empfindet die Umgebung (beispielsweise der Körper oder die Wohnung) und sammelt Daten darüber (z.B. Temperatur und Lichtstärke). Die Übertragung erfolgt entweder in die Richtung eines anderen Geräts, wie zum Beispiel ein Smartphone oder ein Server oder direkt ins Internet.
- **(Verschiedene) Geräte anhand einem Auslöser antreiben**: Das Gerät kann programmiert werden, um andere Apparatur anzutreiben – beispielsweise lassen sich Lichter ein- oder ausschalten. Die Auslöser müssen auch eingestellt werden, beispielsweise können nur dann die Lichter eingeschaltet werden, wenn sich die Lichtstärke in dem vom Benutzer aktuell benutzten Zimmer unter eine gewisse Grenze befindet.

  Während die vorherigen Funktionalitäten auch für allgemeine Sensorik gelten, sind folgende Funktionalitäten normalerweise nur für diejenige Geräte angewendet, die zum Internet of Things gehören:

- **Informationen erhalten**: Eine einzigartige Fähigkeit von IoT Geräte ist die Möglichkeit, eine Verbindung mit der verbundenen Netzwerk oder das Web aufzubauen. Damit lassen sich Informationen über andere in der Netzwerk verfügbaren (IoT) Geräte sowie verschiedene Informationen direkt aus dem Web empfangen. Damit können Live-Updates wie z.B. das Einfügen eines neuen Auslösers oder die Einrichtung neuer Funktionalitäten durchgeführt werden.
- **Assistenz in der Kommunikation**: Weiterhin können IoT Geräte für die Kommunikationsassistenz in der verbundenen Netzwerk zuständig werden – Informationen können zu anderen IoT Geräte oder Netzwerkknoten weitergeleitet werden. Dieses Szenario ist sinnvoll, wenn diejenige zu erreichenen Knoten nicht nah zu einem Endpunkt (beispielsweise ein Router) sind.

Wie in der Auflistung vermerkt, die Funktionalität, die ein IoT Gerät von einem "üblichen" Sensorgerät unterscheidet ist im Prinzip die Herstellung einer direkten oder indirekten Verbindung zum Internet / Web. Durch die vorherige Auflistung lassen sich auch diejenige Charakteristiken von Geräten beschreiben, die durch das Kontext-Modell der Softwarelösung (siehe auch Kapitel 6) abgedeckt werden.

Weiterhin wird den Weg in [TGD+10] beschrieben, wodurch die Verbindung zum Internet von den Geräten verwendet werden kann. Der Domäne von "Web of Things" repräsentiert einen Kreuzungspunkt zwischen zwei Felder, die in der Vergangenheit selten verbunden worden sind – Sensorik mit eingebetteten Netzwerkkomponenten und Web Entwicklung. Die grundlegende Idee der "Web of Things" (WoT) Domäne ist, Web-basierte Interaktionen zwischen eingebetteten Geräten zu ermöglichen. Motivierend dafür ist die hohe Verbreitung und Offenheit der Web Infrastruktur. Deswegen basiert die Umsetzung der WoT Vision auf die Wiederverwendung der Infrastruktur und Web-Standards wie beispielsweise HTTP, XML und RSS.

Ein Teil der Web Infrastruktur ist von REST repräsentiert – das genannte Antreiben anderer Geräte lässt sich durch die Bereitstellung von RESTful APIs abrufen. Die Web of Things Prinzip (Wieder-

verwendung der Web Infrastruktur) ist für diese Arbeit eine wichtige Inspirationsquelle, wie es in dem kommenden Kapitel 5 auch beschrieben wird.

2.7 REST und RESTful Web Services

REpresentational State Transfer (REST) ist ein Architekturmuster für die Realisierung von Verteilten Systemen, das von Roy Fielding in [Fie00] eingeführt wurde. Ein Architekturmuster spezifiziert eine Reihe von Einschränkungen oder Prinzipien, die, wenn sie in der vorgeschlagenen Art und Weise eingesetzt werden, zu einer optimalen Gestaltung der Architektur einer Softwarelösung führen.

Die durch dieses Muster beschriebene Architektur entspricht der grundlegenden Architektur aus der Domäne der Verteilten Systeme, indem mehrere eigenständige Systeme so agieren bzw. kommunizieren und zusammenarbeiten, dass es dem Benutzer transparent bleibt. Es wird dadurch der Eindruck vermittelt, dass der Benutzer ein einzelnes, "ganzes" System benutzt [TS07].

Das Akronym erläutert an sich die wichtigsten Konzepte des Musters und zwar: die Zustände verschiedener Ressourcen werden zwischen den Bestandteilen (die beteiligten Systeme), im Rahmen einer Punkt zu Punkt Kommunikation übertragen.

Diese Beschreibung nennt ein erstes grundlegendes Konzept hinter REST und zwar der Begriff "Ressource". Als Ressourcen lassen sich beliebige, **identifizier**- und **addressierbare** Einzelheiten aus der beschriebenen Domäne bezeichnen. Es reicht jedoch nicht, nur die Existenz einer Ressource zu kennen, die Ressource sollte noch beschrieben werden. Deswegen wird der Zustand einer Ressource zu einem gewissen Zeitpunkt durch eine Repräsentation z.B. in HTML, XML oder JSON vorgegeben.

Im Mittelpunkt der Punkt zu Punkt Kommunikation muss eine bestimmte Ressource stehen. Eine mögliche Analogie an dieser Stelle wäre die Formulierung einer Anfrage vom einem System zu einem anderen. Der Zugriff vom jeweiligen Client zu dem entsprechenden Server erfolgt mithilfe einer standardisierten Schnittstelle wie z.B. HTTP. Die Vorbedingung für die Kommunikation ist das Bestehen eines globalen, normalerweise eindeutigen Pfades bis zur Ressource – im Fall von Web-basierten REST Systemen wird den Pfad durch eine URI bezeichnet. Zudem muss die erwünschte Art der Anfrage bekanntgegeben werden, bzw. welche Operation auf der Ressource zu unternehmen ist: zu diesem Zweck werden Konzepte aus dem Vokabular von HTTP wiederverwendet – die HTTP Verben wie GET, PUT oder POST beschreiben die Aktion oder Operation, die behandelt werden muss.

Eine weitere wichtige Eigenschaft ist die Zustandslosigkeit der Kommunikation: auf dem Server werden keine Informationen über den Client gespeichert. Das heißt, jede formulierte Anfrage vom Client muss genügende Informationen für die Bearbeitung enthalten. Deshalb müssen Informationen über die Sitzung mit dem Server auf dem Client gespeichert werden. Weiterhin müssen die Antworten vom Server **selbstbeschreibend** sein: das heißt, in der Antwort müssen auch Fakten vorhanden sein, die den Vorgang für die Informationsverwertung beschreiben. Beispielsweise muss den Typ der übertragenen Information vorhanden sein, sodass der Client erkennt, welcher Parser einzusetzen ist oder nicht (im Fall von HTTP, wird den *MIME Type*, auch als *media type* bekannt, explizit angegeben) oder ob die Antwort noch zu entkomprimieren ist.

REST verwendet auch verschiedene Abstraktionen, um die Systeme voneinander abzukoppeln: die Clients erwarten nur die Antwort auf die verschickte Anfrage, dem ausgewählten Weg im System. Das heißt, die Existenz anderer Schichten ist dem Clients unbekannt. Aus diesem Grund lassen sich Clients und Server individuell entwickeln, unter Beibehaltung der Interoperabilität bei einer unver-

änderten Schnittstelle. Eine weitere Einschränkung ist die Verwendung von Zwischenspeicherung (engl. Caching): behandelte Anfragen können mit der Information bereichert werden, ob sie für die Zwischenspeicherung geeignet sind oder nicht.

Zu der Abstraktion oder "Automatisierung" des Kommunikationsvorgangs zählt auch die Benutzung von **Hypermedia** (wie beispielsweise Hyperlinks in einem Hypertext). Deswegen soll eine Antwort Referenzen zu weiter verknüpften Ressourcen oder noch verwendbaren Operationen zu dem aktuellen Ressourcen-Typ enthalten. Dadurch wird der Vorgang "automatisch" gesteuert, indem der Client nur entlang der vorgeschlagenen, weiteren Verknüpfungen navigieren sollte, um sich der Antwort zur formulierten Anfrage zu nähern. Für den Umgang mit Hypermedia werden in der Praxis sogenannte Hypertext Application Languages wie HAL[7] oder Hypermedia Types wie Collection+JSON[8] entwickelt. Diese sind als Vorschläge an den W3C eingereicht worden.

Nah verknüpft mit REST sind RESTful Web Services (auch als ein RESTFul Web API bekannt), die mittels der REST Prinzipien HTTP-basierte Web Services definieren. Das erste Merkmal eines RESTful Web Service ist die Basis-URI, die definiert, an welcher URI die HTTP Anfragen formuliert werden müssen. Um auf eine bestimmte Ressource zuzugreifen, muss normalerweise die Basis-URI mit dem Namen der gewünschten Ressource ergänzt werden.

Zusätzlich enthalten die Web Services, wie oben aufgeführt, auch präzise Angaben der unterstützten Repräsentationen der Ressourcen – z.B. JSON, XML o.Ä. Das genannte Vokabular von HTTP-Methoden muss auch vorhanden sein: ein Web Service veröffentlicht auch seine unterstützten HTTP Methoden, z.B. kann die Beschreibung des Web Services deutlich machen, dass beispielsweise das HTTP Verb **PUT** nicht unterstützt wird. Ein letzter Aspekt eines RESTful Web Services ist die Hypermedia-gesteuerte Entdeckung von weiteren möglichen Pfaden bzw. URIs.

Ein Unterschied eines RESTful Web APIs zu beliebigen Web Services wird auch durch die Benutzung von sogenannten "Clean URIs" eindeutig. Ein Clean URI oder ein "Benutzerfreundlicher URI" enthält kein Query String (ein durch ein Fragezeichen getrennter Bestandteil), sondern nur das verwendete Protokoll (z.B. http) und die Domäne (die Autorität, wie z.B. iana.org). Der Unterschied wird mit der Vorstellung eines Beispieles dargestellt:

Eine Query-String basierte URI wie `http://iana.org/service.jsp?cat=int&id=registrar`

ist äquivalent zu der bereinigten Variante `http://iana.org/service/int/registrar`.

Vorteilhaft an Clean URIs ist die Beständigkeit: wenn die Ressource immer noch existiert, wird die URI für eine lange Zeit noch erreichbar sein. Zugleich lässt sich ein Clean URI leichter und logischer eingeben und behalten.

Zusammenfassend lassen sich die REST Prinzipien für die Bereitstellung von REST-basierten Web Services anwenden, die in dieser Arbeit eine wichtige Rolle für die Kommunikation zwischen Frontend und Backend spielen wird, wie im Abschnitt 5.2.1 erläutert wird. Zugleich lässt sich mithilfe von REST Web Services auch der Prinzip von *Service Discovery* umsetzen, wie es in dem kommenden Paragraph vorgestellt wird.

[7] `http://stateless.co/hal_specification.html`
[8] `http://www.amundsen.com/media-types/collection/`

2.7.1 Pico-REST für Komponentenintegration

In [DRAPZ04] werden die Grundprinzipien hinter Pico-REST (Kurzform: pREST) eingeführt: es wir ein Netzwerk- und REST-basierter Zugriffsprotokoll entwickelt, die die vereinfachte aber mächtige Vision des Web über Daten und Dienste, in die Welt der Ubiquitären Computing und damit z.B. auch im Rahmen eines Smart Home Szenario überführen soll. Es wird damit versucht, verschiedene vorhandene Bestandteile oder Komponente der Ziel-Umgebung (wie z.B. Sensorik, HTTP-aufrufbare Dienste) zum allgemeinen Zweck von kontextbewussten Anwendungen (eine Kontext-basierte Anpassung) zu vernetzen. Diese Vernetzung trägt dazu bei, dass heterogene Systeme oder Geräte mittels HTTP integriert bzw. werden können bzw. zusammenarbeiten können.

Die Simplifizierung dieser Technik lässt sich zuerst anhand der verwendeten Technologien bemerken – es werden HTTP Mitteilungen zwischen den einzelnen eingekapselten Komponenten verschickt (s.u.), unter Beachtung der REST Prinzipien. Ein interessanter Aspekt bei der Benutzung von HTTP ist die damit erzielte Flexibilität – es können Web Browser-gesteuerte Interaktionen zwischen den Komponenten realisiert werden. Ein ganzheitliche Perspektive über die für die Verbindung der einzelnen Komponenten erforderlichen Schnittstellen wird vorgegeben: vereinfachte Datentypen wie z.B. Zahlen, Zeichenketten oder boolesche Werte müssen zwischen den Komponenten kommuniziert werden können.

Die von pREST angebotene Funktionalität ist simplifizierend aber effektiv: die Abfrage und Manupulation von Eigenschaften bzw. Daten einer Komponente, sowie die Möglichkeit, eine Funktion anhand angekommenen, Klartext-basierter Mitteilungen. Hiermit lässt sich eine ursprüngliche Konfiguration zwischen einer Client-Anwendung und einer entsprechenden Server-Anwendung realisieren.

Weiterhin unterstützt pREST die lose Kopplung der zu integrierenden Komponente – im Vergleich zu üblichen Herangehensweisen der Vernetzung, die die Etablierung einer gemeinsamen Schnittstelle in Form von IDL (Interface Description Language, eine Sprache womit die Schnittstelle einer Softwarekomponente beschrieben werden kann) oder WSDL (Web Service Description Language, eine XML-basierte Sprache, die die von einer Web Service bereitgestellte Funktionalität beschreibt) Dokumenten vorsehen, stellt pREST eine begrenzte Semantik für die gegenseitige Verbindung vor. Es wird motiviert, dass im Fall von ubiquitären, “in-house” Computing fast nur Mitteilungen der Form “ein Ereignis ist aufgetreten”, “Wert auslesen” oder “Wert setzen” verschickt werden müssen.

Zu diesem Zweck werden nur die Basis-Verben HTTP Verben benutzt, um die einzelnen Komponenten bzw. Ressourcen anfragen zu können: `GET`, `POST` sowie `HEAD`. Eine `GET` Anfrage für einen Dienst wird als eine Art “Discovery” Anfrage angesehen – die Antwort zu der Anfrage besteht aus einer REST-gemäße Repräsentation der für den Aufruf benötigten Parameter. Dementsprechend ist die Formulierung einer `POST` Anfrage für einen Aufruf des Dienstes gehalten.

Das HTTP Protokoll wird weiter ausgenutzt, indem ein vereinfachtes System zur Typisierung (engl. typing system) eingesetzt wird – mithilfe von HTTP Headers und einer MIME-ähnliche Typisierung. Ein `Accept` Header listet die von einer Ressource bearbeitbaren Inhaltstypen (engl. content type) wie z.B. `text/plain` für primitive Datentypen oder `img/*` für Bilder, während ein `Provide` beschreibt, welche von einer Ressource bereitstellbaren Inhaltstypen sind. Vorteilhaft bei der Einbettung dieser “Service Descriptions” in HTTP Headers ist, dass das HTTP Verb `HEAD` zu Nutze gemacht werden kann: es werden dadurch nur die Headers vermittelt und keinen anderen Inhalt.

Der vom pREST vorgestellte Ansatz kann durch seine vereinfachte Herangehensweise beim Aufbauen von Datenpfaden in z.B. einem Sensornetzwerk verwendet werden. Damit wird das grundle-

gende Szenario einer kontextbewussten Anwendung umgesetzt – die Daten aus einem Knoten wird für die Konfigurierung eines anderen Knotens oder die Auslösung eines Dienstes verwendet; die von mehreren Quellen abgeleiteten bzw. gemeldeten Datensätze sind aggregiert und zu relevanten Ereignisse umgewandelt, um eine "Reaktion" bzw. Anpassung in der Umgebung mittels eines Aktors (engl. actuator) zu ermöglichen.

Zusammenfassend erlaubt pREST die ad-hoc, semi-automatische Erstellung von Sensornetzwerke, die bei der Bereitstellung von Kontext-Informationen zu einer oder mehrerer kontextbewussten Anwendungen beitragen kann. Der Vorgang ist in diesem Sinne semi-automatisch, da diejenige Inhaltstypen, die von den neuen Sensoren zu übermitteln sind, vorher bekannt durch z.B. eine Beschreibung des Szenarios müssen. Dafür kann pREST eine mögliche Weiterarbeit an dem erstellten Kontextmodell (siehe auch Kapitel 6) inspirieren, indem die semi-automatische Entdeckung von im ursprünglichen Modell nicht einbezogenen Dienste und Sensorik verwertet werden kann. Ein weiterer Einfluss der genannten Arbeit ist die Ausnutzung von normalerweise unbeachteten HTTP Verben wie `HEAD` oder die Spezifikation der angebotenen Funktionalität durch HTTP Headers. Diese Techniken haben die Beschreibung vorhandener- und eventuelle Entdeckung neuer REST Web Services in der Umgebung inspiriert – weiter Informationen zu diesem Thema sind im Paragraph 7.3.5 zu finden.

2.8 Kontext-Modellierung

In diesem Abschnitt wird Hintergrundwissen übermittelt über den zentralen Fokus dieser Arbeit, unter Berücksichtigung vorheriger und aktueller Entwicklungen aus der Domäne. Dieses Wissen bildet den theoretischen Hintergrund, der dem vorzustellenden Entwurf und Umsetzung des in dieser Arbeit entwickelten Kontext-Modells zugrunde liegt.

2.8.1 Einleitung

Eine erste bedeutende Aussage zum Thema "Kontext" wird in [Meh12] gemacht: Kontext gestaltet das menschliche Erlebnis. Der gegenwärtige Stand ist, dass wir unsere Welt zunehmend durch eine "Linse" der Information betrachten und sogar beeinflussen. Damit wird die Qualität unserer Interaktionen davon abhängig, wie fähig unsere Geräte und Dienste sind, unsere bestimmten Anfragen, Äußerungen oder Gesten zu erkennen und davon unsere breiteren Intentionen und Bedürfnisse abzuleiten.

Menschen können ihre Ideen und Intentionen anderen Menschen zu übermitteln und dadurch eine Reaktion auszulösen. Die Gründe dafür sind vielfältig: zuerst bietet die verwendete Ausdrucksweise bzw. die Sprache reiche Konstrukte zur Äußerung, weiterhin verfügen Menschen in einer Gesellschaft über ein gemeinsames Wissen, welches die verschiedenen Aspekte der Welt umfasst sowie alltägliche Situationen oder Normen und Regeln [DA00].

Wenn Menschen miteinander kommunizieren, werden implizite, situationsbezogene Informationen (zusammengefasst im Begriff "Kontext") benutzt, um die Breite der mitgeteilten Aspekte zu erweitern. Damit ist Kontext ein fast unsichtbarer Akteur in der menschlichen Kommunikation sowie in Aktionen und allgemeinen Umständen. Kontext enthält sachdienliche, zusätzliche Fakten, Regeln oder Axiome, deren Auswertung oder Berücksichtigung unsere Kommunikation effizient, unsere Befehle prozessfähig und unsere Umstände verständlich erbringt [Meh12].

Aus der Perspektive der Mensch-Maschine-Interaktion lässt sich diese menschliche Fähigkeit, Ideen miteinander auszutauschen, sehr schwer in Szenarien ableiten, in denen Menschen mit Rechnern interagieren. Bei der normalen Situation des "Interactive Computing" Situation behält der Benutzer limitierte Mittel, womit Daten eingegeben werden können (d.h. Tastatur, Maus). Auch wenn die Technik inzwischen wesentlich erweiterte Interaktionsmöglichkeiten anbietet, wie multimodale oder taktile Schnittstellen, nutzen die Mechanismen des Mensch-Maschine-Dialogs nicht die "Bandbreite" des Kommunikationskanals aus, da kontextbezogene Details verloren gehen oder nicht genug behandelt werden können [DA00]. Mit einem verbesserten Zugang der Maschine zum Kontext würde sich die Vielfalt der Interaktion verbessern. Somit können Dienste (im Sinne von Systemen oder Softwarelösungen) entworfen werden, die sinnvoller auf ihre Benutzer, deren Situation und Eingaben reagieren.

Um das Konzept von Kontext wirksam nutzen zu können, muss Kontext sowie die Art und Weise, in der er sich nutzen lässt, definiert werden. Ein besseres Verständnis von Kontext wird den Designern einer Anwendung bei der Entscheidung helfen, welche Instanzen von kontextbezogenen Informationen zu beachten sind und wie dieser Vorteil in der Anwendung zu nutzen ist.

Diese Aspekte des Kontext-Bewusstseins werden in den nächsten Abschnitten eingeführt.

2.8.2 Was ist Kontext?

Wie genannt wird in der Mensch-Mensch Interaktion eine große Menge an Information ohne explizite Kommunikation übermittelt. Gesten, Gesichtsmimik, die Beziehung zu anderen Personen und Objekten in der Nähe sowie die geteilten und damit verbundenen Geschichten tragen alle wesentlich dazu bei, die explizite Kommunikation des Kommunikationspartners erfolgreich wahrzunehmen bzw. zu verstehen. Als "Stichwörter" (engl. cues) oder allgemeiner Kontext bekannt, erleichtern diese Hilfsmittel die Aufgabe der Kommunikationspartner, einen gemeinsamen Nenner in der Kommunikation (engl. common ground, grounding) zu finden [DA00].

Die bestimmende Definition von Kontext wurde 2000 von Dey formuliert: Unter Kontext versteht man irgendwelche Information, die verwendet werden kann, um die Situation einer Entität zu beschreiben [DA00]. In diesem Fall lassen sich mit dem Ausdruck "Entität" verschiedene Elemente einbeziehen wie Personen, Orte oder Objekte, die für die Interaktion zwischen Benutzer und Anwendung ("Maschine") als relevant angesehen werden können. Weiterhin gehören zu diesem Begriff sowohl der Benutzer als auch die Anwendung.

Diese Definition von Kontext kann im Fall der Mensch-Maschine -Interaktion kaum angewendet werden, hauptsächlich aus dem Grund, dass fast kein gemeinsamer Kontext zwischen den Handelnden etabliert wird. Zwar können von der menschlichen Seite natürlich Elemente des Kontextes wie z.B. Gesten, Zeichen der Frustration oder Verwirrung bereitgestellt werden, aber die Erkennung oder Wahrnehmung dieser "Hinweise" kann von einem Rechnersystem nicht automatisch erfolgen [DA00].

Diese ersten erkannten Dimensionen von Kontext wurden in [Lee07] zusammengefasst: Identität (Wer), Aktivität (Was), Zeit (Wann), Standort (Wo). Weiterhin lassen sich diese Erkenntnisse in der folgenden "Formel" ausdrücken: Wer + Was + Wann + Wo = Warum. "Warum" repräsentiert die Inferenz-Fähigkeit eines Systems, anders gesagt, aus mehreren Dimensionen und Kontext-Informationen eine Situation oder ein Szenario ableiten zu können und die entsprechende System-

landschaft anzupassen. Schilit, Adams und Want legen in [SAW94] eine ähnliche Perspektive dar: die wichtigsten Aspekte des Kontextes sind “Wo man ist”, “Wer man ist” sowie “Welche Ressourcen sich in deiner Nähe befinden”.

Darüber hinaus ist die Wahrnehmung von Kontext-Elementen bedeutsam im Fall von mobilem, ubiquitärem Computing, bei dem der Benutzer eine erhöhte Bewegungsfreiheit hat. Durch diese Mobilität treten Situationen auf, in denen Kontext als Ganzes (z.B. die aktuelle Lokation einer Person, oder die Elemente der Umgebung) höchst dynamisch ist. Dieses Interaktionsmodell von mobilem Computing erwekct beim Benutzer den Eindruck, dass Informationsdienste zu jeder Zeit und an jedem Ort erreichbar sind. Damit steigern die Anforderungen an Dienste bzw. Anwendungen – sie müssen sich dem aktuellen Nutzerkontext respektive dem breiten Spektrum von möglichen Situationen des Nutzers anpassen, um die Mensch-Maschine-Interaktion zu unterstützen oder besser gesagt mit situationsbezogenen Informationen anzureichern [DA00].

Weiterhin entspricht die oben genannte Definition von Kontext nur einem Teilbereich des möglichen Nutzerkontextes. Nutzerkontext wird definiert in [Meh12] als alles, was ein Nutzer erlebt, liest oder insbesondere äußert, schreibt. Damit gestaltet Nutzerkontext aus, wie die Benutzer eine Mitteilung wahrnehmen und wie sie spezifische und oft mehrdeutige Wörter wie z.B. engl. party (Partei oder Feier) interpretieren. Nutzerkontext bestimmt auch, wie Geräte und Services die Äußerungen und den generierten Inhalt eines Benutzers interpretieren. Ein Beispiel insbesondere für die Dekodierung des gesprochenen Nutzerkontextes ist die Technologie hinter “Siri”[9], eine Software, die der Erkennung und Verarbeitung von natürlich gesprochener Sprache dient. Dieser Technologie zugrunde liegend ist die Erkennung von Entitäten aus einem gegebenen bzw. ausgesprochenen Kontext.

Um den Nutzerkontext benutzen zu können, müssen Systeme und Dienste weiter gehen als nur die Daten der aktuellen, GPS-gesteuerten Nutzerposition auszuwerten oder die Wörter und Anfragen aus Dokumenten zu interpretieren. Sie müssen Orte verstehen, Wissen der Nutzer zu ihren Interessen zuordnen können, integrierend auch die Identität des Nutzers sowie wem oder was sie vertrauen (ihren sozialen Netzwerken und dem sozialen Kontext, in dem sie kommunizieren und zusammenarbeiten). Dies sind Beispiele von Dimensionen des erweiterten Kontextes (siehe auch Abbildung 2.3), die die neue Generation von kontextbewussten Anwendungen prägen werden [Meh12].

Entitäten, die den “größeren”, erweiterten Kontext bestimmen, verschachteln kleineren Entitäten und treiben den Vorgang an, die Präferenzen des Benutzers zu entdecken. Mit jeder Aktivität oder Situation eines Benutzers wird ein Gewinn an Kontext-Informationen erzielt: wie in der Abbildung 2.3 zu bemerken, fängt die Darstellung von erweitertem Kontext mit den üblichen Dimensionen von Zeit und Raum an. Mit der Weiterentwicklung werden verschiedene andere Entitäten erkannt. Dies geschieht mithilfe anderer Dimensionen, wie dem Sozialen oder Semantischen Web sowie durch die Handhabbarkeit von frei verfügbaren Daten (Linked Open Data).

2.8.3 Kontext-Bewusstsein

Die Relevanz von kontextbewussten Anwendungen zeigt sich insbesondere im Fall von mobilem, ubiquitärem Computing, eine Domäne in der die Aufmerksamkeit von Benutzern von großer Bedeutung ist [HIM05].

[9] `http://www.apple.com/ios/siri/`

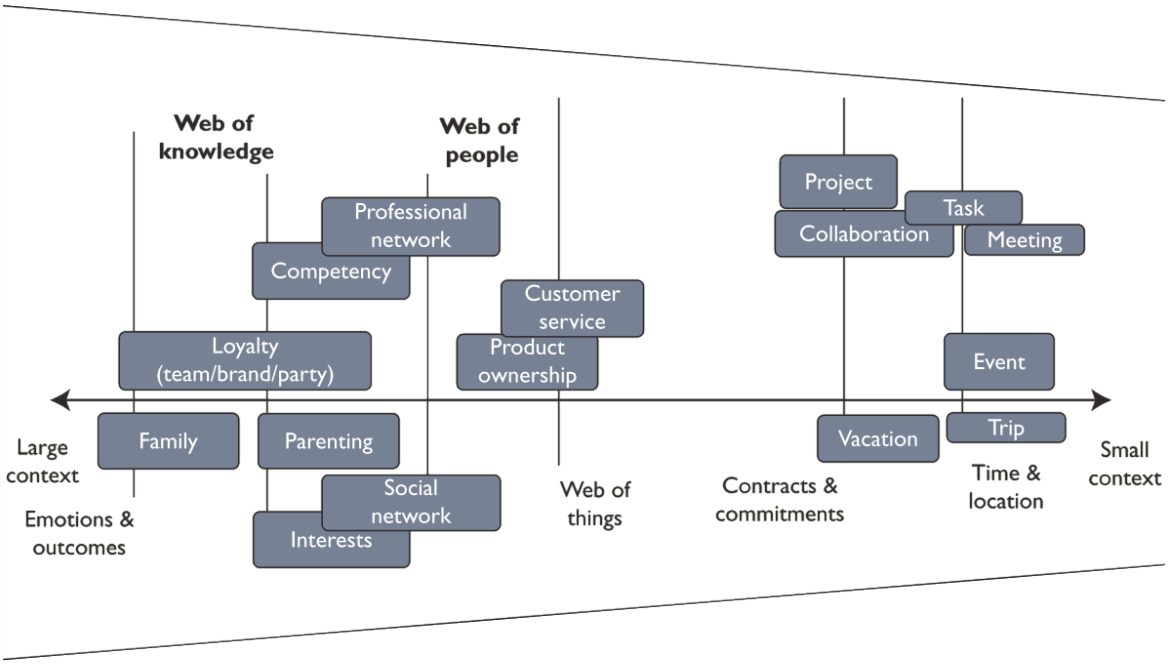


Abbildung 2.3: Erweiterter Kontext (engl. Large Context). Quelle: angelehnt an [Meh12]

Kontext-Bewusstsein (engl. Context Awareness) bezieht sich im weitesten Sinne auf die Fähigkeit von Software, ihre Umgebung wahrzunehmen und darauf zu reagieren. Kontext-Bewusstsein zeigt sich als zentrale Komponente von Context-Aware Computing, eine Disziplin die in [SAW94] folgendermaßen definiert wurde: Context-Aware Computing befasst sich mit der Entwicklung von Software, die den ständig verändernden Kontext eines Individuums untersucht und darauf reagiert.

Weiterhin wird in [Sat02] Kontext-Bewusstsein als ein zentrales Thema in der Entwicklung von ubiquitären Systemen vorgestellt – hiermit wird den Aspekt vorgestellt, dass ein System, das anstrebt, "minimalinvasiv" zu sein, muss auch kontextbewusst sein.

In der Tat hat sich Kontext-Bewusstsein in der letzten Zeit im Forschungsgebiet von Ubiquitären Computing als eine zentrale Technik gezeigt. Insbesondere ist diese Ausprägung für die Adaption mobiler Anwendungen wichtig: hiermit werden Fähigkeiten in den Endgeräte eingebettet, die es erlauben, sich automatisch der Umgebung anzupassen. Eine Folge dieser Automatisierung wäre, dass die expliziten Anleitungen des Anwenders nicht mehr nötig sind.

Das automatisierte Verhalten enthält eine erste Phase der Beobachtung der Umgebung, die als Ergebnis die abgeleiteten Informationen (wie die aktuelle räumliche Lokation und Aktivität des Benutzers oder die aktuell verfügbaren Ressourcen) zurückgeliefert. Des Weiteren findet der Schritt der Anpassung statt, in dem eine Aktion anhand des "wahrgenommenen" Kontextes ausgeführt wird [HIM05].

Sämtliche abgeleiteten Informationen sind unter dem Begriff "Kontext-Informationen" zusammengefasst. Als Quelle für diese Informationen nennen wir Menschen, Sensoren wie GPS-Empfänger, Beschleunigungssensoren, Netzwerk-Monitore oder sogar andere kontextbewusste Anwendungen. Beispiele von kontextbewusste Anwendungen sind digitale Reiseführer: diese stellen zugeschnittene

Informationen vorn, je nach aktueller Lokation und Präferenzen des Anwenders. Eine andere Anpassung kommt vom Smartphone selbst, indem das Rufzeichen und Ruflautstärke aufgrund der Lokation, voraussichtlicher Aktivität sowie das Begleiten des Benutzers anpasst [HIM05].

Darüber hinaus ist Kontext-Bewusstsein ein zentraler Baustein weiterer Technologien, wie der Zukunftstechnologie namens "Ambient Intelligence", die eine intelligente, reagierende Umgebung mittels Rechentechnik aufbaut. Somit werden Häuser, Krankenhäuser und Konferenzzimmer in der Lage sein, die Aktivitäten ihrer Besitzer "wahrzunehmen" und diese Informationen wirksam einzusetzen – indem die Bedürfnisse der Besitzer zu jeder Zeit automatisch erkannt werden.

Auf diesen Aspekt wird in [Cur11] auch eingegangen: durch das Einsetzen von Sensoren in der Umgebung können diese miteinander kommunizieren und gemeinsam zum Erkennen von Bewegungen und letztendlich Aktionen oder Aktivitäten von einem zentralen System beitragen. Eine zur Zeit gängige, obschon "limitierte" Anwendung von Kontext-Bewusstsein im Rahmen von "Ambient Intelligence" Technologien ist die drahtlose Übermittlung von Audio-, Video- und allgemeinen Datensätze zu Geräten in der Wohnung, die einen drahtlosen Zugang zu Informationen und Entertainment ermöglichen.

"Ambient Intelligence", verknüpft mit dem Konzept von Kontext-Bewusstsein führt zu einer anschaulichen Interaktion (sei es über Stimme, Bewegung oder Geste gesteuert) der Einwohner mit ihrer Umgebung im Alltag. Eine Aussage an dieser Stelle ist folgende: durch "Ambient Intelligence" wird eine neue Art von "hands free" Nutzerlebnissen ermöglicht.

2.8.4 Historische Entwicklung

Kontextbewusste Anwendungen sind auf die impliziten Formen von "Eingaben" wie Sensor-erfasste und -abgeleitete Daten angewiesen, um zu vermeiden, die Kontext-Informationen explizit vom Benutzer verlangen zu müssen.

Die ersten naiven Überlegungen zum Thema "Kontext-Erfassung" beschreiben eine simplifizierende Herangehensweise, mit der Kontext erfasst werden könnte: eine explizite Auflistung aller in einer Situation relevanten Aspekte konnte vom Benutzer ausgeführt werden [DA00]. Dieses Verfahren muss aber mit der Zielsetzung von kontextbewusstem Computing (engl. Context-Aware Computing) verglichen werden: die Interaktion mit Rechnersystemen zu erleichtern, was in diesem Fall nicht passiert.

Folglich lässt sich diese Perspektive, Nutzer zu zwingen, die Informationsmenge im System manuell zu vergrößern, nur schwierig und ermüdend lösen. Weiterhin würden möglicherweise durch diesen Denkansatz andere Probleme auftreten und zwar: voraussichtlich werden die Benutzer nicht wissen, ob eine Information eine höhere Relevanz für das System oder für die aktuelle Situation besitzt und es nicht bekannt machen bzw. eingeben.

Die plausiblere und praxistauglichere Methode, das Kontext-Bewusstsein in einer Anwendung zu integrieren, beschreibt einen Ablauf, in dem eine automatische Datenerfassung stattfindet, gefolgt von der Bereitstellung der abgeleiteten Daten zum laufenden Rechensystem oder der Anwendung. Als letztes muss die Maschine "entscheiden", welcher Aspekt der Daten für das entworfene Szenario relevant ist und entsprechend darauf reagieren. Die für diese Entscheidung wichtigen Informationen müssen vorab entworfen worden sein.

Für den Entwurf von Kontext-Bewussten Anwendungen mithilfe der vorher genannten Herangehensweise waren, wie im Jahr 2000 in [DA00] beschrieben, alle Vorbedingungen vorhanden: Zum einen verbreiteten sich kommerzielle, serienmäßig produzierte Erfassungstechnologien, was die Erfassung vom Kontext in variierten Umgebungen durchführbar machte; außerdem wurde der Zugang zu leistungsfähigen, vernetzten Rechnersystemen leichter, welche erlauben würden, die Technologien der Zeit zu benutzen, um Kontext-Informationen zwischen mehreren Anwendungen zu verteilen.

Was aber geschehen ist, ist vergleichbar mit dem geringeren Erfolg der Smart Home Technologien: ein Mangel an Standards, an einheitlichen Methoden zur Entwicklung und Ausführung von kontextbewussten Anwendungen verhinderte die Verbreitung.

Die meisten kontextbewussten Anwendungen wurden für den Einzelfall oder hinsichtlich einer einzigen Klasse von Anwendungen entwickelt, indem die Technologie hinter der Erfassung des Kontextes Vorrang hatte. Damit wurde weder eine Allgemeinheit der Lösungen erzielt noch geschafft: jede neue Anwendung musste von Grund auf neu gebaut werden. Eine erste Erkenntnis, die die Entwicklung von allgemeinen kontextbewussten Anwendungen wesentlich erleichtern würde, wäre eine gemeinsame Architektur, die generische Mechanismen für die Erfassung und Verwaltung von Kontext-Informationen bereitstellen würde. Der nächste Schritt wäre die Spezifizierung und Umsetzung der domänenspezifischen Kontext-Elemente [DA00].

Die Empfehlung von Dey wurde im Laufe der Zeit durch mehrere Lösungsansätze zur Kontext-Modellierung umgesetzt. Dennoch zeigen Studien wie [HIM05], dass sich bis jetzt keine Herangehensweise durchgesetzt hat. Die meisten Ansätze wenden Konzepte aus verbundenen Feldern wie Datenbank-Modellierung oder Semantic-Web -Technologien an. Hierbei ist von Vorteil, dass diese übernommenen Techniken eine gute Tool-Unterstützung sowie teils eine Interoperabilität anbieten. Trotzdem überwiegen die Nachteile: die Fähigkeiten dieser Ansätze entsprechen nicht allen Anforderungen der Kontext-Modellierung bzw. können gewisse Kontext-Informationen nicht modellieren. Deswegen brauchen rein übernommene Techniken gewisse Erweiterungen oder eine Problemumgehung.

Um eine kurze Übersicht über vorherige Kontext-Modellierungstechniken zu geben, werden die ersten Lösungsansätze aus [BBH+08] genannt: Schlüssel-Wert Modelle verwenden zum einen die zu modellierenden Attribute als Schlüssel und lagern deren Ausprägungen als Werte. Der Nachfolger zeigt sich in Form von Auszeichnungssprachen, die Sprachen wie SGML oder XML verwendeten. Weiter führt der W3C einer Standard für die Beschreibung von mobilen Geräten namens “Composite Capabilities / Preference Profile” (CC/PP) ein, die RDF benutzte und vereinfachte Einschränkungen (engl. constraints) und Beziehungen zwischen Kontext-Elementen erlaubte. Zusammen mit dieser Technik wurden auch grundlegende Inferenz-Fähigkeiten mithilfe von jeweils spezialisierten Reasoners vorgestellt.

Wie oben genannt wenden aktuelle Modellierungstechniken Konzepte aus der Welt der relationalen Datenbanken oder Techniken des Wissenmanagements an. Des Weiteren benutzen sie allgemeine Inferenzmethoden, die der Modellierungstechnik entsprechen (wie z.B. die Prädikatenlogik oder die Beschreibungslogik) [BBH+08].

2.8.5 Phasen der Kontext-Modellierung

Nachdem einen Überblick der bisherigen Entwicklung der Kontext-Modellierung im Paragraph 2.8 geschaffen wurde, ist die wichtige Definition eines Kontext-Modells nennenswert. Diese wurde in [BN04] eingeführt und lautet folgendermaßen: ein Kontext-Modell ist eine formelle Beschreibung der relevanten Aspekten einer Umgebung (Teil der realen Welt).

Der Entwicklungsprozess einer kontextbewussten Anwendung kann wesentlich erleichtert werden, wenn ein Kontext-Modell in die Entwicklung mit einbezogen wird. Der Grund dafür ist, dass eine Abstrahierung der Kontext-Erfassung (engl. context sensing) ermöglicht die Herstellung einer "Verbindung" zwischen der realen, nicht-technischen Welt und der technologisierten Ansicht der Anwendung ermöglicht [BN04]. Diese beschriebene Verbindung erfolgt anhand von vordefinierter, erweiterbare und dynamische Kontext-Modelle, die einen gewissen Ausschnitt der Welt beschreibt.

Im Folgenden wird eine Beschreibung der einzelnen Phasen, die zu der eigentlichen Entwicklung eines Kontext-Modells gehören können, gegeben. Zu diesem Zweck wurden ähnliche Meinungen aus der Literaturrecherche zusammengefasst.

In [LRBA10] wird der Lebenszyklus für die Entwicklung von kontextbewussten Anwendungen genannt. Dabei werden drei Phasen identifiziert, die auch mit einem verallgemeinerten Software-Entwicklungslebenszyklus vergleichbar sind:

- **Entwurf**: in dieser Phase wird das Datenmodell des Kontext-Modells festgelegt, normalerweise mithilfe eines UML Tools. Diese Phase lässt sich auch in [CWGN11] sowie in [BPP07] wiederfinden, in Form der äquivalenten Phase der "Modellierung".

  Im Laufe dieser Phase müssen die für die Anwendung relevanten Kontext-Elemente (auch als "Situationen" bekannt) definiert werden, zusammen mit voraussichtlichen Anpassungen, die als Reaktion zu entgegengenommenen Kontext-Daten von der Anwendung unterstützt werden sollen [LRBA10].

  Weiterhin wird in [BOQ+11] eine alternative Beschreibung der hauptsächliche Aufgabe dieser Phase gegeben und zwar: es müssen Beziehungen zwischen jedem Kontext-Element und den dazugehörigen relevanten Aspekt des modellierten Szenarios (wie z.B. Regeln, Benehmen bzw. Anpassung der Anwendung sowie die Präsentation oder Darstellung dieser Aspekte) erstellt werden.

  In dieser Phase lassen sich nur Annahmen vom Entwickler treffen, da wesentlich präzisere Informationen über eine Umgebung nur in der Phase der Konfiguration (s.u.) zu sammeln sind. Hilfreich an dieser Stelle ist die Verwendung eines Metamodells für die Bereitstellung von Annahmen und allgemeinen beobachteten Regeln im beschriebenen System [LRBA10] – solche Techniken werden im Abschnitt 6.1.1 beschrieben.

  Diese Phase liefert als Ergebnis ein statisches Modell, welches mit dynamischen Daten aus der Umgebung befüllt wird, sobald die mit dem Modell verbundene Anwendung aufgestellt wird.

- **Konfiguration**: die Aufgabe dieser Phase ist die Erstellung von konkreten Objekten bzw. Instanzen der modellierten Kontext-Elemente – sämtliche Informationen wie z.B. bestimmte, feste Attribute oder die Lokalisierung verschiedener Elemente (falls bekannt) können hiermit erfasst und in dem erstellten Kontext-Modell gespeichert werden [CWGN11]. Statische Informationen wie z.B. die IP-Adressen der benötigten Netzwerkinfrastruktur könnten als Beispiel gegeben werden.

Darüber hinaus wird in [LRBA10] eine weitere Aktivität dieser Phase genannt: die Ausführung von Tests, die die Funktionalität des Systems und die unterstützten Anfragen ausprobieren, z.B. durch ein Hilfsmittel wie ein "Web Service Testing Tool", beispielsweise SoapUI[10].

Als Ergebnis dieser Phase ist ein dynamisches Kontext-Modell zu liefern [CWGN11], welches den aktuellen bzw. den Start-Zustand der Umgebung beschreibt. Dieses Modell wird als nächstes der in der Umgebung laufenden kontextbewussten Anwendung bereitgestellt.

- **Ausführung**: Die letzte Phase beschreibt den laufenden Betrieb des entwickelten Kontext-Modells – das Modell wird den Anwendungen aus der Umgebung zur Verfügung gestellt, in Form von "programmatischen Objekten" bzw. Instanzen der erstellten Objekte [BPP07].

  Hiermit ist die Definition der genannten Anpassungen an neue Werte aus der Umgebung bedeutsam. Die Werte "strömen" ins Kontext-Modell und eine "Reaktion" ist erwünscht – damit verbunden ist entweder die Erzeugung oder das Auslösen verschiedener Aktionen, wie in [BBH+08] erläutert.

Wie in der Beschreibung der Entwurfsphase genannt, lassen sich generische Methoden oder Techniken für die Erstellung von Kontext-Modellen anwenden. Dieser Vorgang wurde mit der Existenz von Verknüpfungspunkten zwischen der einzelnen Phasen motiviert. Weiter zu behandeln sind die genannten Techniken, welche im Abschnitt 4.3.4 behandelt werden.

[10] `http://www.soapui.org`

3 User-Stories

In diesem Kapitel werden zwei User-Stories eingeführt, welche die Akteure und die Funktionalität der Anwendungen vorstellen. Diese Beispiele dienen dazu, einen Überblick des gesamten Konzeptes zu schaffen. Diese User-Stories beschreiben eine einigermaßen idealisierte Benutzung der Anwendung und werden in den kommenden Kapitel weiter ausgeführt.

Zunächst werden grundlegende Aspekte (wie einen Übersicht des "Diskursuniversums" der Anwendung) eingeführt, die in der Beschreibung der User-Stories eingegangen werden werden.

3.1 Akteure, Umgebung, Systeme

Im Folgenden werden die grundlegenden Bestandteile der User-Stories beschrieben und nämlich die Akteure, deren Umgebung und das Ihnen zur Verfügung gestellten System.

Das Konzept von "Home Sharing" enthält hauptsächlich zwei Gruppen von Akteuren und zwar:

- **Wohnungseigentümer**: Der Wohnungseigentümer besitzt oder mietet eine Wohnung, die direkt oder mithilfe eines Dienstleisters (s.u.) (unter-)vermietet wird.
- **Dienstleister**: Der Dienstleister spielt eine indirekte Rolle und entspricht in der Regel einer Firma, die den Kontakt zwischen Wohnungseigentümer und Wohnungsmieter herstellt. Über den Dienstleister laufen z.B. die Buchungen und die ersten Vereinbarungen zwischen den zwei Gruppen.
- **Wohnungsmieter**: Der Wohnungsmieter mietet die Wohnung des Wohnungseigentümers für einen gewissen, normalerweise kürzeren Zeitabschnitt (beispielsweise einige Tagen bis zu einem Monat).

Die Akteure interagieren aus Sicht der Anwendung und der User-Stories in einer offenem Umgebung, die aber räumlich begrenzt ist. Die Umgebung, in der diese Stories entfalten, besteht aus der folgendem Bestandteile:

- **Wohnung**: Die Wohnung ist vom System als ein räumlich begrenzten Raum angesehen. In diesem Raum werden die meisten Interaktionen zwischen den Akteuren stattfinden. Es wird zwischen einer "üblichen" Wohnung und einer "Smart" Wohnung unterschiedet. Die "Smart Home" kann erweiterte Interaktionsmöglichkeiten anbieten, wie z.B. eine automatisierte Regelung der Temperatur und Feuchtigkeit, oder des Lichtsystemes, automatische Entdeckung von unerwarteten Ereignisse wie ein Gasaustritt oder eine Geländeüberwachung. Darüber hinaus bieten "Smart Home" Umgebungen personalisierte Entertainment-, Präsenz-Erkennung- und Assistenzsysteme, die anhand der installierten Sensorik den Einwohnern Vorschläge generieren können oder sogar in tiefgehenden Situationen "intelligent" agieren bzw. eingreifen können.
- **Umgebung**: Relativ zu der eigentlichen Wohnung können auch Bestandteile der Umgebung betrachtet werden, jedoch dient dies nur dazu, dass die Wohnungsmieter hilfreichen Örtlichkeiten mithilfe des Wohnungseigentümers lokalisieren können. Zum Beispiel können Adressen von Supermärkten in der Nähe der Wohnung den Mietern bereitgestellt werden.

Darüber hinaus müssen bestimmte, technische Systeme zur Verfügung gestellt werden, um sich die Vision der Anwendungen vorstellen zu können. Diese Systeme begleiten die Anpassung der Wohnungsmieter am neuen Kontext und stellen die technische Realisierung des Konzeptes dar. Folgende Systeme werden benötigt:

- **Frontend**: Der clientseitige Teil des Systems besteht aus zwei mobile Anwendungen, welche jeweils den beiden Gruppen von Akteuren zur Verfügung gestellt werden. Das Frontend kommuniziert mit dem Backend, um die Funktionalität des Systems gewährleisten zu können. Es handelt um die folgenden Anwendungen:
  - Anwendung für den Wohnungseigentümer: Diese Anwendung stellt Werkzeuge zur Erfassung der Daten über die Wohnung und deren Bestandteile bereit. Hiermit werden Instanzen des Kontext-Modells (siehe auch 1.5.2) erzeugt.
  - Anwendung für den Wohnungsmieter: Diese Anwendung stellt Werkzeuge bereit, die die Ausnutzung der vorher erfassten Informationen ermöglichen.
- **Backend**: Der serverseitige Teil des Systems wird betrieben auf einem dedizierten Server-System in der Wohnung und bearbeitet die meisten Daten-bezogenen Aufgaben des gesamten Systems. Außerdem ist das Server-System in der Wohnung verortet und schließt sich an dem lokalen Netzwerk ([W]LAN) und gegebenenfalls auch am Internet.

Nachfolgend werden die grundlegenden Abläufe oder Phasen des Konzeptes beschrieben, die sich während der Beschreibung der User-Stories ergeben werden.

3.1.1 Grundlegende Abläufe oder Phasen im System

Ein erster Ablauf im System ist implizit und wird nicht durch einen Anwendungsfall abgedeckt: die **Buchung** der Wohnung wird als ein Ereignis außer Kontrolle des Systems betrachtet. Darauf folgend kommt das System im Einsatz während einer Buchung. Laut Definition beschreibt eine Buchung eine vordefinierte Zeitspanne, in dem die Wohnung des Wohnungseigentümers von den Wohnungsmietern gemietet wird.

Des Weiteren werden vom System die folgenden “Kern-Phasen” des “Home Sharing” Szenarios betrachtet:

- **Vorbereitungsphase**: Die Vorbereitungsphase bezieht sich auf die Zeitspanne zwischen Buchungen und lässt sich mehrmals ausführen. In dieser Phase werden die **Kontext-Elemente** vom Wohnungseigentümer modelliert oder verändert, mit dem Ziel, diese Elemente den neuen Wohnungsmieter zur Verfügung zu stellen. Weitere Details über diese Phase lassen sich aus dem Kapitel 6 entnehmen.
- **Distributionsphase**: Die mobile Anwendung, die von den Wohnungsmietern benutzt wird, muss zuerst verteilt werden. Diese Phase muss bei jeder Buchung wiederholt werden, sodass die Wohnungsmieter über geeignete Zugriffsinformation verfügen können. Für weitere Informationen bezüglich der Umsetzung dieser Phase, siehe auch Paragraph 7.3.3.

- **Mietphase**: Die Mietphase stellt die tatsächliche Laufzeit einer Buchung dar. Während dieser Phase werden diejenige freigeschaltete **Kontext-Elemente** von den Wohnungsmietern **entdeckt** (siehe auch 7.4.3) und beachtet bzw. "benutzt".

Zusammenfassend zeigt Abbildung 3.1 eine graphische Repräsentation der beschriebenen Phasen.

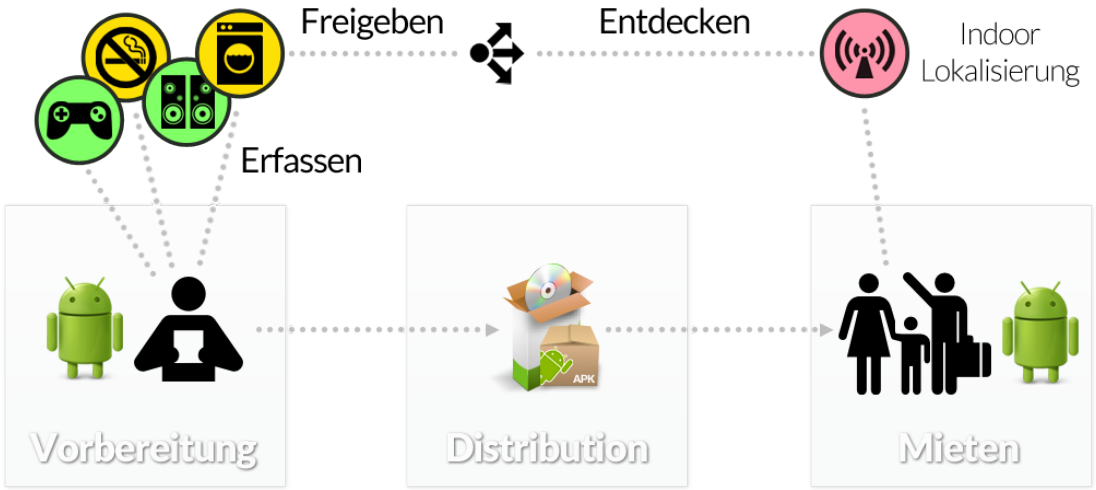


Abbildung 3.1: Die grundlegenden Abläufe oder Phasen im System.

3.2 Konkrete User-Stories

In diesem Absatz wird eine User-Story vorgestellt, durch welche die Funktionalität und mögliche Erweiterungen des Systems herausgestellt wird. Zunächst wird eine übliche Wohnung vorgestellt. Darauf folgt die Beschreibung einer "Smart Home" Umgebung.

3.2.1 Normale Wohnung

Der Wohnungsbesitzer namens Martin A. hat seine erste Anzeige mittels der Internet-Plattform "MyHomeSharing‘" (ein fiktiver "Home Sharing" Dienst) eingestellt und schon eine Anfrage bekommen.

Diesbezüglich wendet sich die viel reisende Kathrin D. an ihn für eine dreitägige Unterkunft in seiner komfortablen Wohnung mitten in der Hamburger Altstadt. Martin beantwortet die Anfrage positiv und freut sich auf den kommenden Besuch.

Dabei erinnert sich Martin daran, dass er von einem Freund etwas über eine neue Anwendung gehört hat. Die Anwendung erlaube einem Wohnungsbesitzer die Gegenstände seiner Wohnung mit einer digitalen "Intelligenz" anzureichern. Die Anwendung soll Gäste dabei unterstützen, sich in der Wohnung und der neuen Umgebung besser zurecht zu finden. Dies bewegt Martin dazu, die Anwendung auszuprobieren.

Im Folgenden werden die beiden Phasen, des in Kapitel 1.5 beschrieben Konzeptes, detaillierter vorgestellt.

Vorbereitungsphase

Martin hat die Möglichkeit das Backend auf verschiedene Arten zu installieren: entweder lässt er von MyHomeSharing eine geeignete Server-Umgebung (sprich ein Server-System, ein Rechner) liefern und installieren oder verwendet seinen eigenen Rechner dafür. Martin entscheidet sich für die letzte Variante, da er in seine Freizeit für Technik und Gadgets interessiert.

Zu diesem Zweck lädt er das "Do-It-Yourself" Installationskit und die dazugehörige Android Anwendung herunter. Nachdem er die Installationsbeschreibung gelesen hat, installiert er die Backend-Anwendung auf seinem PC, der ebenfalls mit dem lokalen Netzwerk verbunden ist. Das Backend-System ist hochgefahren und er kann mit dem Test der mobilen Anwendung (das Frontend) anfangen.

Beim ersten Lauf der Android Anwendung für Wohnungsbesitzer wird der Benutzer aufgefordert ein grundlegendes Profil zu erstellen. Martin wählt einen Benutzername und ein Profilbild aus. Dazu benennt er seine 2-Zi. Wohnung nicht, es fällt ihm kein interessanter Namen ein. Diese nicht eingegebenen Daten werden anhand Martins (eingegebenen) Namens generiert, z.B. ein leeres Profilbild oder ein generischer Name für die Wohnung wie "Die Wohnung von Martin".

Als letztes gibt er die Adresse der Wohnung ein, sodass seine Gäste den Weg ggf. wiederfinden können. Als Erweiterung zu dieser Aktion könnte man sich vorstellen, dass das Profil automatisch mithilfe freigegebenen Dienstleister, wie beispielsweise Social Media Webseiten erstellt werden könnte.

Im Hintergrund werden die eingegebenen Daten in einer Datenbank persistiert. Zuerst wird die Datenbank initialisiert, indem die Relationen erstellt und teilweise mit Daten wie z.B. vordefinierte Kategorien von Kontext-Elementen gefüllt werden. Dieser Prozess ist in dem Frontend (bzw. in der Anwendung für den Wohnungseigentümer) dargestellt durch ein Ladesymbol. Weiterhin werden bestimmte Services und Funktionalitäten auf dem eigentlichen mobilen Gerät auch vorbereitet (siehe Abschnitt 6.2.2 für weitere Informationen). Dieses Szenario könnte als eine Alternative zu einem Kommandozeile-basierten betrachtet werden.

Als nächstes lassen sich Kontext-Elemente vom Wohnungsbesitzer mithilfe der dedizierten mobilen Anwendung erstellen. Zuerst muss Martin eine Kategorie auswählen, zu dem der neue Gegenstand (engl. Asset) gehören wird. Im Anschluss müssen bestimmte Attribute wie z.B. ein Namen, eine Beschreibung oder die Sichtbarkeit (dem Wohnungsmieter App freigegeben oder nicht) angegeben werden. Durch Betätigen des "Speichern" Buttons werden die Informationen an das Backend übertragen. Damit werden Martins modellierte Wohnungsbestandteile, wie die Einleitungen zu der neuen, vollautomatisierten Kaffeemachine oder die wichtige Regel des Rauchverbots in seiner Wohnung auf einer Datenbank persistiert.

Die Vorbereitungsphase ist beendet wenn Martin die generierten QR-Codes für seine relevante Bestandteile (z.B. die Kaffeemaschine) ausdruckt und an einer gut sichtbaren Lokation, wie zum Beispiel neben dem entsprechenden Gerät aufklebt. Damit können Gäste sich die hintergelassenen Informationen anschauen.

Zum Abschluss wird Martin von der Anwendung daran erinnert, dass die Signatur, die für seine nächsten Gäste eindeutig ist, noch denen mitgeteilt werden soll. Er verschickt diese Signatur per E-Mail (eine der möglichen Umsetzungen der Distributionsphase). Diese Bestätigung der Identität

könnte wieder mithilfe von NFC oder mit der Fertigstellung eines signierten Installationspakets für die Gäste erfolgen, dieses Thema wird weiter erläutert im Abschnitt 7.3.3.

Mietphase

Der Ankunftstag seines Gastes Kathrin kommt und Martin begrüßt sein Gast. Er erzählt Kathrin von der neuen Technologie, welche er in seiner Wohnung anbietet. Kathrin ist zuerst skeptisch aber versucht ihre Meinung noch zu ändern nachdem sie mehr über die "komplizierte Methode" zur perfekt gekochten Kaffee von Martin hört und den erklärten Vorgang zum ersten Mal alleine ausführt.

Kathrin lädt die von Martin genannte Anwendung herunter (eine Alternative der Distributionsphase). Sie erstellt ein minimales Profil über ihre Person, ohne Profilbild. Wie schon genannt, könnte die Anwendung die manuelle Erstellung des Profils vermeiden, durch die Benutzung einer API zu einem Dienstleister wie Airbnb oder die Übernahme vom Informationen im Web (selbstverständlich nachdem der Benutzer dies bestätigte und autorisiert hat).

Nach der Erstellung ihres Profils, wird automatisch ein Kalendereintrag mit Erinnerung für ihren Auszugstermin (verfügbar über die Buchungsdaten) angelegt. Martin hat vergessen eine Checkliste für seine Gäste noch zu erstellen, aber Kathrin wird auch ohne klarkommen und die Wohnung in einem guten Zustand hinterlassen können.

Der angebrachte QR-Code kommt am nächsten Tag zum Einsatz. Martin befindet sich auf der Arbeit und Kathrin versucht die besonderen Kaffeetassen zu finden, von denen Martin ihr erzählt hat. Hilfesuchend scannt sie den prominent platzierten QR-Code mit ihrer Anwendung. Angezeigt bekommt sie dann Bedienungsanleitungen in Form einer Checkliste, die den Prozess zur Erstellung "à la carte" begleitet.

Im Hintergrund wird diese Information (voraussichtliche Benutzung der Kaffeemaschine) in der Echtzeit-Datenbank zwischengespeichert, sodass in einem vom Kathrin gemeldeten Notfall mehrere Informationen über die zuletzt getätigten Aktionen übermittelt werden können. Im Falle eines Notfalls könnten diese Informationen in einer "Smart Home" Umgebung genutzt werden, um entsprechende Sensorinformationen wie z.B. ein Rauchmelder oder ein "Stromüberspannungsmonitor" abzufragen.

Als ihr gelungen ist, die perfekte Tasse zu kochen, will Kathrin Martin alles darüber erzählen. Dafür versucht sie Martin über den *Kommunikationskanal* zu erreichen. Da Martin zu dieser Zeit auf der Arbeit ist und deswegen offline, hinterlässt sie eine Mitteilung, in der Hoffnung dass Martin dies sieht und sich freut.

An einem anderen Tag will Kathrin das Stadtviertel rund um die Wohnung erkunden, konnte aber Martins Tipps über die umgebende Lokale nicht finden. Sie erschienen in der Liste der verfügbaren Kontext-Elementen gar nicht! Diese kleine Kritik nahm Martin gut entgegen und versprach, das noch zu regeln. Am selben Abend berichtete er über seine Freude, als er die Mitteilung von Kathrin gesehen hatte.

Der letzte Tag von Kathrins Aufenthalt kommt eher als gedacht. Sie zieht aus und bedankt sich bei Martin für die schöne Zeit. Martin führt die Vorbereitungsphase wieder durch, diesmal denkt er auch an Kathrins Tipps und versucht auch diese abzubilden. Er hofft, die Technologie hat Kathrins Zeit in seiner Wohnung verbessert und freut sich schon auf den nächsten Gast und die Möglichkeit die Anwendung weiter zu testen.

3.2.2 “Smart Home”

Eine weitere mögliche User-Story, welche auf dem in Abschnitt 1.5 vorgestellten Konzept basiert, beschreibt die Aufstellung und Wartung der einzelnen Teilsystemen einer “Smart Home” Umgebung. Diese Aufgaben repräsentieren eine der zentrale Herausforderungen, die den Erfolg von Smart Home Technologien beim breiten Publikum verhindern [RQBA09]. Aktuelle Herangehensweise erfordern eine umfangreiche initiale Systemzusammenstellung und weitere technische Unterstützung von Experten.

Daher stellt sich die Frage, ob eine entfernte Instandhaltung möglich wäre. Dies würde es erlauben, das System aktuell zu halten, eventuelle Sicherheitslücken zu vermeiden sowie neue Funktionalitäten einzubinden. In diesem Fall wird nicht mehr über Wohnungsmieter gesprochen, sondern über Wohnungsbesitzer und Techniker. Der Techniker konnte ebenfalls bei einer spezialisierte Dienstleister von Smart Home Anwendungen angestellt sein. Demnächst werden mögliche Verknüpfungspunkte mit dem allgemeinen Szenario von “(Smart) Home Sharing” eingeführt, ohne eine neue User-Story aufzubauen.

Eine erste Funktionalität beschreibt die angepasste Nutzung des Kommunikationskanals zwischen Wohnungsbesitzer und Techniker. Auf diesen Kanal können nützliche Informationen ausgetauscht werden, jeweils abhängig vom Kontext (in diesem Fall, z.B. bei welchem Gerät des Smart Homes das Problem auftritt und spezifische Merkmale des Defekts).

Ein weiter möglicher Anwendungsfall bezieht sich auf die anfängliche Konfigurierung des Systems: der Techniker könnte dem Wohnungseigentümer bestimmte Kontext-verbundene Anleitungen in der Wohnung für die ersten Benutzungen zurücklassen. Zusätzlich können solche “Metainformationen” weiter benutzt werden, wenn die periodische Wartung von einem anderen Beauftragter durchgeführt wird.

Darüber hinaus würde auch die Möglichkeit bestehen, den umgekehrten Weg zu betrachten und zwar: das intelligente Haus konnte im Fall von unerwartete Ereignisse Fehlermeldungen an die Wartungsstelle zuschicken. Dieser Anwendungsfall wurde nur zur Übersicht eingeführt und überschreitet voraussichtlich den vorgestellten Umfang dieser Arbeit.

Außerdem lassen sich Erweiterungen bzw. Verbesserungen der vorgestellten Techniken im Rahmen einer “Smart Home” Umgebung nennen: beispielsweise könnte die Indoor-Lokalisierung automatisch erfolgen, ohne den Eingriff (z.B. das Einscannen einer QR-Code) des Benutzers zu erfordern, wie bereits im Abschnitt 2.5 vorgestellt. Jedoch können einige Zukunftstechnologien für die Lokalisierung und Informierung seiner Gäste eingesetzt werden: NFC Tags können in die Wohnung bzw. in einzelnen Räume oder auf Gegenstände “installiert” werden, um eine automatische Kontext-Anpassung zu ermöglichen.

3.3 Zusammenfassung

Im diesen Kapitel wurden mögliche User-Stories vorgestellt, die die gestrebten Funktionalitäten der zu entwickelnden Anwendungen in einen textuellen Form vorstellen. Diese Texte werden im Anschluss die Basis für die Erstellung von Anforderungen an der zu entwickelnden Anwendungen herstellen. Nachdem in vorherigen Kapitel eine Vorstellung der Konzepten dieser Arbeit erfolgte, wird in den nächsten Kapitel den Weg zur Spezifikation einer Software-Architektur verfolgt.

Der erste Schritt dafür ist die Definition von Anforderungen an das zu entwickelnde Softwareprodukt und dessen Bestandteile (technische Architektur bzw. Server-System sowie mobile Anwendungen), eine Thematik die in dem kommenden Kapitel vorgestellt wird.

4 Anforderungserhebung

Im Rahmen dieses Kapitels werden die Anforderungen für die zu entwickelnde Softwarelösung erhoben bzw. definiert. Als Startpunkt wird die Einführung einiger Grundlagen verwendet; am Ende des Kapitels werden Anwendungsfälle aus der etablierten Anforderungen erstellt und beschrieben.

4.1 Analyse aktueller Anbieter von “Home Sharing” Lösungen

Nachdem in dem vorherigen Kapitel mögliche User-Stories eingeführt worden sind, lässt sich eine Analyse der Anforderungen an einer möglichen Softwarelösung für die Unterstützung und Erweiterung (wie im Abschnitt 1.5.1 sowie im Abschnitt 2.4.1 motiviert) des Home Sharing “Erlebnisses” bzw. dieser Stories durchführen. Es wird damit untersucht, ob der letzte Stand in der Industrie eine Umsetzung der User-Stories erlauben würde.

Um die Bezeichnung “Erweiterung von Home Sharing” des Konzeptes zu begründen, wurde eine Analyse der zur Verfügung gestellten Funktionalitäten für die größten Anbieter von Home Sharing Plattformen ausgeführt. Die untersuchten Anbieter sind: 9flats, Wimdu und Airbnb.

Die Analyse wurde anhand der online, aus der Startseite abrufbaren Informationen geführt – Seiten wie “Häufig gestellten Fragen”, “Wie es funktioniert” oder “Über uns” wurden verglichen. Der Zweck der Analyse, war die Überprüfung folgender Annahme: aktuelle Lösungen aus der Home Sharing Domäne setzen den Akzent nicht unbedingt auf die technologische Unterstützung der Wohnungsmieter und -eigentümer durch das Mietintervall. Die Kriterien, die die Analyse gesteuert haben, sind die folgende:

1. **Erfassung von tiefgreifender Datenbestände über die Wohnung und deren Umgebung**: Unter “tiefgreifende Datenbestände” lassen sich erweiterte Informationen über die Wohnung wie z.B. Hausregeln, allgemeine Richtlinien oder einzelne Anleitungen zur Verwendung oder Verfügbarkeit verschiedener Bestandteile der Wohnung wie beispielsweise Geräte oder Zimmern. In der Welt der Immobilie lassen sich solche Informationen durch den Bezeichner “Ausstattungsmerkmale” zusammenfassen.

   Hierhin gehören auch Daten über die Umgebung außerhalb der Wohnung wie z.B. Annehmlichkeiten in der Nähe, zu bedenken Regeln des Wohnblocks oder Quartiers.

2. **Förderung des spontanen Kontakts während des Mietintervalls**: Während die Kontaktaufnahme bevor der Buchung wichtig ist, wichtiger ist die Existenz einer Kontaktmethode während der Aufenthalt in der Wohnung. Hiermit können spontan Probleme verschiedener Schwierigkeitsgrade besprochen werden, unter bestimmte Annahmen wie Verfügbarkeit oder Privatsphäre.

3. **Unterstützung der Mobilität**: Ein wesentlicher Punkt ist auch die Existenz einer mobilen Anwendung für die angebotene Dienstleistung. Die Anwendung würde dann das erste Kriterium weiter unterstützen, indem die Datenbestände mobil und im besten Fall auch außerhalb der Wohnung abrufbar wären.

Die Ergebnisse der Analyse, gruppiert nach Kriterium, sind in der folgenden Auflistung zu finden:

- **Erfassung von tiefgreifender Datenbestände über die Wohnung und deren Umgebung**: Alle Webseiten erlauben die Annotierung des Inserats mit einer Beschreibung der Wohnung und dessen Gegenstände. Jeweils hier zu beachten ist die Allgemeinheit, es können keine Annotierungen auf bestimmte Elemente erfolgen. Die Granularität dieser Beschreibungen könnte dann verbessert werden, für den Fall dass der modellierte Element präzisere Angaben hinsichtlich der Verwendung oder Pflege braucht.

  Weiterhin erlauben alle Anbieter die Erfassung von Hausregeln. Jeweils lassen sich nur Freitext Einträge erstellen, es fehlt jedoch eine weitere Kategorisierung dieser Einträge oder die eventuelle Verbindung zu einem modellierten Gegenstand (engl. "asset") der Wohnung. Trotzdem ist eine Kategorisierung zu bemerken, im Fall aller Anbieter: eine so gesehene "allgemein bekannte Hausregel" wie die Möglichkeit, Haustiere in der Wohnung unterzubringen ist behandelt jedoch nicht als eine Hausregel, sondern in einer ähnlichen Art und Weise wie die Eigenschaften der Wohnung (u.Ä. Anzahl der Zimmer). Es besteht also keine Allgemeinheit des "Hausregeln" Konzeptes und keine Möglichkeit, eine Hausregel zu einer bestimmten Gerät oder zu einem Gegenstand in der Wohnung zuzuweisen. Wünschenswert an dieser Stelle wäre eine einheitliche, zu jeder Zeit abrufbare Darstellung.

  Bezogen auf die Umgebung lässt sich nur bei manchen Anbieter eine erweiterte Beschreibung (das heißt, außer die ungefähre oder genaue Adresse) finden. Airbnb unterscheidet sich bezüglich dieses Kriteriums von den anderen Anbieter, indem ein "Reiseführer" für die Gäste erstellt werden kann. Dadurch werden bestimmte Örtlichkeiten oder Sehenswürdigkeiten aufgelistet und auf einer Landkarte lokalisiert. Dazu lassen sich online Routen zu jeder Örtlichkeit berechnen, anschließend kann den Reiseführer ausgedruckt werden. Die Benutzbarkeit dieser Auflistung ist nicht äußerst hoch: diese Daten verknüpfen nur den Standort bzw. die vermietete Wohnung mit der Umgebung und erlauben keine einfache Navigation dahin. Während eine Weiterleitung zu einer spezialisierten Dienst für die Beschreibung solcher Örtlichkeiten möglich ist, ließt sich diese Möglichkeit nur online abrufen.

- **Förderung des spontanen Kontakts während des Mietintervalls**: Im Fall dieses Kriteriums lässt sich eine Gleichheit aller Anbieter bemerken – es wird ein internes, privates Nachrichtensystem benutzen. Eine Bemerkung dazu wäre, dass in keinen der verfügbaren "Gebrauchsanleitungen" eine Erwähnung gemacht wird, dass das Nachrichtensystem auch *während* und nicht nur bevor der Buchung verwendet werden könnte. Eine eventuelle weitere Investigation könnte noch herausstellen, dass diese Funktionalität doch eingebettet im Produkt ist. Die Alternative zu der Kommunikation während des Mietintervalls ist nicht direkt vorhanden – nur eine Kommunikation zwischen Mieter und Dienstleister oder zwischen Eigentümer und Dienstleister wird bei allen Anbieter erwähnt, in Form einer Rufnummer für die Kundenbetreuung (die nur im Fall von Airbnb rund um die Uhr erreichbar ist). Aus diesem Grund ist die explizite oder technische Realisierbarkeit einer spontanen Kontaktaufnahme vom Mieter zu Eigentümer und umgekehrt nicht vorhanden.

  Darüber hinaus ist das angebotene Kommunikationsmodell eher statisch geprägt, mit der bei allen Anbieter einheitlichen Verwendung des "Inbox" Modells. Es besteht also keine Möglichkeit, spontan und interaktiv Mitteilungen auszutauschen.

- **Unterstützung der Mobilität**: Auch hinsichtlich dieses Kriteriums ist ein erheblicher Unterschied zwischen den analysierten Anbieter zu bemerken – nur Airbnb bietet eine mobile Anwendung für das angebotene Produkt bzw. Plattform für Home Sharing. Die Anwendung unterstützt alle Funk-

tionalitäten des Produkts; Außerdem verfügt nur die Airbnb Webseite über eine mobile Version. Das Wichtigste an diesem Kriterium ist die Stellung eines Anbieters gegen das mobile Nutzen des angebotenen Produktes. Darüber hinaus wird keiner der vorherigen Kriterien vollständig in der mobilen Version unterstützt: es lassen sich auf dem ersten Blick keine erweiterte Zusatzinformationen wie Hausregel oder Gegenstände modellieren mittels der Anwendung. Im Fall der Kommunikation können nur Buchung-bezogene Mitteilungen verschickt werden. Eine spontane, interaktive Kommunikation, wie vorher erläutert, ist auf einem mobilen Gerät nicht möglich.

Zusammenfassend betrachten bzw. definieren die genannten und analysierten Anbieter von “Home Sharing” Lösungen das Erlebnis von Home Sharing im Ganzen (mit den verschiedenen Vorgänge wie Buchung, Mieten und Auschecken) aber nicht unbedingt im Tiefen. Eine Unterstützung des Benutzers bzw. des Wohnungsmieters während des Mietintervalls, aus einer technologischen sowie mobilen Hinsicht, ist eventuell zu oberflächig definiert oder nicht unbedingt mit einbezogen im gesamten Konzept. Diese Arbeit setzt sich das Ziel, dieses gesamte Erlebnis anzureichern und damit als eine möglich hilfreiche Erweiterung der “Home Sharing” Idee zu agieren.

4.2 Grundlagen

Es muss zuerst eine Methodologie, ein Prozess für die Erhebung der Anforderungen verwendet werden – in diesem Fall wird das oft genannte Modell zur Anforderungserhebung aus [Rup09] benutzt. Um die Anforderungen einheitlich und effizient definieren zu können, wird in der vorher genannte Arbeit das folgende Schema vorgeschlagen:

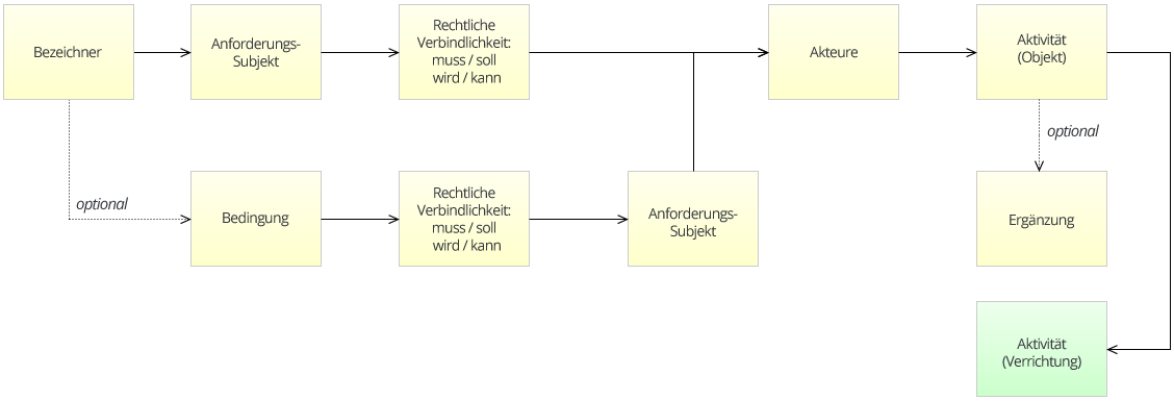


Abbildung 4.1: Eine Schablone für die Feststellung bzw. Formulierung von Software-Anforderungen. Quelle: Angelehnt an [Rup09]

Zusätzlich zu dieser Gliederung lassen sich weitere nützliche Empfehlungen aus [Win11] abbilden – es wurden verschiedene Regel, die für jede Definition einer Anforderung gelten sollten. Zwar sollte jede Anforderung folgende Richtlinien beachten:

- Formulierung in einem Hauptsatz
- Inklusion einer einzigen Anforderung pro Satz

- Die Bestandteile einer Anforderung sind:
  - Ein eindeutiger Bezeichner (z.B. eine ID-Nummer)
  - Eine Aktivität (die Verrichtung am Objekt)

  Weiterhin müssen auch folgende Aspekte in einer Anforderungsdefintion vorhanden sein:
- Akteure (sodass die Aktivität von einer Entität ausgeführt wird), mögliche Akteure wären:
  - Das System selbst
  - Verknüpft mit Benutzerinteraktion
  - Verknüpft mit einem Objekt aus einem Fremdsystem
- Die rechtliche Verbindlichkeit (eine Formalisierung)
- *Optional*: Ergänzungspunkte der Aktivität und zwar
  - Subjekte, die die Verrichtung unterstützen
  - Merkmale der Qualitätssicherung (Qualitätseigenschaften)
- *Optional*: Weitere Randbedingungen wie z.B.
  - Vorbedingung
  - Auslöser (bspw. ein Ereignis)

Unter Verwendung der aufgelisteten Kriterien lassen sich die folgende Kategorien von Anforderungen unterscheiden:

1. **Pflichtanforderungen (Schlüsselwort: Muss)** – Hiermit werden Anforderungen beschrieben, die unbedingt erfüllt werden müssen
2. **Wunschanforderungen (Schlüsselwort: Sollte oder Soll** – Anforderungen, die erfüllt werden sollten, insofern die zeitliche Planung dies erlaubt bzw. zulässt
3. **Vorschlaganforderungen (Schlüsselwort: Kann)** – Diese Art von Anforderungen werden nur dann erfüllt, wenn alle andere Anforderungen erledigt sind (anders gesagt, falls die benötigten Ressourcen verfügbar sind). Durch diese Anforderungen werden normalerweise optionale Funktionalitäten bedeckt.

Nachdem die meist verwendeten Grundregel für die Formulierung von Anforderungen eingeführt sind, werden alle Arten von Anforderungen näher eingegangen.

4.3 Anforderungsdefinition

In dem aktuellen Absatz werden die wesentlichen Anforderungen (AF) an der zu entwickelnden Softwarelösung aufgelistet, je nach Anforderunsklasse (Funktionale oder Nicht-Funktionale Anforderungen). Die Anforderungen wurden anhand der User-Stories und der durchgeführten Analyse erfasst. Als Startpunkt werden die funktionalen Anforderungen festgelegt.

4.3.1 Funktionale Anforderungen

Die “Funktionale” Klasse von Anforderungen stellt fest, was die Softwarelösung leisten soll. Im Folgenden werden die Anforderungen aufgeschlüsselt, aus Sicht der jeweilig betroffenen Systemen (die zwei mobilen Anwendungen aus dem Frontend sowie das Backend).

4.3.1.1 Backend

Die folgenden Anforderungen beziehen sich auf der “Backend” Komponente, welcher Struktur bzw. Architektur in dem folgenden Kapitel vorgestellt wird. Die beschriebene Aktivität ist in diesem Fall der Normalbetrieb bzw. die Beantwortung von Anfragen aus dem Frontend.

Identifikationsnummer	Anforderung
AF-F 1.1	Das Backend muss das Szenario von Home Sharing unterstützen
AF-F 1.2	Das Backend muss die in dem Kontextmodell definierten Services entwerfen und umsetzen
AF-F 1.3	Das Backend muss mehrere RESTful Services zur Verfügung stellen
AF-F 1.4	Das Backend muss mit dem Frontend kommunizieren können
AF-F 1.5	Das Backend muss die Persistierung von kontextbezogenen Informationen gewährleisten
AF-F 1.6	Das Backend muss Erweiterungen an dem vorhandenen Basis-Kontextmodell erlauben
AF-F 1.7	Das Backend muss mit geographischen Daten umgehen können
AF-F 1.8	Das Backend muss die Sicherheitskonzepte umsetzen
AF-F 1.9	Das Backend soll eine Schnittstelle für den Kommunikationskanal und die Distribution von Kontext Elementen bereitstellen
AF-F 1.10	Das Backend soll in einer möglichst schnellen Weise Antworte zu gleichzeitigen Anfragen abliefern
AF-F 1.11	Das Backend soll automatisiert Einstellungen in dem Frontend in der Vorbereitungsphase abändern können

Tabelle 4.1: Funktionale Anforderungen am Backend

4.3.1.2 Mobile Anwendungen (Frontend)

Die folgenden Anforderungen beziehen sich auf der “Frontend” Komponente, welcher Struktur bzw. Architektur in dem folgenden Kapitel vorgestellt wird. Die beschriebene Aktivität ist in diesem Fall der Normalbetrieb bzw. die Benutzung der Anwendungen (für Wohnungseigentümer sowie Wohnungsmieter).

Identifikationsnummer	Anforderung
AF-F 2.1	Das Frontend muss mit dem Backend kommunizieren können (müssen sich im selben Netz befinden)
AF-F 2.2	Das Frontend muss den Sicherheitskonzepte entsprechen
AF-F 2.3	Das Frontend muss die allererste Benutzung des Apps erkennen können

AF-F 2.4	Das Frontend soll die Erstellung eines Nutzerprofils anbieten
AF-F 2.5	Das Frontend kann den Kommunikationskanal von Eigentümer zu Benutzer bereitstellen
AF-F 2.7	Das **Wonungseigentümer** Frontend muss verschiedene Möglichkeiten zur Befüllung des Kontext-Modells anbieten
AF-F 2.8	Das **Wonungseigentümer** Frontend muss geographischen Daten erfassen können
AF-F 2.9	Das **Wonungseigentümer** Frontend muss nachträglich Elemente des Kontext-Modells anpassen können (bearbeiten, einfügen, löschen)
AF-F 2.10	Das **Wonungseigentümer** Frontend soll Elemente des Äußeren-Kontextes erfassen können
AF-F 2.11	Das **Wonungseigentümer** Frontend kann eine Übersicht der modellierten Elementen in der Wohnung anzeigen
AF-F 2.12	Das **Wonungseigentümer** Frontend kann ein Element des Kontext-Modells mit eigenen definierten Eigenschaften erweitern
AF-F 2.14	Das **Wohnungsmieter** Frontend muss das befüllte Kontextmodell anfragen können
AF-F 2.15	Das **Wohnungsmieter** Frontend muss eine Auflistung der modellierten Elementen des Kontext-Modells anzeigen können
AF-F 2.16	Das **Wohnungsmieter** Frontend muss zumindest eine Methode der Indoor-Lokalisierung anwenden können
AF-F 2.17	Das **Wohnungsmieter** Frontend kann erweiterte Methoden zur Indoor-Lokalisierung wie z.B. NFC benutzen

Tabelle 4.2: Funktionale Anforderungen am Frontend

Weiterhin lassen sich nicht-funktionale Anforderungen definieren.

4.3.2 Nicht-funktionale Anforderungen

Unter der Kategorie der nicht-funktionalen Anforderungen fallen feststellbare Aspekte eines Systems, die nicht direkt zu einer Funktionalität eines Systems verbunden sind [Pre11]. Beispiele nicht-funktionale Anforderungen sind Eigenschaften bzw. Parameter bezüglich der erzielten Performanz oder Verfügbarkeit des zu entwickelnden Systems.

Darüber hinaus werden nicht-funktionale Anforderungen in weiteren Untergruppen gegliedert [Win11]:

- **Technische Anforderungen** – diese Kategorie enthält Anforderungen, die auf die einzusetzenen Technologien, das System selbst und dessen Umfeld bezogen sind.

- **Prozessanforderungen** – an dieser Stelle wird den geplanten Ablauf des Softwareentwicklugsprozesses beschreiben, wie z.B. das Vorgehensmodell.

- **Qualitätsanforderungen** – hiermit werden qualitativen Merkmale des Systems enthalten, wie z.B. die Verwendung von Testgetriebenen Entwicklung zur Validierung verschiedener Qualitätsmerkmale.
- **Sonstige Anforderungen** – hier werden diejenige Anforderungen umgefasst, die als zu keiner vorher erläuterten Kategorie passend gehalten werden.

Als erstes werden die Technischen Anforderungen behandelt, je nach betroffenen System.

4.3.2.1 Backend

An dieser Stelle gelten die im Abschnitt 4.3.1.1 genannten Merkmale.

Identifikationsnummer	Anforderung
AF-T 3.1	Das Backend muss mit dem Heimnetzwerk (LAN) verbunden sein
AF-T 3.2	Es muss eine objekt-orientierte und dynamische Programmiersprache verwendet werden, z.B. Ruby
AF-T 3.3	Das Backend muss eine performantes, relationales Datenbankmanagementsystem verwenden, z.B. PostgreSQL
AF-T 3.4	Das Backend muss die verwandten Anforderungen des Kontext-Modells berücksichtigen
AF-T 3.5	Es muss ein performanter, nicht sequentieller Web Server für die Bereitstellung der REST API benutzt werden, z.B. Thin
AF-T 3.6	Das Backend kann eine Middleware-Komponente bereitstellen, für die Realisierung des Kommunikationskanals, z.B. RabbitMQ
AF-T 3.7	Das Backend kann ein Middleware-Protkoll für die Funktionalitäten des Kommunikationskanals bereitstellen
AF-T 3.8	Es können verschiedene Web basierten Services benutzt werden, insoweit Internetzugriff vorhanden ist

Tabelle 4.3: Technische Anforderungen am Backend

4.3.2.2 Frontend

An dieser Stelle gelten die im Absatz 4.3.1.2 genannten Merkmale.

Identifikationsnummer	Anforderung
AF-T 4.1	Das Frontend muss sich mit der Heimnetzwerk (LAN) z.B. über WLAN verbinden können
AF-T 4.2	Es muss eine Plattform für mobile Entwicklung benutzt werden, z.B. Android
AF-T 4.3	Das Frontend muss zumindest Android API Level 10 unterstützen
AF-T 4.4	Es sollen die neusten Interaktionsparadigmen der mobilen Plattform angewendet werden, z.B. Android 3.0+
AF-T 4.5	Das Frontend muss sich leicht bedienen lassen

Tabelle 4.4: Technische Anforderungen am Frontend

4.3.2.3 Prozessanforderungen

Durch Prozessanforderungen werden Anforderungen zusammengefasst, die den allgemeinen Rahmenbedingungen des zu leistenden Softwareentwicklungsprozesses betreffen.

Identifikationsnummer	Anforderung
AF-P 5.1	Der Projektlaufzeit beträgt höchstens 6 Monate
AF-P 5.2	Es müssen periodische Meetings mit den Betreuern gehalten werden

Tabelle 4.5: Prozessanforderungen an der Softwarelösung

4.3.2.4 Qualitätsanforderungen

Die Definition von Qualitätsanforderungen an einer Softwarelösung gehört zu der Disziplin der Software-Qualitätsmanagement. Die Qualitätsanforderungen befassen sich damit, die einzuhaltenen Qualitätsziele sowie die Maßnahmen zur Erfüllung dessen festzustellen.

Identifikationsnummer	Anforderung
AF-Q 6.1	Es muss eine Evaluation des finalen Prototyps durchgeführt werden
AF-Q 6.2	Die Evaluation muss die definierten funktionalen Anforderungen anhand bezüglich ihrer Erfüllung untersuchen
AF-Q 6.3	Eine subjektive Qualitätsanalyse mithilfe eines Experteninterviews kann durchgeführt werden
AF-Q 6.4	Es müssen bestimmte Qualitätsanforderungen an der Dokumentation des Qualltextes eingehalten werden
AF-Q 6.5	Es sollen Prinzipien der objektorientierten Software Entwicklung mitberücksichtigt und eingehalten werden, wie z.B. SOLID
AF-Q 6.6	Es sollen Test-betriebene Entwicklungsmethoden verwendet werden, wie z.B. BDD

Tabelle 4.6: Qualitätsanforderungen an der Softwarelösung

4.3.3 Zusammenfassung

Die hiermit vorgestellten Anforderungen beschreiben die Eigenschaften und Rahmenbedingungen, die den Entwurf und folglich die Implementierung der Softwarelösung ausprägen werden. Die Anforderungen werden als Basis für die Definition des Entwurfs der in dieser Arbeit entstehenden Softwareprodukte – ein Thema, welches als nächstes in Kapitel 5 behandelt wird.

Darüber hinaus müssen in diesem Kapitel weitere Anforderungen gestellt werden, die den hauptsächlichen Beitrag dieser Arbeit umfassen – die Anforderungen, die eine Technik zur Erstellung eines Kontext-Modells (im Folgenden als eine *Modellierungstechnik* gekennzeichnet) erfüllen muss.

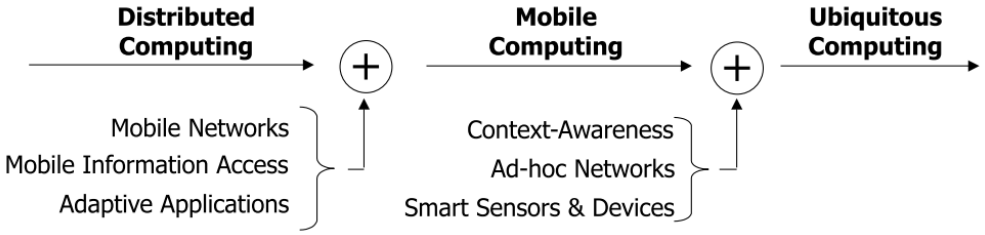


Abbildung 4.2: Entwicklungshierarchie von Ubiquitous Computing Systeme. Quelle: [SLP04]

4.3.4 Anforderungen an eine Modellierungstechnik

In diesem Abschnitt werden aus der Literatur bekannte Anforderungen an eine Kontext-Modellierungstechnik vorgestellt. Da diese aus mehreren Quellen stammen, wird keine konkrete Technik beschrieben, sondern verschiedene Bestandteile, die für die Entwicklung einer hybriden Modellierungstechnik verwendet werden könnten.

Eine erste Kategorie von Anforderungen stammt aus der Entwicklungshierarchie von Ubiquitous Computing Systemen, die eine spezialisierte Klasse von mobilen sowie verteilten Systemen sind [SLP04]. Diese Tatsache wird auch in der Abbildung 4.2 graphisch dargestellt.

Damit lässt sich zuerst eine Auflistung der Anforderungen an generische verteilte Systeme erstellen:

- **Unterstützung der Mobilität** [BPP07]: Da alle wesentlichen Komponenten des Kontext-Modells (insb. Sensoren und Anwendungen) mobil sein können, müssen gezielt Kommunikationsprotokolle ausgewählt werden, die flexible Formen von Routing unterstützen wie z.B. HTTP oder TCP/UDP. Außerdem können Kontext-Informationen zusammen mit den Komponenten des Systems “auswandern” (engl. migrate).

- **Fehlertoleranz** [BPP07]: Im Normalbetrieb eines kontextbewussten Systems werden Sensoren und andere Komponenten des Systems mit einer erhöhten Wahrscheinlichkeit fehlschlagen. Deswegen müssen Inkonsistenzen und Unvollständigkeiten in den Daten erlaubt sein und im Betrieb berücksichtigt werden. Weitere Strategien nennen die Möglichkeit einer Interpolation von unvollständigen Datensätzen (vgl. [SLP04]). Des Weiteren können auch Unterbrechungen in der Verbindung eintreten – das System muss ungeachtet dessen den Betrieb fortsetzen, ohne viele Ressourcen zur Problemidentifikation und -behandlung zu benötigen.

- **Berücksichtigung der Heterogenität** [BPP07]: Ein Faktor in der Entwicklung von verteilten Systeme ist die Überwindung der Heterogenität – es können verschiedene Systemen integriert werden, die ihre eigene Vorstellung des Datenmodells oder des Kommunikationsprotokolls haben.

Weitere Anforderungen können von der Klasse der Ubiquitous Computing Systeme abgeleitet werden und zwar:

- **Performanz** [HIM05]: Inmitten eines kontextbewussten Systems steht die Anforderung an die Echtzeit-Verarbeitung von Daten, die von mehreren verteilten Quellen gemeldet werden können.

Aus dieser Perspektive müssen Anwendungen eine geringere Reaktionszeit bereitstellen, um ein Gefühl einer nahtlosen Mensch-Maschine-Interaktion zu gewährleisten. Diese Aufgabe stellt sich noch schwieriger in der Laufzeit-Umgebung eines mobilen Gerätes heraus.

- **Skalierbarkeit** [HIM05]: Die Größe des Systems kann wesentlich mit der Zeit variieren, z.B. durch das Einfügen einer erheblichen Anzahl von Sensoren, oder die Vergrößerung des Nutzer- und Anwendungskontingentes.

Aus Sicht der System-Architektur können folgende zu beachtende Punkte eingeführt werden:

- **Modularität und Flexibilität** [BPP07]: Die Architektur des gesamten Systems muss so modular wie möglich aufgebaut werden, indem Komponenten für jede benötigte Funktionalität entwickelt werden. Durch die Modularität werden die Details des Kontext-Modells von der Anwendungslogik abgekoppelt (vgl. [HIM05]). Die damit gewonnene Flexibilität des Systems zeigt sich bei der dynamischen Erweiterung der Funktionalität – z.B. hinzugefügte Dienste, Geräte oder Sensoren. Darüber hinaus sollten Anwendungen als "Einsteckbausteine" angesehen werden: ihr Hinzufügen sollte für keinen Systemausfall bzw. Systemneustart sorgen (vgl. [EPR06]).
- **Transparente Verteilung** [BPP07]: Die im Kontext-Modell abzubildenden Sensoren können physisch verteilt sein, sodass sie nicht zu einem einzigen System verbinden können. Eine identische Eigenschaft prägt die Anwendungen, die das Kontext-Modell benutzen möchten – diese können auf mehreren Geräten laufen. Die Transparenz in diesem Fall bezieht sich auf die Zustellung der benötigten Kontext-Informationen zu den Anwendungen: es muss eine Trennung zwischen den Kontext-Konsumenten und den Kontext-Produzierenden (engl. separation of concerns) entstehen, sodass die Anwendungen keine (direkte) Aufsicht der Implementierungsdetails haben. In [DA00] wird vorgeschlagen, die Komponenten des Kontext-Modells unabhängig von jener Anwendung zu entwerfen.
- **Verteilte "Struktur"** [SLP04]: Als eine abgeleitete Form von verteilten Systemen, erben Ubiquitous Computing Systeme dasselbe Paradigma: der Mangel einer zentralen Instanz, eines zentralen Systems, das die Rolle des "Supervisors" übernimmt. Somit müssen kontextbewusste Systeme selbst für die Erstellung, Aufstellung sowie Wartung der einzelnen (Meta-)Daten (wie z.B. Beschreibungen des Kontextes durch ein Kontext-Modell) und Dienste zuständig sein.
- **Automatisierung** [BPP07]: Ein weiterer interessanter Aspekt des Systementwurfs wäre die Idee, den Übergang zwischen Entwurf und Aufstellung zu automatisieren. Dies kann mithilfe von z.B. Modell-getriebenen Architekturen wie MDA erfolgen. Die Vision wäre dann, eine komplette Methodologie zum generischen Entwurf und zur Bereitstellung umsetzen zu können.

Weiterhin muss ein Kontext-Modell hinsichtlich seiner Fähigkeiten beschrieben werden – welche Eigenschaften muss das fertige Modell bereitstellen, um dem angestrebten Ziel näher zu kommen. Die folgende Auflistung aus der Literatur schlägt die relevanten Konzepte vor:

- **Partielle Validierung** [SLP04]: Durch die Entwicklung eines Kontext-Modells wird nur eine Vorlage für die zu erfassende Kontext-Informationen erstellt. Aus diesem Grund müssen diese Informationen demnächst auf ihre Vollständigkeit geprüft werden, auch wenn nur partiell. Die Unvollständigkeit der Kontext-Informationen stammt aus der Verteilung des gesamten Systems, wobei

keine einzige Stelle oder sogar kein bestimmter Zeitpunkt gibt, indem sich die vollständige, interpretierte Information auf einem separaten System befindet.

- **Ausdrucksstärke** [BOQ+11]: Eine wichtige Eigenschaft des Kontext-Modells ist die Ausdrucksstärke, die durch die modellierten Elemente abgebildet worden ist. Zum einen besteht diese Eigenschaft aus dem Anfrage-Potential: die Anfragen nach Kontext-Informationen müssen complex sein und auf eine möglichst große Anzahl von Arten der Kontext-Informationen anwendbar sein. Darüber hinaus beschreibt die Ausdrucksstärke eine wichtige Komponente des Kontext-Modells und zwar die Notwendigkeit der Inferenz-Fähigkeiten. Die Entscheidungsfindung wird normalerweise durch Inferenz-Komponenten namens Reasoners geleistet: angemessene Aktionen bzw. Reaktionen und Anpassungen der Anwendung anhand der verfügbaren Kontext-Informationen müssen ausgewählt werden. Der weitere Nutzen der Reasoner-Technologie wird in [BBH+08] eingeführt, indem die Reasoners auch für die Interpretation der erfassten Daten zuständig sind (die Ableitung vom "höheren" Kontext aus rohen Daten) sowie für die Schlussfolgerung der komplexeren Nutzersituationen.

4.3.5 Weitere Anforderungen an eine Modellierungstechnik

Zusammenfassend wird ein Überblick von [WX08] auch gegeben, in Form konkreterer Anforderungen an eine Modellierungstechnik. Der Arbeit zufolge müssen verschiedene Aspekte relativ zu der Nutzung des Kontextes berücksichtigt werden:

- Eine **dynamische Integration** von Sensorik, im Rahmen einer "Smart Home" Umgebung
- Die Definition von **Dienst-spezifischen Kontext- Modellen** – je nach angebotenen oder zusammengefügten Funktionalitäten muss ein Kontext-Modell als Unterstützung bzw. Rahmen der Anpassung definiert werden
- Als letzter Aspekt soll eine **graphische Benutzeroberfläche** bereitgestellt werden, welche das Kontext-abhängige Verhalten des Systems beschrieben bzw. präsentieren wird. Dieser Schritt ist optional, da nicht alle kontextbewusste Systeme mit einer Benutzeroberfläche vorgesehen sind – z.B. Navigationssysteme basierend auf Vibrationen.

Während die ersten Anforderungen durch vorher genannte Prinzipien abgedeckt werden können, bleibt jedoch interessant, den letzten Aspekt weiter zu untersuchen. Die genannte Arbeit (bzw. [WX08]) setzt den Fokus darauf, eine geeignete Benutzeroberfläche zu entwickeln, die für die Kommunikation bzw. Darstellung von kontextbezogenen Informationen zuständig ist. Während diese Thematik im Fall von AAL Lösungen von besonderer Wichtigkeit ist, wurde die Idee in dieser Arbeit grob angewandt: in Kapitel 7 wird die Entwicklung einer spezialisierten und für erfahrene Benutzer entwickelten Benutzeroberfläche beschrieben, die für die Eingabe spezifischer Erweiterungen des Kontext-Modells zuständig ist.

Auf die oben aufgeführte Erweiterung des Kontext-Modells durch Benutzereingabe wird in [WX08] auch eingegangen: es wird ein Szenario vorgestellt, in dem der Nutzer nicht zufrieden mit dem anfänglichen Angebot des Systems an kontextbewussten Diensten ist. Zu diesem Zweck wählt der Nutzer aus einer Art "Service Marketplace / Store" weitere Dienste, die in das bestehende System zu integrieren sind. Darüber hinaus besteht in dem beschriebenen Szenario die Möglichkeit, neue Sensorik

in das vorhandene “Smart Home” System einzubringen, sodass die neu integrierten Dienste von den Fähigkeiten der Umgebung in voller Breite profitieren können.

Die Idee der “Nutzerbefähigung” (engl. user empowerment) ist ursprünglich aus der Welt der Business Intelligence (BI) entstanden , wo dieser Prinzip auch als “self-service” oder “end-user oriented” bekannt ist und in Visionen wie Business Intelligence 3.0[1] enthalten ist. Dasselbe Prinzip wird zudem in [KTP11] erläutert – in der genannten Arbeit wird der Entwurf eines Frameworks vorgestellt, in dem die “Entwicklung” von kontextbewussten Anwendungen von den Einwohnern selber durchgeführt werden kann, obwohl sie voraussichtlich keinerlei Erfahrungen mit Softwareentwicklung gesammelt haben.

Darüber hinaus wird ein anderer Aspekt in [KLN08] bezüglich der “Nutzerbefähigung” diskutiert: als eine Technologie wie z.B. eine “Smart Home” Umgebung die verbreitetete kommerzielle Einführung nähert, muss den Fokus auf die Entwicklung von Nutzer-zentrische Systeme gesetzt werden. Detailliert sollen Systeme entwickelt und geliefert werden, die idealerweise von den End-Benutzern *selbst* installiert und verwaltet werden kann. Diese Perspektive ist von besonderen Bedeutung in einem Szenario wo das System in die eigene Wohnung installiert und betrieben werden muss, wo ein Gefühl der Kontrolle entstehen muss (siehe unten). Diese Idee kann die Akzeptanz gegenüber Smart Home Lösungen verbessern (da diese Lösung die Benutzer aktiv involviert) und ist normalerweise mit einer Senkung der Kosten verbunden, aufgrund dessen, dass die Instandhaltung nicht mehr unbedingt von einem spezialisierten Techniker durchgeführt werden muss.

Weiterhin könnte diese Idee als eine mögliche Lösung für die “Inversion of Control” Problematik aus der “Smart Home” Domäne angesehen werden. Das “Inversion of Control” Problem wird in [DLY+06] erklärt: während in den üblichen, Desktop-basierten Modellen der Mensch-Maschine-Interaktion die Systeme eigentlich als “hilflose” Ausführende der menschlichen Angaben angesehen werden, verändert sich diese Situation im Fall von fortgeschrittenen kontextbewussten Systemen. Dafür muss eine passende Abwägung gefunden werden, zwischen der Erweiterung der Funktionalität des Systems und der Beibehaltung des “Kontroll-Gefühls” (engl. Sense of Control).

[1] Vgl. die kommerzielle IT-Publikation “Business Intelligence 3.0: Revolutionizing Organizational Data”, 2011, `http://www.panorama.com/resources/resource-library/`

5 Entwurf

Im Laufe dieses Kapitels werden die hauptsächlichen Merkmale der Architektur der Softwarelösung vorgestellt. Zuerst wird ein abstrahierter Überblick einzelner betroffenen Systeme gegeben. Hierfür behalten jeweils die Bezeichner “Backend” / “Server-(System)” sowie “Frontend” / “Client(-System)” ihre Gleichwertigkeit und werden abwechselnd eingesetzt.

5.1 Überblick

Die Softwarelösung besteht aus den grundlegenden Einzelteilen eines verteilten Systems und zwar: ein System spielt die Rolle des **Servers** und stellt eine Reihe von Diensten zur Verfügung, während mehrere **Clients** sich zum Server verbinden und bestimmte Operationen auf den zugänglichen Diensten ausführen.

Die Dienste werden in Form von einzelnen RESTful Web Services angeboten, die auf einem Web Server laufen und durch eine HTTP Verbindung zu erreichen sind. Ein Web Service, der die Beschränkungen von REST (siehe auch 2.7) einhält, wird als *RESTful* bezeichnet. Die Operationen, die ausgeführt werden können, lassen sich unter der Bezeichnung CRUD zusammenfassen. Das Akronym steht für “Create, Read, Update und Delete” und stellt damit die wesentlichen verfügbaren Operationen zur Erstellung und Aktualisierung sowie zum Auslesen und Löschen einer persistierten Ressource dar.

Die Verteilung in diesem Fall ist physisch, aus dem Grund, dass die Funktionalitäten des Clients und entsprechend des Servers auf verschiedenen End-Geräten ausgeführt werden: als End-Gerät für das Client kommt ein mobiles Betriebssystem wie z.B. Android infrage, im Fall des Servers wird die Windows- Plattform benutzt.

Darüber hinaus können die Systeme räumlich getrennt werden. Wie in Kapitel 1.5 erläutert, befinden sich im Normalfall das Client und der Server im selben Raum bzw. in der Ziel-Wohnung. Eine Abweichung kann entstehen, indem das mobile Client-System die Wohnung verlässt und lokal gespeicherte Daten wie z.B. die Liste der Sehenswürdigkeiten um die Wohnung herum benutzt.

Das definierte Protokoll für die Kommunikation zwischen Client und Server ist das HTTP Protokoll, mit möglicher Erweiterung auf TLS/SSL für einen erhöhten Sicherheitsgrad. Die über das Netzwerk verschickten Informationen werden vom Sender in einem bestimmten Format serialisiert und auf der anderen Seite vom Empfänger wieder deserialisiert.

Zugleich mit dem Server und Client enthält die Architektur ein Message-Oriented Middleware [Cur04] (dt. etwa Mitteilung-bezogene Zwischenanwendung) für die Realisierung des Kommunikationskanals zwischen Wohnungseigentümer und -mieter. Hiermit werden Mitteilungen verschickt und empfangen, und im Fall von begrenzter Erreichbarkeit auch persistiert.

Die Graphik 5.1 stellt die Hauptarchitektur der Softwarelösung dar. Des Weiteren wird auf die genannten Teileinheiten eingegangen und deren erweiterte Architektur vorgestellt.

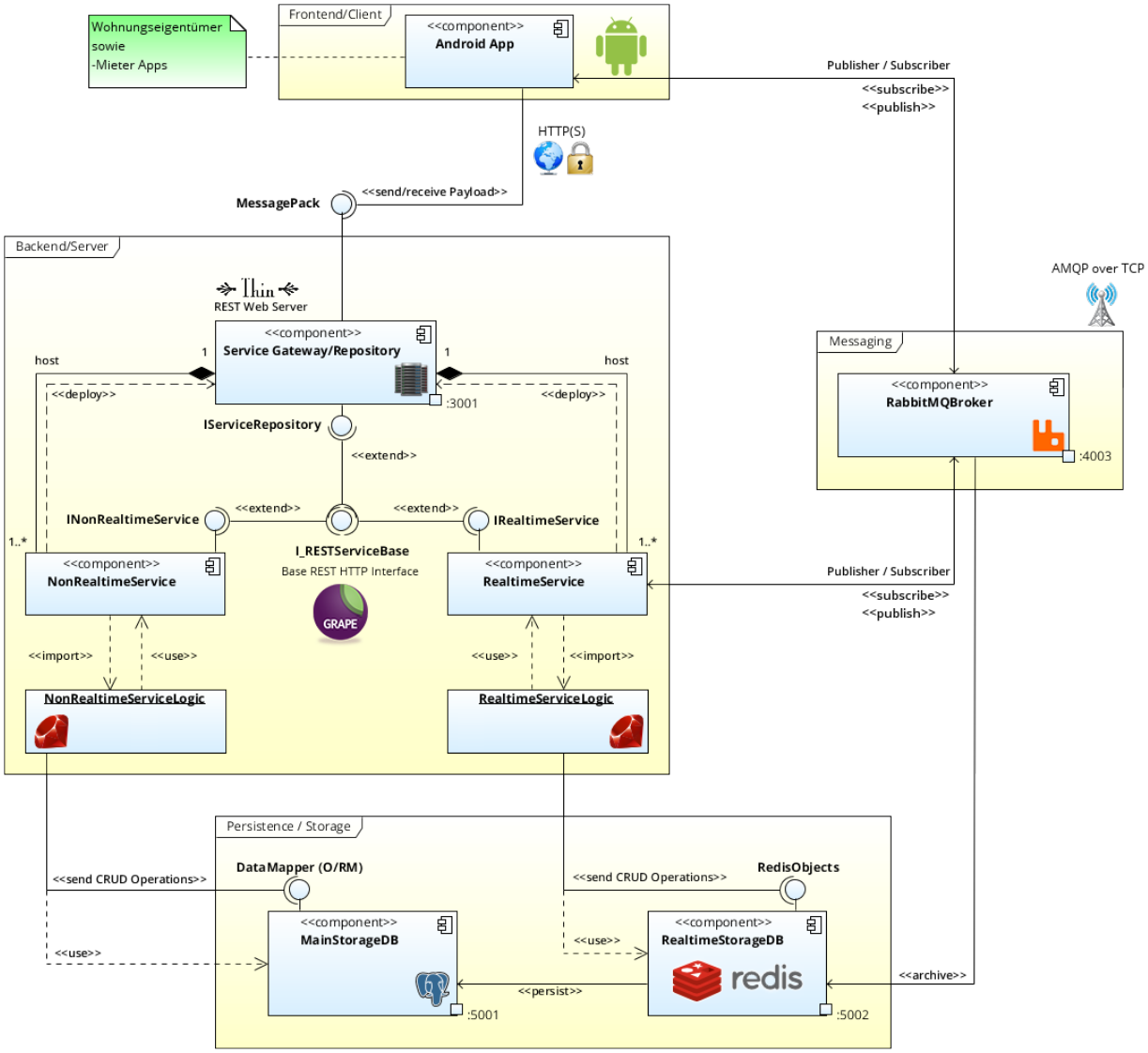


Abbildung 5.1: Überblick der Architektur

5.2 Server-Architektur und -Funktionalität

Im Folgenden werden die Bestandteile des “Server” Systems beschrieben. Es werden die bereitgestellten Dienste und weitere angebotenen Funktionalitäten wie die Datenspeicherung-Komponente sowie die Komponente für die Umsetzung des Kommunikationskanals eingeführt. Als erstes wird ein Überblick der auf dem Server-System laufenden Dienste gegeben.

5.2.1 Dienste

Ein wesentliche Aufgabe des Server-Systems ist die Bereitstellung mehrerer Dienste, die mittels einer REST-kompatiblen Schnittstelle den Clients Funktionalitäten anbietet. Jeder Dienst behält Zugriff auf mehreren Unterkomponenten des Servers, die demnächst eingegangen werden. Ein erster

Dienst ist von der Persistenz-Komponente repräsentiert, welche die vom Client entgegengenommenen Datenbestände sichert.

Speicherung des Datenbestandes

Als Hauptkomponente für die Speicherung des Datenbestandes bietet der Server eine Komponente, die Daten persistieren, bearbeiten und auslesen kann. Damit wird der Zugang zu einem relationalen Datenbanksystem ermöglicht, welches CRUD Operationen ausführen kann.

Die Schnittstelle zwischen dem Server und dem Datenbanksystem wird von einer ORM Komponente gewährleistet. ORM steht für Object-Relational Mapping und beschreibt einen Dienst, der eine beidseitige Verbindung zwischen einem bestimmten Eintrag in der Datenbank und dessen Repräsentation als Objekt bzw. Instanz in einer Objekt-orientierten Programmiersprache herstellt. Eine Schreibvariante des Akronyms ist auch O/RM, um den Unterschied zwischen Object-Relational Mapping und Object-Role Mapping deutlicher zu machen.

Vorteilhaft an der Verwendung eines ORM Dienstes ist die Einbettung gewisser Verbindungen und Eigenschaften oder Attributen der Ressource in eine einheitliche Komponente bzw. Klasse. Weiterhin können mithilfe fortgeschrittener ORM Dienste Relationen im Ganzen erstellt werden. Dies geschieht anhand einer Beschreibung der Datentypen abgesonderter Attribute und Verbindungen zu anderen Relationen des Schematas.

Ein wichtiger an dieser Stelle zu beachtenden Aspekt ist die Einkapselung der Persistenz-bezogenen Funktionen in eigenständigen REST Web Services, welche z.B. für jede Operation (Create, Read, Update, Delete) auf eine Ressource eine entsprechende HTTP-Methode bereitstellt. Dieser Aspekt ist ein Beispiel für ein *NonRealtimeService*; Details über die Umsetzung dieses Aspektes lassen sich weiter aus dem Absatz 7.3.5 entnehmen.

Aus Sicht der Datenhaltung kann der Server auch Zugang zu einer nicht- relationalen Datenbank bereitstellen. Diese Datenbank ist für den Umgang mit sehr schnell einkommenden, sogenannten “Echtzeit” Daten relevant. Wie im Fall der relationalen Datenbank wird eine Schnittstelle verwendet, mit dem Zweck, die Ausführung von CRUD Operationen zu erleichtern. Die Softwareprodukte, die für die Umsetzung der Datenhaltung, werden in 7.3.1 benannt.

Während die interne Kommunikation zwischen dem Server und den einzelnen für die Datenhaltung zuständigen Teilsystemen durch ORM Dienste realisiert wird, lässt sich die eingehende Kommunikation (vom Client zum Server) mittels des genannten Kommunikationsprotokolls, nämlich HTTP, realisieren. Jeder Dienst kann nach außen eine Schnittstelle definieren, in Form eines REST Web Services, in eine Art und Weise die im kommenden Paragraph beschrieben wird.

REST Web Services

Die im Abschnitt 2.7 eingegangenen Konzepte hinsichtlich REST-Technologien sind in der Architektur der entwickelten Softwarelösung enthalten (Bereich “Backend/Server”). An dieser Stelle werden mehrere *Services* definiert, die durch REST ansprechbar sind und somit einen SOA-ähnlichen Konzept der Service-Orientierung beschreiben.

Eine Service-Orientierte Architektur (engl. Service-Oriented Architecture, SOA) beschreibt eine Software-Architektur, die aus mehrere lose gekoppelte Dienste oder Services besteht. Somit werden die einzelnen Bestandteile und Funktionalitäten der beschriebenen Anwendung in selbständige

Dienste geteilt. Für die Kommunikation zwischen der Anwendung und den Services oder zwischen den einzelnen Services werden Standard-Schnittstellen wie z.B. REST eingesetzt.

Bezüglich der von den Diensten bereitgestellten Schnittstelle wird im Fall dieser Architektur eine Schnittstelle definiert, die die grundlegenden REST Methoden enthält, die von jedem Dienst bereitgestellt werden müssen (siehe auch 7.3.5). Darüber hinaus definiert jeder Dienst eine eigene erweiterte Schnittstelle, um die eigene Funktionalität bereitzustellen.

Die verfügbaren Dienste können in einem "Repository" oder "Gateway" gelagert werden und mittels einem Web Server nach außen zu den Clients bereitgestellt. Ein Vorteil einer SOA-ähnliche Lösung ist, dass die Dienste individuell auf verschiedenen Systeme gelagert werden können, um z.B. eine bessere Fehlertoleranz zu erzielen. Im Rahmen des Architektur-Konzeptes dieser Arbeit werden die Dienste in einem Gateway zusammengepackt und mithilfe eines hochperformanten Servers zur Verfügung gestellt, wie im Detail im Laufe des Abschnittes 7.3.5 dargestellt wird.

Weiterhin wird zwischen "normale", nicht-Echtzeit fähige Dienste und Echtzeit-fähige Dienste unterschieden, indem die dafür zuständige Komponente für die Zwischenspeicherung oder Persistierung (siehe auch vorheriger Paragraph) unterschiedlich ist. Während sich ein *NonRealtimeService* mit der relationalen Datenbank in Verbindung setzen kann, spricht hingegen ein *RealtimeService* die Echtzeit-Datenbank an. Ein weiterer Unterschied zwischen den zwei Arten einbezogener Diensten wäre, dass die deklarierten Schnittstellen unterschiedlich sind. Des Weiteren verwenden beide Arten von Diensten eigene "Unterkomponenten", die die verschiedenen Persistenz-bezogene Programmlogik einkapseln.

Ein weiterer Aspekt bezüglich der Bereitstellung von REST Web Services, nämlich die Entdeckung vorhandener Dienste, wird im darauffolgenden Abschnitt behandelt.

Entdeckung und Beschreibung vorhandener REST Web Services

In [KTP11] wird die Idee eingeführt, dynamisch eine maschinenlesbare Darstellung vorhandener RESTful Services zu erzeugen. Dies würde dazu dienen, einen einheitlichen und standardisierten (REST-basiert) Datenaustausch mit den aufgelisteten Services zu ermöglichen. Ein Hilfsmittel für diese Darstellung wird von der vorgeschlagenen WADL[1] Spezifikation repräsentiert. WADL (Web Application Description Language) ist eine Sprache zur Beschreibung von Web Anwendungen, die im Jahr 2009 zwecks Standardisierung bei W3C vorgeschlagen worden ist. Jedoch ist WADL noch nicht standardisiert.

WADL ist ein XML-basiertes Dateiformat, das eine Plattform- und Programmiersprache-unabhängige Beschreibung von HTTP-basierten Web Anwendungen anbietet. Eine verbreitete Perspektive von WADL schildert dies als das RESTful Äquivalent von WSDL [KTP11]. Obgleich die Beschreibung von REST-basierten Web Services möglich[2] ist, stellt sie sich als eine zu schwerfällige Annäherung dar.

In dieser Arbeit wurden die Prinzipien von WADL in einer "verkürzten" Form übertragen: mithilfe des häufig übersehenen HTTP Verbs namens **OPTIONS** und die benutzten Methoden zur Beschrei-

[1] `http://wadl.java.net/`

[2] Artikel von IBM: `https://www.ibm.com/developerworks/webservices/library/ws-restwsdl/`

bung von URIs aus der Implementierung der Rack[3] Schnittstelle können Beschreibungen vorhandener REST Services (auch als engl. Endpoints bekannt) dynamisch erstellt werden.

Rack ist eine Softwarelösung, die eine minimale, modulare API für die Verbindung von Ruby-unterstützenden Web Servers zu Ruby-basierten Web Frameworks sowie zu sämtlichen Middleware-Komponente anbietet. Mittels Rack werden HTTP Anfragen in einer sehr einfachen, uniformen API (im minimalster Form, auch durch eine einzige Zeile) gekapselt.

Als Inspiration für die Erstellung des Entdeckungskonzeptes diente die neue Technologie namens RESTdesc[4], die die maßgebliche REST Beschreibung von Fielding [Fie00] insb. hinsichtlich der Benutzung von Hypermedia sowie vorhandener *Semantic Web* Technologien[5] wie Notation3 (N3), Konzepte aus DBPedia oder Reasoners verwendet.

Weiterhin zeigen aktuelle Forschungsrichtungen wie [LG12], dass die fehlende Unterstützung von Hypermedia und die Benutzung vieler verschiedener Repräsentationen der Ressourcen (was nicht mit den "RESTful" Prinzipien übereinstimmt) zu zerbrechlichen Architekturen führt. Damit werden vorhandene REST Architekturen nicht mit der Datenexplosion, die mit der Entwicklung des Internet of Things Konzeptes zu erwarten ist, Schritt halten können. In der genannten Arbeit wird JSON-LD vorgestellt: eine Repräsentation der Ressourcen, die Konzepte der "Semantic Web" Welt unterstützt, als eine mögliche Methode, ein wahrhaftes RESTful Web Service zu entwickeln. LD steht für Linked Data und stammt ebenso aus der Welt der "Semantic Web" Technologien. Die Umsetzung dieses Konzeptes wird in voller Breite weiter in Kapitel 7.3.5 erläutert.

Die eingeführten Einzelteile der Architektur beschrieben sicherheitskritische Aspekte wie z.B. die Austausch von HTTP-Mitteilungen, welche Änderungen der gespeicherten Daten erlauben können. In dieser Hinsicht müssen Konzepte zur Datensicherheit und Sichere Kommunikation entworfen werden. Diese Beschreibung erfolgt anhand des nächsten Abschnittes.

5.2.2 Sicherheitskonzepte

In dem vorher eingeführten Paragraph 1.5.5 wurden die grundlegenden Sicherheitsziele vorgestellt, die ein zu einem erhöhten Sicherheitsgrad führen können. Im Folgenden werden diese Prinzipien aus Sicht des Entwurfs und mit minimalen Implementationsdetails vorgestellt.

Vertraulichkeit

Vom oben nach unten in der Hierarchie "Backend – Anwendungen (Frontend) – Datenbank" müssen folgende Sicherheitsmaßnahmen eingehalten werden: erstens muss das Server-System mit einer starken Administrator-Passwort abgesichert werden, sodass den direkten Zugang zum Server möglichst vermeidbar ist. Weiterhin sollte die serverseitige Anwendung (das Backend) keine direkte Passwörter (z.B. in Quelltext eingebetteten Eingaben) direkt speichern, sondern diejenige benötigten Passwörter aus dem Dateisystem des Servers auslesen. Die Umsetzung dieses Prinzips wird im Abschnitt 7.3.1 erläutert.

[3] `http://rack.github.com/`
[4] RESTdesc Konzept: `http://restdesc.org/about/concept`
[5] Beispiel zum gesamten Durchlauf eines Anwendungsfalles mit RESTdesc: `http://notes.restdesc.org/2011/images/`

Diese beiden Anforderungen können mithilfe von der Rechtenverwaltung des Betriebssystems geregelt werden (eingeschränkte Rechtevergabe auf Benutzer- und Dateiebene). Letztendlich im Fall der Vertraulichkeit ist auch wichtig, die Absicherung der Datenbank selbst mit einem sicheren Superuser oder Administrator-Passwort durchzuführen. Starke Passwörter müssen für die einzelnen Datenbank-Benutzer auch ausgewählt und benutzt werden. Die Implementierung dieser Aspekte wird weiter im Paragraph 7.3.1 beschrieben.

Teilnehmerauthentizität

Dieses Sicherheitsziel könnte zuerst von dem Dienstleister gewährleistet werden: z.B. die Authentizität einer Buchung wird vorher geprüft, indem ein Login-Verfahren erforderlich ist. Dazu könnte ein gesichertes Bezahlungsverfahren garantieren, dass die Teilnehmer authentisch sind. Dadurch wird mit einer hohen Wahrscheinlichkeit die tatsächliche Identität des Wohnungsmieters festgestellt; dasselbe Prinzip kann im Fall des Wohnungseigentümers eingesetzt werden.

Trotzdem wird im Fall dieser Softwarelösung ein Signatur-Verfahren eingesetzt: der Wohnungseigentümer generiert für jeden seinen Gästen bzw. Gästengruppen eine kryptographische Wert (z.B. ein Hash), die zu den Wohnungsmietern nach der Buchung oder bei der Anreise verteilt wird. Damit lässt sich ein Teilnehmer bzw. ein Wohnungsmieter innerhalb der Softwarelösung identifizieren. Darüber hinaus könnte die Idee weiterentwickelt werden, indem eine bestimmte Hash-Wert nach einem gewissen Zeitraum ablaufen kann (durch z.B. die Verwendung eines Zeitstempels für die Zeit der Generierung). Die Details bezüglich der Umsetzung dieser Idee sind im Paragraph 7.3.5 zu entnehmen.

Nachrichtenauthentizität

Dieses Ziel wird mithilfe von “Message Authentication Codes” (MACs) erreicht. Bekannt aus dem TLS/SSL Protokoll ist die Verwendung von MACs, die auf kryptographischen Hashfunktionen basieren. Ein Beispiel an dieser Stelle wäre HMAC, die als grundlegender Prinzip der Teilung einer privaten Schlüssel zwischen den in der Kommunikation beteiligten Parteien eingesetzt wird.

Im Fall der aktuellen Softwarelösung wird jeden Austausch zwischen Client und Server mittels einer HMAC signiert und an dem Empfängerseite überprüft. Somit wird sichergestellt, dass eine HTTP Anfrage, die eventuelle Schäden zufügen kann (wie z.B. das Löschen bestimmter Daten), nur von berechtigten Teilnehmer ausgeführt werden kann. Weitere Details werden im Abschnitt 7.3.5 diskutiert.

Datenintegrität

Die vorherige Umsetzung des Prinzips “Nachrichtenauthentizität” nennt Message Authentication Codes als einen verwendeten Baustein. Zusätzlich zu der Etablierung der Nachrichtenauthentizität dient eine MAC zudem zur Gewährleistung der Datenintegrität. Dafür wird zusätzlich bei der Benutzung des HMAC “Signaturverfahrens” eine andere Hashfunktion für die Bestimmung der Datenintegrität der zu verschickenden Mitteilung generiert und mitversendet.

Wenn durch die erneute Berechnung der Hashfunktion-Wert für die entgegengenommenen Daten keinen Unterschied zu der mitverschickten Wert zu bemerken ist, dann ist bei der Übersendung der Mitteilung keine Manipulation entstanden und somit gilt die Nachricht als erfolgreich durch den unsi-

cheren Kanal “korrekt” bzw. unmanipuliert empfangen. Die mit diesem Sicherheitsziel verbundenen Umsetzungsdetails werden im Absatz 7.3.5 präsentiert.

Absicherung der Datenübertragung

Ein weiterer Punkt einer Sicherheitspolitik insbesondere im Fall von Verteilten Systemen ist die Absicherung der Datenübertragung zwischen den einzelnen Endpunkten (engl. endpoints). Eine Methode zur Erreichung dieses Zieles ist die Verwendung einer abgesicherten Verbindung – hierfür kann TLS (Transport Layer Security), ein kryptographischer Protokoll zur Absicherung der über ein nicht gesichertes Netz entstehenden Kommunikation.

TLS wird auch im Fall des HTTPS (Hypertext Transfer Protocol Secure) eingesetzt und diente als eine weitere Inspiration dieser Arbeit – die in der TLS Spezifikation beschriebene Methode zur Nachrichtenauthentifizierung mittels HMAC wurde auch in dieser Arbeit eingesetzt (siehe auch Abschnitt 7.3.5). Die Verbindung zu den Datenbank-Systemen, zum Broker des Kommunikationskanals sowie zum Frontend (Client) können per HTTPS abgesichert werden, sodass eine sichere Kommunikation der z.B. Personen-bezogenen Daten entstehen kann.

Data Liberation

Das im Abschnitt 1.5.5 vorgestellte Projekt namens “Data Liberation” hat diese Arbeit dazu inspiriert, auf die gespeicherten Daten (insbesondere die Wohnung- und Ortsbezogene Daten, sowie persönliche Daten derjenigen Mieter und Eigentümer) zu achten. Es wird damit ermöglicht, die Daten mithilfe einer zur Zeit Kommandozeile gesteuerten Anwendung exportieren zu lassen – dieses Verfahren wird näher im Paragraph 7.3.1 eingegangen.

Ein weiterer Aspekt zur Vermeidung der Vorratsspeicherung ist die automatische Vernichtung aller personenbezogenen oder verallgemeinert persönlichen Daten eines Eigentümers und dessen Mieter. Dieser Vorgang könnte so eingestellt werden, dass es beim Eintritt eines spezifischen Ereignisses (beispielsweise nachdem ein Gast ausgecheckt ist) ausgelöst wird. Diese Idee ist nur auf einem theoretischen Niveau geblieben und könnte beispielsweise mithilfe des *Triggers* Konzeptes aus der Welt der relationalen Datenbanksverwaltungsystemen gelöst werden.

Nachdem die wesentlichen Komponente des Server-Systems erläutert worden sind, kommt gelegen eine zweite Komponente einzuführen. Diese Komponente stellt ein Kommunikationskanal zwischen den Wohnungseigentümer und den Wohnungsmietern bereit und kann zudem auch für die Kommunikation zwischen kontextbewussten Anwendungen oder die Verteilung von Kontext-Informationen verwendet werden. Die Beschreibung dieser Komponente folgt im anschließenden Paragraph.

5.2.3 Entwurf des Kommunikationskanals

Eine weitere Funktionalität des Servers ist die Bereitstellung einer technischen Architektur für die Distribution von Mitteilungen, die von einem oder mehreren Erzeuger (engl. producer) zu einem oder mehreren Verbraucher (engl. consumer) verschickt werden. Diese verallgemeinerte Ansicht von Erzeuger und Verbraucher lässt sich in mehreren Szenarien benutzen. Im Rahmen dieser Arbeit können folgende Anwendungsfälle angesehen werden:

- (Echtzeit) Kommunikation zwischen Wohnungseigentümer und -mieter

- Verteilung von Kontext-Informationen, von einem Sensor in der Umgebung, zu einem zentralen System oder anderen Sensoren

- Kommunikation zwischen kontextbewussten Anwendungen

Zur Unterstützung dieser Anwendungsfälle wird das AMQP-Netzwerkprotokoll eingesetzt, wie weiter im 7.3.4 vorgestellt.

Advanced Message Queuing Protocol (AMQP) ist ein standardisiertes, frei benutzbares Netzwerkprotokoll, welches die Kommunikation zwischen einer Anwendung und ein Server-System (ein "Broker") ermöglicht. Die Bedingung dafür ist natürlich, dass sowohl die Anwendung als auch das Broker dem beschriebenen AMQP-Standard übereinstimmen.

Der offene Standard AMQP wurde 2004 von dem Finanzinstitut namens JPMorgan Chase eingeführt als eine Lösung für Message Queuing (MQ)-bezogene Bedürfnisse und Topologien. Das *Message Queuing* Konzept wurde im Jahr 1985 entwickelt und lässt sich anhand der ursprüngliche Beschreibung – ein "Software Datenbus" (später als *The Information Bus* (TIB) bekannt), die Information von einer Anwendung zu einer anderen interessierten Anwendung übermittelt. Durch dieses Konzept wurde die häufig eintretende Problematik der Integration verschiedener Softwarelösungen angegangen. Darüber hinaus sollte die Lösung eine immer steigende Belastung an Mitteilungen aufrechterhalten [VW12]. An dieser Stelle kann das vorhersehbare Beispiel einer Anwendungsfall für AMQP gegeben werden: der Handel mit Aktien oder Trading.

Aus diesem Grund ist die geleistete Interoperabilität ein nennenswerter Aspekt des AMQP-Protokolls – es können AMQP-konforme Systeme miteinander kommunizieren, trotz technologischer Heterogenität (z.B. andere Ziel-Plattformen, Programmiersprachen). Darüber hinaus lassen sich beliebige Typen von Mitteilungen (engl. payload) verschicken, wie z.B. ein JSON-Objekt, eine String-Wert, binäre Inhalte oder sogar Medien [VW12].

Erforderlich für die Kommunikation ist die Präsenz einer Bezeichnung (engl. label), die den Namen des Exchange (s.u.) enthält. Die Aufgabe des Brokers ist mögliche Verbraucher anhand dieser Bezeichnung auszuwählen und sie über die Mitteilung zu informieren. Die Benennung eines bestimmten Empfängers vom Erzeuger ist im Vergleich zum TCP-Protokoll daher *nicht* benötigt [VW12]. Ein Empfänger wird wie gesagt vom Broker ausgewählt.

Aus Sicht der Architektur wird ein "Broker" (auch als Message-Oriented Middleware bekannt) benötigt, die die Kommunikation verwaltet. Die Verwaltung (bzw. die Distribution oder Routing, sowie die Persistenz der nicht zustellbaren Mitteilungen) hat als Basis verschiedene AMQP-spezifische Konzepte, die in der Abbildung 5.2 zusammengefasst sind.

Abbildung 5.2: Zentrale Konzepte des AMQP-Netzwerkprotokolls.
Quelle: Angelehnt an `http://rubyamqp.info/articles/getting_started`

Die Konzepten lassen sich kurz folgendermaßen veranschaulichen: ein oder mehrere Erzeuger (Publisher) verbinden sich mit dem Broker und verschicken mehrere Mitteilungen mittels AMQP, die für einen bestimmten "Posteingang" oder "Poststelle" (Exchange) gemeint sind. In der "Post" werden Kopien dieser Mitteilungen zu den entsprechenden "Abonnenten" (die Verbraucher) anhand spezifischer Regel auf den Weg (Route) zu dem für einen bestimmten Verbraucher (Consumer) zuständigen "Postfach" (Queue) geliefert.

Ein wesentlicher Unterschied zwischen AMQP und anderen Protokollen für den Austausch von Mitteilungen (Instant Messaging, IM) wie z.B. XMPP (Extensible Messaging and Presence Protocol, ein Netzwerkprotokoll für den Austausch von Mitteilungen, auf XML basierend) ist folgende: wegen der Abkoppelung der wesentlichen an der Kommunikation teilnehmender Parteien (die Erzeuger und die Verbraucher) lässt sich das Konzept von "Präsenz" (engl. presence) im Fall vom AMQP nicht umsetzen. Trotzdem lassen sich dafür andere Systeme anwenden, wie in Kapitel 7.3.4 erläutert wird.

Weiterhin besteht einen anderen wesentlichen Vorteil vom AMQP darin, dass verschiedene Kommunikationsmuster umgesetzt werden können: die Distribution kann in einer "Broadcast" Art und Weise erfolgen (1 Erzeuger erreicht ein oder mehrere Verbraucher), was bei Protokolle für Instant Messaging nicht erfolgt – diese unterstützen den normalen eins-zu-eins Kommunikationsmuster [VW12].

Im Kapitel 7.3.4 werden Details über die prototypische Anwendung des AMQP-Protokolls im Rahmen der Arbeit gegeben. Dies erfolgt mit der Unterstützung des Brokers namens RabbitMQ und dessen spezifische Erweiterungen des AMQP-Protokolls.

5.3 Client-Architektur und -Funktionalität

In den kommenden Abschnitten wird die allgemeine Architektur der Client-Anwendung eingegangen. Diese richtet sich nach den Prinzipien, die vom Android Software-Plattform für die Entwicklung eigener Android-Anwendungen (Kurzform: Android Apps) bereitgestellt werden.

5.3.1 System-Architektur des Android Plattform

Android ist keine Hardware-Plattform, sondern repräsentiert eine Software-Plattform (auch engl. Software Stack) oder eine Software-Umgebung, die für mobile Geräte entwickelt worden ist. Als Plattform bietet Android umfangreiche Funktionalitäten, wie z.B. ein Linux-basiertes Betriebssystem, eine komplette Lösung für die Gestaltung der Benutzeroberfläche, End-Benutzer Anwendungen, eigene Frameworks für die Entwicklung dessen, Unterstützung von Medien und viele andere Fähigkeiten [ACS09].

Ein interessantes Eigenschaft der Android Plattform ist die folgende: es besteht keinen Unterschied zwischen mitgelieferten Anwendungen und End-Benutzer Anwendungen, die (von Dritten) mithilfe des Software Development Kits (SDK, eine Kollektion von Tools und spezifische Anwendungen, die benutzt werden können, um ein eigenes Softwareprodukt zu erstellen) entwickelt worden sind und auf dem End-Gerät installiert worden sind. Es können aus dieser Hinsicht fortgeschrittene, leistungsstarke Anwendungen geschrieben werden, die die Ressource des End-Geräts zu Nutze machen [ACS09]. Der Zusammenhang zwischen den Bestandteilen des Android Plattform und das entsprechende End-Hardware Plattform (die Laufumgebung) werden in der Abbildung 5.3 dargestellt.

Ein weiterer Vorteil der Plattform ist ihre freie Verfügbarkeit: Android ist open-source Software – aus diesem Grund lassen eventuell fehlende Zusammenhänge oder Funktionalitäten der ursprünglichen Software-Paketen aus der Welt der frei verfügbaren Software erweitern.

Abbildung 5.3: Systemarchitektur der Android Plattform. Quelle: Angelehnt an `http://developer.android.com/about/versions/index.html` und [ACS09]

Nachdem die allgemeine System-Architektur von Android dargestellt wurde, folgt im nächsten Abschnitt eine Beschreibung der obersten Ebene, die für diese Arbeit von Bedeutung ist – zwar die allgemeine Architektur einer Android Anwendung.

5.3.2 Architektur einer Android Anwendung

Die für diese Arbeit relevanten Bausteine einer Android Anwendung sind in der Abbildung 5.4 zusammengefasst – eine Reihe von (evtl. lose) gekoppelten “Aktivitäten”, die Ressourcen benutzen und letztendlich in einer Manifest-Datei aufgelistet sind. Andere Komponente der Android Plattform sind Intent Receiver (helfen, Daten aus einer anderen Anwendung bearbeiten zu können), Services (langlebige Dienste) sowie Content Provider (helfen , Daten zu einer anderen Anwendung mitteilen zu können).

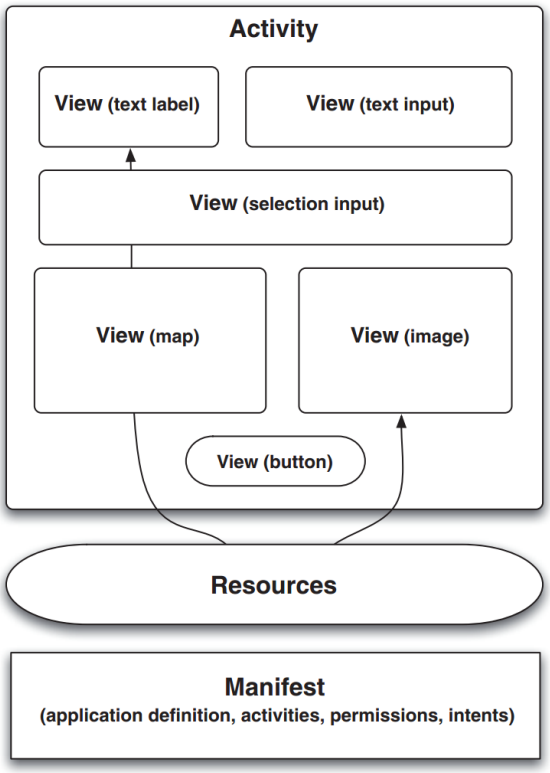


Abbildung 5.4: Grundlegende Architektur einer Anwendung für die Android Plattform. Quelle: [ACS09]

Grundsätzlich werden Activities als die wichtigsten Komponenten angesehen – eine Aktivität repräsentiert die auf dem Bildschirm des End-Geräts dargestellte Elementen. Darüber hinaus verfolgt eine Aktivität über einen festen "Lebenszyklus", die besagt wann die Aktivität dekonstruiert bzw. nicht mehr angezeigt werden soll, wann die wieder dargestellt werden soll usw. Die visuelle Ausprägung einer Aktivität wird von einem Layout repräsentiert, welche die jeweiligen Komponenten (auch als Views bekannt) enthält. Views sind das, was der End-Benutzer sieht und damit interagiert [ACS09].

Views und Layouts können bestimmte visuellen Attributen wie z.B. String-Werte, Farben, eine Theme oder sämtliche Grafiken für die Darstellung brauchen. Diese Dateien werden vom Anwendungsentwickler zur Verfügung gestellt; sie werden vorkompiliert und im "Lieferumfang" der Anwendung eingebettet, sodass die laufende Anwendung auf sie zurückgreifen kann, als Ressourcen für die jeweilige Aktivität oder View.

Ein letzter Element von Interesse an dieser Stelle ist die Architektur der Benutzeroberfläche (GUI), die von Android bereitgestellt wird. Die Android GUI ähnelt normale Java-basierte GUI Bibliotheken wie Swing oder SWT, indem ein einzelner Thread zur Verfügung gestellt wird, zusammen mit einer Bibliothek von ineinander schachtelbaren Komponenten, die auf Ereignisse aus der Laufzeitumgebung reagieren können (engl. event-driven). Die Organisation der Android GUI-Komponenten verfolgt das "MVC" Prinzip (Model–View–Controller, ein Software-Architekturmuster, welches zur Strukturierung von Software-Lösungen hilft [Wei03]) und ist in Abbildung 5.5 dargestellt:

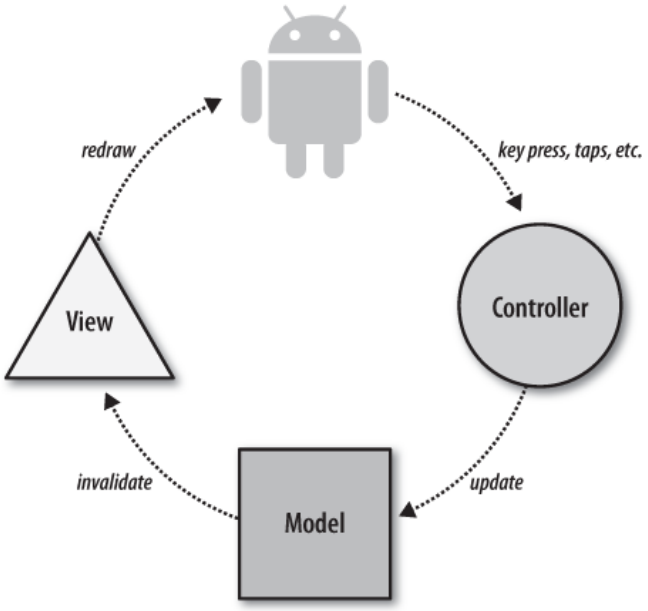


Abbildung 5.5: Model-View-Controller Architekturmuster in Android. Quelle: [RLB10]

Die von Android bereitgestellten Baukästen sind Views, die graphische Informationen auf dem Bildschirm des mobilen Gerätes abbildet sowie Controllers, die Benutzereingaben (wie Tastendrücke oder Tap-Gesten auf dem Bildschirm) behandeln.

Nachdem kurz die grundlegenden Aspekte der Architektur der Android-Plattform, dessen Anwendungen und Benutzeroberfläche eingeführt worden sind, lassen sich die im Rahmen dieser Arbeit entwickelten Anwendungen eingehen. Dies erfolgt dementsprechend im Paragraph 7.4.

Das nächste zu behandelnde Thema ist die visuelle Gestaltung der Benutzeroberfläche der entworfenen Anwendungen, welches im kommenden Abschnitte erfolgt.

5.4 Konzept zum Nutzungserlebnis (UX)

In diesem Absatz wird der Entwurf des “Nutzungserlebnisses” der zu entwickelnden Softwarelösung vorgestellt. Im Folgenden wird eine kurze Einleitung in diese breite Domäne gegeben, um danach den Fokus auf relevante Aspekte dieser Arbeit zu legen wie z.B. die mobile Benutzeroberfläche.

5.4.1 Einleitung

Als Einstiegspunkt lässt sich das Akronym “UX” entschlüsseln: es kommt aus dem Englischen “User Experience” und repräsentiert alle Facetten der Erfahrungen eines Benutzers, die während der Interaktion mit einem Produkt, einem Dienst oder einer Umgebung erfasst oder beobachtet werden können.

Das beachteter deutsch-internationale Blog “Smashing Magazine” definiert User Experience in dem Artikel “What is User Experience Design? Overview, Tools and Resources”[6] folgendermaßen: UX beschreibt wie sich ein Benutzer bei der Benutzung eines bestimmten Systems fühlt. Anscheinend abstrakt, fasst diese Definition mehrere verbundene Domänen, die in der Abbildung 5.6 bekanntlich gemacht werden.

Eine hauptsächliche Komponente des “UX” Begriffs ist *Usability* – dieser Begriff stammt aus dem Englischen und setzt sich aus zwei Worten zusammen, to use (benutzen) und the ability (die Fähigkeit). Übersetzt wird der Begriff mit Gebrauchstauglichkeit oder auch Brauchbarkeit.

Häufig werden User Experience und Gebrauchstauglichkeit als Synonyme interpretiert, obwohl sie zwei verschiedene Felder des Gebiets Mensch-Maschine-Interaktion bezeichnen. Trotzdem spielt Gebrauchstauglichkeit auch eine wesentliche Rolle in der User Experience, zusammen mit anderen Disziplinen wie z.B. Information Architecture.

Eine Übersicht der breiten Domäne von “UX” wird in der folgenden Veranschaulichung gegeben:

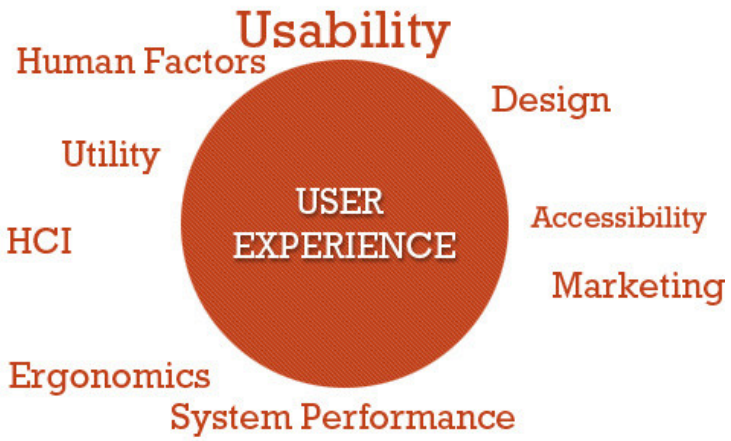


Abbildung 5.6: Überblick der verwandten Domäne im UX-Umfeld

Weitere Details über Gebrauchstauglichkeit können aus den dazugehörigen ISO Standards entnommen werden, nämlich **ISO 9241-210:2010**, “Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems” sowie **ISO 9241-11:1998**, “Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability”, wo UX und Usability in vollem Detail erläutert werden. Das einflussreiche Buch von Jakob Nielsen [Nie93] sowie ein sehr aktuelles Buch über UX [HP12] können weitere Einblicke in dieses sehr breiten Forschungsfeld geben.

Im kommenden Abschnitt wird der Zusammenhang zwischen UX und Ubiquitous Computing aus der Perspektive der “Calm Technology” Vision von Mark Weiser erläutert.

[6] Erscheinungsdatum: 5. Okt 2010, `http://uxdesign.smashingmagazine.com/2010/10/05/what-is-user-experience-design-overview-tools-and-resources/`

5.4.2 Nutzungserlebnis in Ubiquitous Computing

Nachdem der Begriff “Nutzungserlebnis” eingeführt wurde, lässt sich dies mit denjenigen Konzepten, die von Mark Weiser in der Arbeit “The coming Age of Calm Technology” [WB96] eingeführt worden sind, verknüpfen.

In der Arbeit werden mögliche Gestaltungsrichtlinien und -prinzipien eingeführt, die die Entwicklung der Anwendungen aus dem neuen Paradigma des Computing namens “Ubiquitous Computing” [Wei91] antreiben sollen. Die neuen Technologien sind zu dem Begriff “Calm Technology” (dt. etwa “beruhigende Technologie”) zusammengefasst. Eine erste Motivation bezüglich der Auswahl des Begriffs “beruhigend” wird in der Arbeit gegeben – die Evolution der Technologie führt zu wesentlichen Änderungen, was die Integration bzw. den Platz der Technologie in unseren Leben angeht. Wichtig zu bemerken ist nicht *wie* sich die Evolution entwickeln wird, sondern welche die Beziehungen zu den Nutzern sein werden [WB96].

Die neuen Anwendungen müssen zwei auf den ersten Blick widersprüchliche Anforderungen erfüllen: sie müssen “beruhigen” und informieren. Ein erster Gedanke, welches mit dem Prinzip der Beruhigung assoziiert werden könnte, ist die genannte Problematik von “Inversion of Control”. Darüber hinaus aber stellt die genannte Arbeit eine interessante These vor: die Technologien beruhigen uns, während sie unsere Peripherie befähigen – erstmal sollte sich die beruhigende Technologie einwandfrei vom Zentrum der Interaktion zur Peripherie “bewegen” können. Während die Peripherie benutzt wird, entsteht die Befähigung dadurch, dass *mehrere* Details angeboten werden, was an sich ein Gefühl der Kontrolle über die Technologie vermitteln könnte.

Damit lässt sich das Grundprinzip von “beruhigenden” Technologien formulieren: ein Ergebnis der Interaktionen mit der Technologie sollte eine Art “Sicherheitsgefühl” sein – der Benutzer muss sich “daheim”, in einem bekannten Ort fühlen [WB96].

Ein Durchlauf der “Domänen-Hierarchie” (Ubiquitous Computing – Ambient Intelligence – Smart Home, wie in Kapitel 2 erläutert) zeigt, dass die Eigenschaftswörter, die für die Beschreibung von “Ambient Intelligence” Technologien benutzt werden, sich in dem selben Raum wie die beschriebenen “beruhigenden” Technologien bewegen: außer “beruhigend”, werden auch unsichtbar und “unauffällig” benutzt [LLCS10].

Zusammenfassend beschreibt die Vision von “Calm Technology” eine Welt von unauffälligen, nützlichen und meist unsichtbar agierenden Anwendungen. Implizit wird hier ein Nutzungserlebnis beschrieben, weil die genannte Beziehung der Nutzer und dessen Gefühle (wie das erwähnte Sicherheitsgefühl) betrifft.

5.4.3 Aspekte des Nutzererlebnisses in mobilen Geräten

Im Rahmen dieses Abschnittes werden die Konzepte einer einflussreichen Arbeit [SB07] zusammengefasst. Diese Arbeit hat sich mit dem Kontext mobiler Interaktionen befasst und gibt Auskunft über die damals nur im Entstehen befindende Domäne der Gestaltung für mobile Geräte, aus Sicht einiger Strategien, die für eine optimale Gestaltung verwendet werden können.

Eine erste Rücksichtnahme im Fall eines mobilen Nutzererlebnisses ist der Kontext, in dem die Interaktionen stattfinden – dieser Aspekt bestimmt das entstehende Erlebnis völlig. Befreit von der Arbeitsplatz-verbundenen PC-Ära, finden mobile Interaktionen häufig im Alltag des Benutzers statt,

sie sind daher tief im persönlichen Kontext des Benutzers bzw. Besitzers eingebunden. Um dieser Kontext und dessen Bestandteile ausnutzen zu können, ist eine Analyse der einzeln überlappenden Ebenen erforderlich. Diese entstand in der genannten Arbeit und lässt sich anhand der Abbildung 5.7 erläutern.

Abbildung 5.7: Überblick des mobilen Kontextes, in dem mobile Interaktionen stattfinden. Quelle: [SB07]

Die Mächtigkeit des Modells überschreitet die übliche Tiefe, an der eine mobile Interaktion stattfindet und eignet sich aus dieser Hinsicht nicht für alle Arten von Anwendungen. Jedoch gibt das Modell einen Überblick für diejenige Fälle, in dem tiefgreifende Aspekte wie Kultur oder eine Vielfalt von unterstützten Aktivitäten des Benutzers wichtig sind.

Außerdem lassen sich aus der Arbeit wichtige Perspektiven mitnehmen, bezüglich zu berücksichtigender bester Methoden zur Gestaltung von mobilen Interaktionen. Diese basieren auf der Theorie und Praxis der Domäne der Mensch-Maschine-Interaktion und sind in der folgenden Auflistung zusammengefasst worden:

- **Alle mobilen Interaktionen sind Benutzer-getrieben**: Es ist eine hohe Relevanz der Anwendung gefragt, um diese nutzen zu *wollen* – wie genannt "greift" eine mobile Interaktion "ein" in die persönliche Welt des Benutzers. Deshalb müssen Inhalte und Aktivitäten gewünscht sowie angefordert werden. Ein Vergleich an dieser Stelle mit einer E-Mail Anwendung ist aufschlussreich – der Benutzer hat bei der Auswahl einer Anwendung eine wesentlich niedrigere Toleranz für "Spam".

Mit dieser Idee ist auch ein hoher Bedarf an Bedienbarkeit verknüpft: viele Kontexte sind von einer einhändig ausgeführten Aktion bestimmt; darüber hinaus erfordert die üblicherweise kleine Tastatur nur eine kurze Eingabe an. Hierfür können kurze URIs oder QR-Codes angewendet werden.

- **Neue mobile Erlebnisse konkurrieren mit alten Nutzermodellen**: Die neuen mobilen Erlebnisse bewegen sich in oft unergründete Richtungen, versuchend neue Interaktionsmodelle aufzustellen. Bei der Gestaltung ist aber zu beachten, dass die neuen Modelle mit den alten konkurrieren – dieser Aspekt spiegelt das von Weiser beschriebene "Sicherheitsgefühl" wider, indem einem eingearbeiteten Weg schwierig zu verlassen ist. Es kann ein Gemisch zwischen der neuen und der veralteten Interaktion entstehen, wie im Fall der üblichen "Einwählen" Aktion (engl. "dial"), die immer noch die Existenz eines Drehwählers unterstellt.

  Weiterhin ist an dieser Stelle die Idee der "Calm Technology" zu berücksichtigen: davon abgesehen, dass das mobile Gerät im Laufe eines Tages näher an dem Benutzer sein wird als der übliche PC, konkurriert das mobile Gerät mit vielen anderen Ereignissen, die auch eine gewisse Maß an Aufmerksamkeit verlangen. Aus diesem Grund müssen Anwendungen "abwägen", wenn die darzustellende Aktivität zum Vordergrund gerufen werden soll und wenn sie lieber in der Peripherie bzw. im Hintergrund bleiben sollte.

- **Mobile Geräte sind von der Verarbeitungsleistung nicht begrenzt**: Ein mobiles Gerät kann insbesondere im Fall eines verteilten Systems als nur ein Knoten angesehen werden – die anderen beteiligten Knoten aus dem gesamten System können oder sogar müssen über das Netzwerk angesprochen werden, um ein breiteres Angebot an Funktionalitäten gewährleisten zu können. In dem Anwendungsfall, in dem die Geräte nur als Sender und entsprechend Empfänger agieren, wird von einem "dumb terminal" gesprochen, welches nur die Funktionalitäten einer "Fernbedienung" für weitere "smart" Umgebungen oder die Funktionalität eines persönlichen Displays anbietet.

  Trotzdem kann das Szenario des reinen "dumb terminal" heutzutage übersehen werden, da die rechnerischen Fähigkeiten von mobilen Geräten in der letzten Zeit hoch angestiegen sind und noch kein Zeichen der Abbremsung geben. Ein Verknüpfungspunkt mit den im vorherigen Abschnitt vorgestellten Ideen der "Calm Technology" ist auch im Fall dieser Empfehlung zu finden: als möglicher Teil eines verteilten Systems müssen Netzwerk-Operationen ausgeführt werden, aber der Zustand dieser Operationen wird meistens nicht angezeigt. Während in [WB96] diese Mangel an Darstellung im Fall von PCs aufgrund physischer Anzeiger wie z.B. das Schwirren einer Festplatte abgelehnt wurde, muss im Fall von mobilen Geräten eine unauffällige Anzeige der Netzwerkaktivität vorhanden sein.

Außer dieser spezifischen "Best Practices" lassen sich weitere allgemeinere Empfehlungen aus [SB07] entnehmen: es muss wenn möglich ein asynchrones Interaktions- und Kommunikationsmodell verwendet werden, sodass andere Aktivitäten wie z.B. die Benutzeroberfläche nicht unterbrochen werden.

Weiterhin sollen die Interaktionen intuitiv und möglichst schnell durchführbar gehalten werden. Der Schnelligkeitsfaktor kann durch das folgende Beispiel aus [SB07] verdeutlicht werden: wenn der Benutzer sich für ein Treffen verspätet hat und sich dadurch beeilen muss, wird der Schwellenwert der Lernfähigkeit sehr niedrig sein. Jedes Hindernis auf dem Weg zum Ziel (bspw. der Treffpunkt) muss aus diesem Grund entfernt werden, sodass man schnell auf die benötigte Information zugreifen kann.

Im aktuellen Absatz wurden die von Mark Weiser genannten Prinzipien der “Calm Technology” als Hinweise für eine optimale Gestaltung des Nutzungserlebnisses betrachtet; dafür wurden einige Richtlinien in der Anwendungsdomäne der zu entwickelnden Softwarelösung bereitgestellt. Der nächste Punkt auf der Agenda wäre, diese Prinzipien anzuwenden – hierfür sind verschiedene Attrappen (engl. mockups) entworfen, die sich in der Softwarelösung wiederfinden werden und in den kommenden Paragraphen eingegangen werden.

5.4.4 Entwurf mehrerer Mockups für die Softwarelösung

Nach der Einführung von Prinzipien zur Realisierung eines (mobilen) Nutzungserlebnisses, werden diese angewandt, indem Mockups erstellt werden, die eine gute Idee über die visuelle Gestaltung des zu entwickelnden Softwarelösung geben können.

Eine kurze Definition der Bezeichnung “Attrappe” (engl. mockup) wäre die folgende: ein Mockup ist eine Form der Prototypenentwicklung, bei der ein rudimentärer Prototyp der künftigen, umzusetzenden Benutzeroberfläche eines Softwareproduktes dargestellt wird. Ein Mockup dient meistens dazu, ein Grundgerüst bereitzustellen, welches keine weitere Funktionalität enthält.

Es wurden im Rahmen dieser Arbeit insgesamt acht Mockups erstellt, die im Folgenden einzeln veranschaulicht und beschrieben werden. Eine Identifikationsnummer ist auf jedem Mockup (obenlinks zu sehen). Sie dient zur Navigation. Als Erstes wird ein “Splash Screen” (dt. Begrüßungsbildschirm) zusammen mit einem “Dashboard” (dt. etwa Instrumententafel) in der folgenden Abbildung dargestellt:

In der Abbildung 5.8 wird zuerst links der Begrüßungsbildschirm dargestellt. Ein Ladevorgang stellt fest, dass die Bedienelemente bereit sind und bestimmte Vorbedingungen (bspw. eine vorhandene Verbindung mittels WLAN mit dem lokalen Netzwerk) erfüllt sind. Nachdem die Vorbedingungen asynchron geprüft worden sind, erreicht der Benutzer den nächsten Bildschirm, nämlich das Dashboard.

Das Dashboard stellt ein Gitter mit bestehenden Aktionen bereit, die ausgeführt werden können. Die Aktionen lassen sich mit einer Benachrichtigung hinsichtlich der Anzahl ungelesener Ereignisse “beschriften”. Weiterhin ist in diesem Fall des Dashboards eine erste Anpassung am Kontext der Anwendung zu bemerken: je nachdem, welche Anwendung verwendet wird (bzw. ob Mieter- oder Eigentümer-Anwendung), passen sich die dargestellten Elemente in dem Dashboard an, indem die häufig verwendeten Aktionen einen vorrangigen Platz im Gitter befüllen.

Außerdem wird auf dem Dashboard der Zustand des Kommunikationskanals dargestellt (Präsenz-Information), gefolgt von Möglichkeiten, die Einstellungen der Anwendung oder das Profil des eingeloggten Benutzers zu verändern.

Die möglichen weiteren Aktionen, die mit einer Tap-Geste auf dem entsprechenden Kasten aufrufbar sind, werden als nächstes vorgestellt.

In der Abbildung 5.9 werden die Bildschirme abgebildet, die nach einer Tap-Geste auf dem “Neues Element” Button des Dashboards angezeigt werden. In dem linken Teil der genannten Abbildung wird ein Bildschirm für die Auswahl der Kategorie angezeigt – eine Kategorie ist eine Sammlung von weiteren untergeordneten Kategorien oder einzelne Geräte bzw. Gegenstände der Wohnung (siehe auch 7.3.1). Es werden der Name der Kategorie und dessen Beschreibung angezeigt. Mittels Tap-Gesten

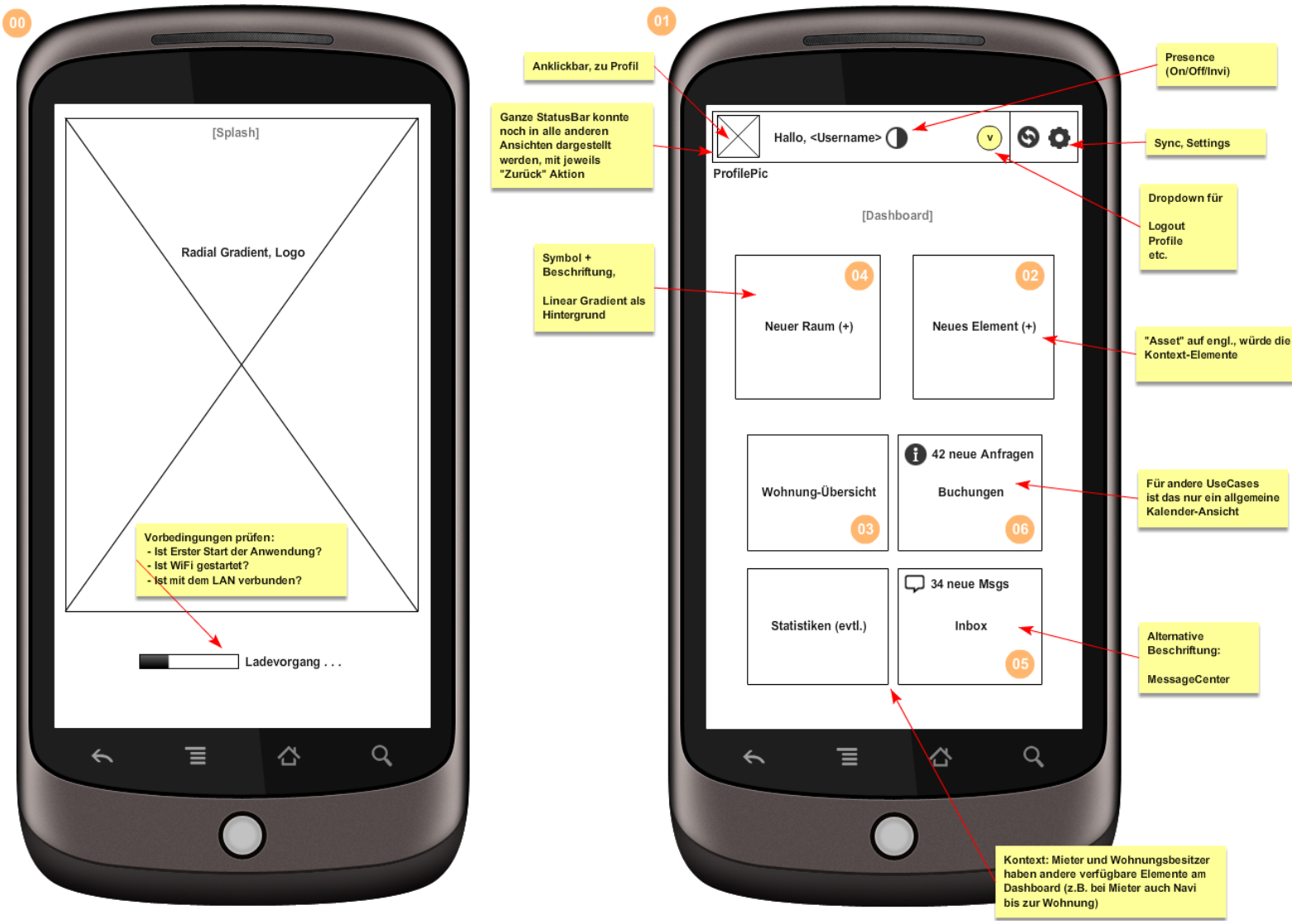


Abbildung 5.8: GUI Mockup – Splash Screen und Dashboard

gelangt der Benutzer zu der jeweiligen tieferen Ebene der Kategorie, mit einer Long-Tap-Geste wird die aktuell dargestellte Kategorie ausgewählt. Das Ergebnis der Auswahl ist die Darstellung vom linken Bildschirm.

Auf der rechten Seite der genannten Abbildung ist das Formular zum Hinzufügen eines Kontext-Elementes dargestellt. Hiermit lassen sich Daten über das Element wie ein Name oder eine Beschreibung erfassen, sowie die Feststellung erweiterter Einstellungen (Bilder zuweisen, Sichtbarkeit ändern). Weiterhin kann das Element verortet werden oder mit Attributen, Regeln und Anwendungshinweisen erweitert bzw. “angereichert” werden.

Des Weiteren befinden sich auf dem Dashboard Möglichkeiten, zu einem Wohnungsübersicht zu wechseln sowie einen neuen (symbolischen) Raum in der Wohnung hinzuzufügen – die dazugehörigen Mockups sind in der Abbildung 5.10 dargestellt:

Abbildung 5.9: GUI Mockup – Neues Kontext-Element einfügen

Die Ansicht zur Wohnungsübersicht präsentiert die Anzahl von modellierten Kontext-Elementen und weitere Überlegungen zur Darstellung: eine Art “Aktivitätsgrad” Anzeige, die z.B. von der Anzahl der erfolgreichen Indoor-Lokalisierungen bestimmt werden kann oder eine Art “Checkliste” Anzeige, welche die vorstellbaren Vorbereitungen darstellt, die zu den einzelnen Phasen des Szenarios (siehe auch 3.1.1) gehören können.

Außerdem wird auf der rechten Seite eine andere Überlegung zur Anzeige der aktuellen Lokation in der Wohnung dargestellt. Damit verknüpft ist die Aktion, welche ermöglicht einen (symbolischen) Raum anhand der aktuellen Lokation zu definieren. Diese Anzeige lässt sich noch vereinfachen, indem eine Liste mit den zu dem entsprechenden Zeitpunkt definierten Räumen dargestellt wird. Dadurch wird die Möglichkeit gegeben, einen neuen Raum relativ zu dem ausgewählten Raum, hinzuzufügen.

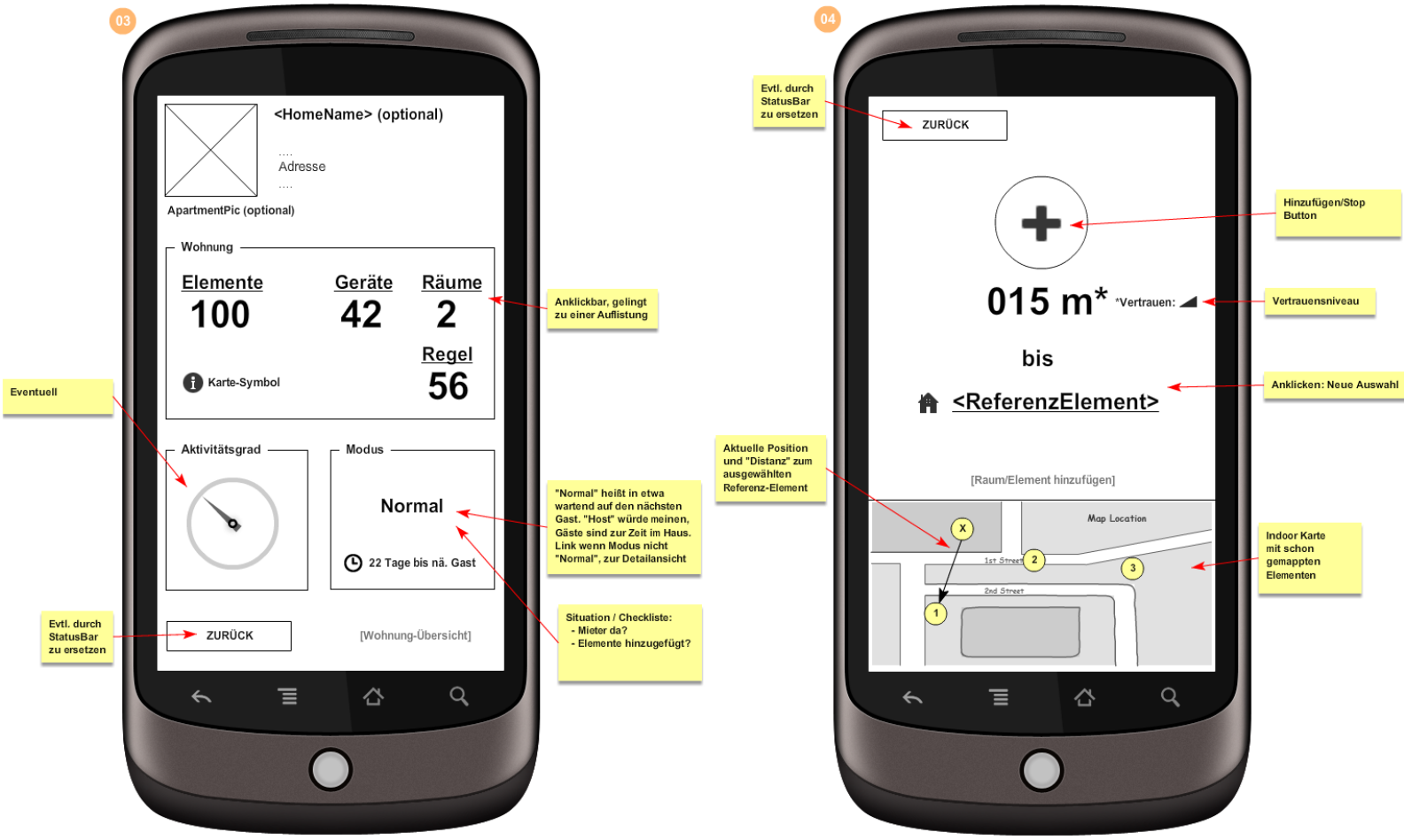


Abbildung 5.10: GUI Mockup – Wohnungsübersicht, Formular "Neuer Raum hinzufügen"

Als letztes werden weitere mögliche Ansichten vorgestellt, die die Verwendung des Kommunikationskanals sowie eine mögliche Darstellung der aktuellen Buchungen eines Wohnungseigentümers illustrieren.

Schließlich zeigt der linke Bereich der Abbildung 5.11 eine mögliche Gestaltung eines "Inbox" Konzeptes für den Kommunikationskanal, wo die einzelne Mitteilungen je nach Thema bzw. Thread sortiert werden und im Detail aufgerufen werden können. Im rechten Bereich befindet sich eine zusammengefügte Benutzerschnittstelle, die aus einem Kalender mit allen eingetragenen Buchungen sowie aus einer Übersicht der nächsten Buchungen besteht. Die Darstellung erfolgt auf der nächsten Seite.

5.4.5 Zusammenfassung

In diesem Abschnitt wurden mehrere anleitende Prinzipien bezüglich der Gestaltung des Nutzungserlebnisses eingeführt. Diese bilden die Basis zur Entwicklung einer Benutzerschnittstelle, die visuell vorgestellt wurde. Der nächste Schritt ist die Beschreibung der tatsächlichen Art und Weise, in der sich diese Prinzipien in der Softwarelösung wiederfinden könnten, ein Thema welches in dem kommenden Absatz eingegangen wird.

Abbildung 5.11: GUI Mockup – Inbox des Kommunikationskanals, Erweiterte Kalender-Ansicht für Buchungen

6 Kontext-Modellierung

In diesem Kapitel werden die zugrundeliegenden Konzepte der Kontext-Modellierung eingeführt, zusammen mit dem Entwurf und der Implementierung des im Rahmen dieser Arbeit entworfenen Kontext-Modells.

6.1 Techniken der Kontext-Modellierung

Kontext-Modellierung wird in [BPP07] definiert als eine Methode, die ein erhöhtes Abstraktionsniveau anhand Kontext-Informationen schafft. Ein gut entworfenes Modell stellt der kontextbewussten Anwendung eine essentielle Schnittstelle zur Kontext-Information zur Verfügung. Noch wichtiger wäre die Gestaltung eines verallgemeinerten Kontext-Modells, von dem mehrere Klassen von Anwendungen profitieren könnten [SLP04]. Verbunden mit der Entwicklung eines allgemeinen Kontext-Modells sind auch Gewinne aus Sicht der Softwaretechnik zu berücksichtigen: die Komplexität von kontextbewussten Anwendungen verringert sich, zugleich steigt die Wartung und das Weiterentwicklungspotential (engl. evolvability) [BBH+08].

Wie weiter in [BBH+08] erläutert, ist die Entwicklung von Anwendungen im "Ubiquitous Computing" Umfeld mithilfe von Techniken des Kontextbewusstseins ein sich entwickelnder Forschungsbereich – es wird untersucht, welche Methoden flexible und anpassungsfähige Lösungen liefern könnten, die auch eigenverantwortlich im Namen des Benutzers agieren.

Ein erster Schritt zur Evaluation spezifischer Modellierungsstrategien ist die Bereitstellung von Anforderungen, die jede Strategie erfüllen müssen. Auf diesen Aspekt wurde in den Paragraph 4.3.4 näher eingegangen.

Nachdem die Anforderungen an eine Modellierungstechnik für Kontext-Modelle aufgestellt worden sind, folgt im nächsten Abschnitt eine kurze Einführung in eine aus der Welt der Softwaretechnik bekannte Technik, zusammen mit ihren Verknüpfungspunkten mit dieser Arbeit.

6.1.1 Modellbetriebene Entwicklung eines Kontext-Modells

Unter "Model Driven Architecture" (MDA) wird eine modellgetriebene Methodologie zur Softwareentwicklung beschrieben, die als Grundprinzip eine Trennung von abstrahierten Konzepten bzw. zu entwickelnden Funktionalitäten einer Softwarelösung und die für die Entwicklung dessen erforderliche Technik anwendet[1]. Damit wird eine Softwarelösung entwickelt, die die Evolution und Änderungen des Geschäftes und der Softwaretechnologie überstehen kann.

MDA definiert eine Hierarchie von Meta-Modellen (zusammengefasst in dem Begriff Meta Object Facility, MOF), die von "M3" (Meta-Metamodell) bis "M0" (die Objekte der realen Welt) beschriftet sind, um die Abstraktionsstufen zu unterscheiden. Das Meta Object Facility (MOF) wurde vom Object Management Group (OMG) eingeführt und beschreibt eine spezielle Architektur für Metadaten. Weiterhin spielen verschiedene Technologien wie UML (Unified Modeling Language) oder XMI (XML Metadata Interchange) eine Rolle in dieser Hierarchie, welche in der Abbildung 6.1 anhand eines Beispieles zusammengefasst ist.

[1] Angelehnt an `http://www.omg.org/mda/faq_mda.htm`

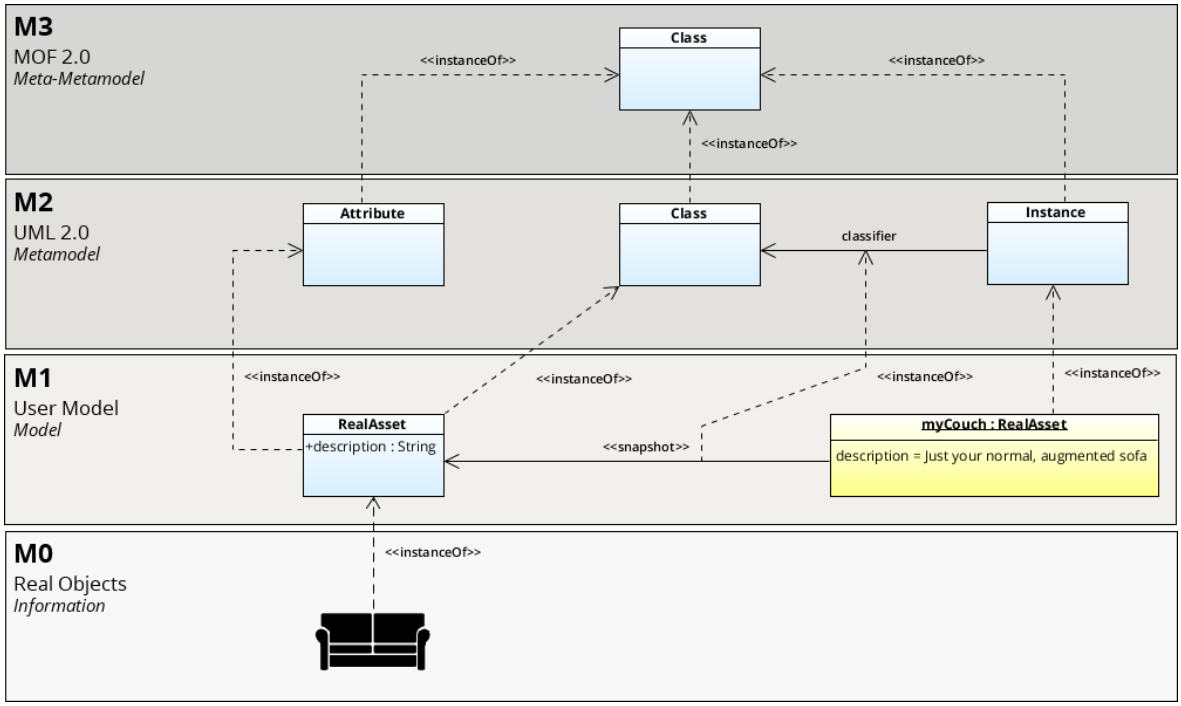


Abbildung 6.1: Überblick der Modellhierarchie von MDA. Quelle: Eigene graphische Herstellung, angelehnt an einer lizenzfreien Graphik:
`http://upload.wikimedia.org/wikipedia/commons/9/93/M0-m3.png`

Ein fundamentales Konzept der MDA-Vision ist das *Modell*. MDA definiert ein Modell als eine Repräsentation eines Bestandteils des "Diskursuniversums" – in diesem Fall das beschriebene Softwaresystem. Ein Bestandteil kann eine gewisse Funktion, eine Struktur und/oder Verhalten des Systems sein. Letztendlich dient ein Modell dazu, Fragen über das System zu beantworten, ohne eine direkte Untersuchung des Systems durchführen zu müssen [LRBA10].

Unter diesem Gesichtspunkt lassen sich dynamische sowie statische Modelle erstellen, die sich im Verlauf der Zeit nicht ändern. Im Fall von statischen Modellen wird (nur) eine "Momentaufnahme" des Systemzustandes zu einem gewissen Zeitpunkt gemacht [LRBA10]. Dafür lassen sich nur Untersuchungen in Form von "Was ist?" Fragen durchführen, während zusätzlich im Fall von dynamischen Modellen auch die Entwicklung eines Systems mitgespeichert wird und sich damit auch Fragen wie "Was ist geschehen?" (vorherige Zustände) oder "Was wäre, wenn..?" (Prognosen) beantworten lassen. Diese Modelle sind in der MDA Terminologie als "Plattform-unabhängige Modelle" (engl. Platform-Independent Base Model, PIM) bekannt.

Die letzte Art von Modellen sind die "ausführbaren" (engl. executable) Modelle, die dynamisch sind und den Zustand des modellierten Systems zur Laufzeit widerspiegeln. Der Begriff für diese Modelle ist "Plattform-spezifische Modelle" (engl. Platform-Specific Models, PSM).

Diese Vorgänge können von den im Absatz 2.8.5 beschriebenen Phasen der Kontext-Modellierung abgeleitet werden, was die Möglichkeit der Entwicklung eines Kontext-Modells mittels dem MDA begründen kann.

Der erste Vorgang in MDA wäre die Feststellung eines Meta-Modells. Im Fall der Kontext-Modellierung lässt sich ein verallgemeinertes Meta-Modell anwenden, welches in [WX08] folgendermaßen beschrieben ist:

1. **Kontext-Entität / Kontext-Element**: auf eine Entität wird durch einen Namen verwiesen. Eine Entität stellt ein physisches oder konzeptuelles Objekt dar: Beispiele von Entitäten sind Personen, Geräte oder Gebäude.

2. **Kontext-Dimension**: Weiterhin umfasst eine Dimension mögliche Eigenschaften, die zu mehreren Entitäten und Beziehungen (s.u.) zugewiesen werden können. Beispiele aus dieser Kategorie sind: ein Zeitstempel, ein gewisser Zustand oder eine Lokation.

3. **Kontext-Attribut**: Durch ein Atribut wird die Assoziation zwischen einer konkreten Eigenschaft einer Entität oder Beziehung (s.u.) und der dazugehörigen Dimension beschrieben.

4. **Kontext-Beziehung**: Eine gewisse Entität wird mit einer anderen Entität in Zusammenhang gebracht, mittels einer einseitig wirkenden Beziehung. Eine spezialisierte Art von Beziehungen ist die Verallgemeinerung.

Weiterhin kann mittels MDA die in [Wag04] gekennzeichnete Herausforderung der Kontext-Modellierung (die Definition eines Mechanismus zur Verfeinerung des Kontext-Modells, welcher vom Entwurf-Niveau bis hin zur Umsetzung seine Gültigkeit behält) bewältigt werden. Darüber hinaus stellt die genannte Arbeit eine weitere Anforderung an den Verfeinerungsmechanismus: dieser muss ermöglichen, mehrere alternative Verfeinerungen (bzw. Anpassungen des Modells) definieren zu können. Die Verfeinerungen können im Fall von MDA als weitere Bestandteile des Plattform-unabhängigen Modells (PIM) entworfen werden. Die Auswahl einer bestimmten Verfeinerung würde dann anhand der aktuellen Systemparameter (beispielsweise wahrgenommene Werte aus der Umgebung) erfolgen.

Mit Hilfe des vorgestellten Meta-Modells können konkrete Ausprägungen bzw. Kontext-Modelle definiert werden – um dieses Prinzip zu erläutern, wird in [WX08] die Abbildung 6.2 als ein Beispiel eingeführt:

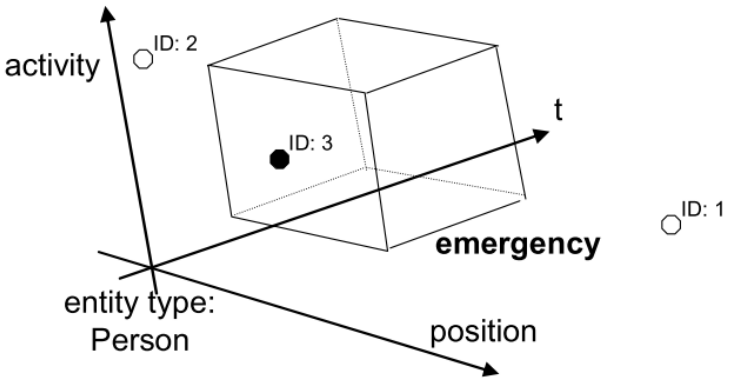


Abbildung 6.2: Beispiel eines konkreten Kontext-Modells als Unterraum des durch das Kontext-Metamodell umgefassten, konzeptuellen Raumes. Quelle: [WX08]

In der Abbildung 6.2 lässt sich ein Kontext-Modell für außergewöhnliche Situationen (*Emergency*) erkennen, die durch die Entität *Person* sowie die Dimensionen *Aktivität*, *Position* und *Zeit* bestimmt werden; einzelnen ausgewerteten Situationen wird jeweils eine Identifikationsnummer gegeben. Ein weiteres Beispiel wäre die Definition einer Beziehung, die ein Treffen zwischen zwei *Person* Entitäten beschreiben kann – verknüpft sein müssen an dieser Stelle die zwei Entitäten über ein Attribut, welches auf dieselben oder relativ gleiche Ausprägungen der entsprechenden Dimension *Position* achtet.

Zusammenfassend erlaubt MDA die Verwendung einer technischen Architektur, womit Kontext-Modelle erstellt werden können. In [LRBA10] wird ein Beispiel einer modellgetriebenen Entwicklung eines Kontext-Modells für "Smart Environments" ausführlich beschrieben.

Aus diesem Abschnitt zu entnehmen ist die Möglichkeit einer völlig modellgetriebenen Entwicklung eines Kontext-Modells, basierend auf der MDA-Methodologie. Die Möglichkeit, von einem UML-entwickelten Modell eine zur Laufzeit benutzbare Umsetzung benutzen zu können, war für diese Arbeit interessant und dessen Nützlichkeit wurde auch im Laufe der Literaturrecherche bestätigt: in [CWGN11] wird über die umfassende Technik der "Context Modeling Language" (CML) diskutiert, bei der ein in dem ein konzeptuelles Schema automatisch in eine relationales, persistierbares Datenbanksschema umgewandelt wird, um das Datenmanagement der Kontext-Informationen in einer Datenbank zu ermöglichen.

In dieser Hinsicht wurde in dieser Arbeit danach gestrebt, einen möglichst automatisierten Vorgang zu entwickeln, welcher eine schnelle Anpassung der zu persistierenden Bestandteile eines konkreten, domänenspezifischen Kontext-Modells auf Datenbank-Ebene gewährleistet. Dieser Vorgang wird weiter in Kapitel 7 beschrieben.

Nachdem die Bausteine für die Definition eines Kontext-Modells eingeführt sind, wird das im Rahmen dieser Arbeit entworfene Kontext-Modell vorgestellt und im Detail erläutert.

6.2 Definition eines Kontext-Modells

Im Laufe dieser Arbeit wurde ein Kontext-Modell entwickelt, das als Zielsetzung die Beschreibung verschiedener Anwendungsfälle einer kontextbewussten Anwendung hat. Zu diesem Zweck wurde versucht, die im Absatz 4.3.4 eingeführten Anforderungen umzusetzen, sodass die in der Forschungsfrage aufgelisteten Kriterien beibehalten bzw. umgesetzt werden können, wie z.B. die Allgemeinheit des Kontext-Modells und dessen Umgang mit Echtzeit-Daten. Abbildung 6.3 gibt einen Überblick über das Kontext-Modell und die Beziehungen zwischen den einzelnen Bestandteilen.

Hinsichtlich der Architektur kann das entwickelte Kontextmodell in zwei Teilen aufgeteilt werden: ein *Modell*, bestehend aus Kontext-Informationen, deren Metadaten, geographische Daten sowie leichtgewichtige Services und eine *Service-Schicht*, die aus Dienste besteht, die für die Distribution, Feedback/Generierung und Anfrage-Fähigkeiten des Modells zuständig sind. Darüber hinaus ermöglicht das Kontext-Modell die Definition von kontextbewussten Anwendungen, die das Modell und die Service-Schicht benutzen können und somit eine Kontext-Anpassung realisieren. Nachdem die Darstellung einer kontextbewussten Anwendung im Kontext-Modell vorgestellt wird, werden die restlichen beiden Teilen näher eingegangen hinsichtlich ihrer Komponente und Funktionalität.

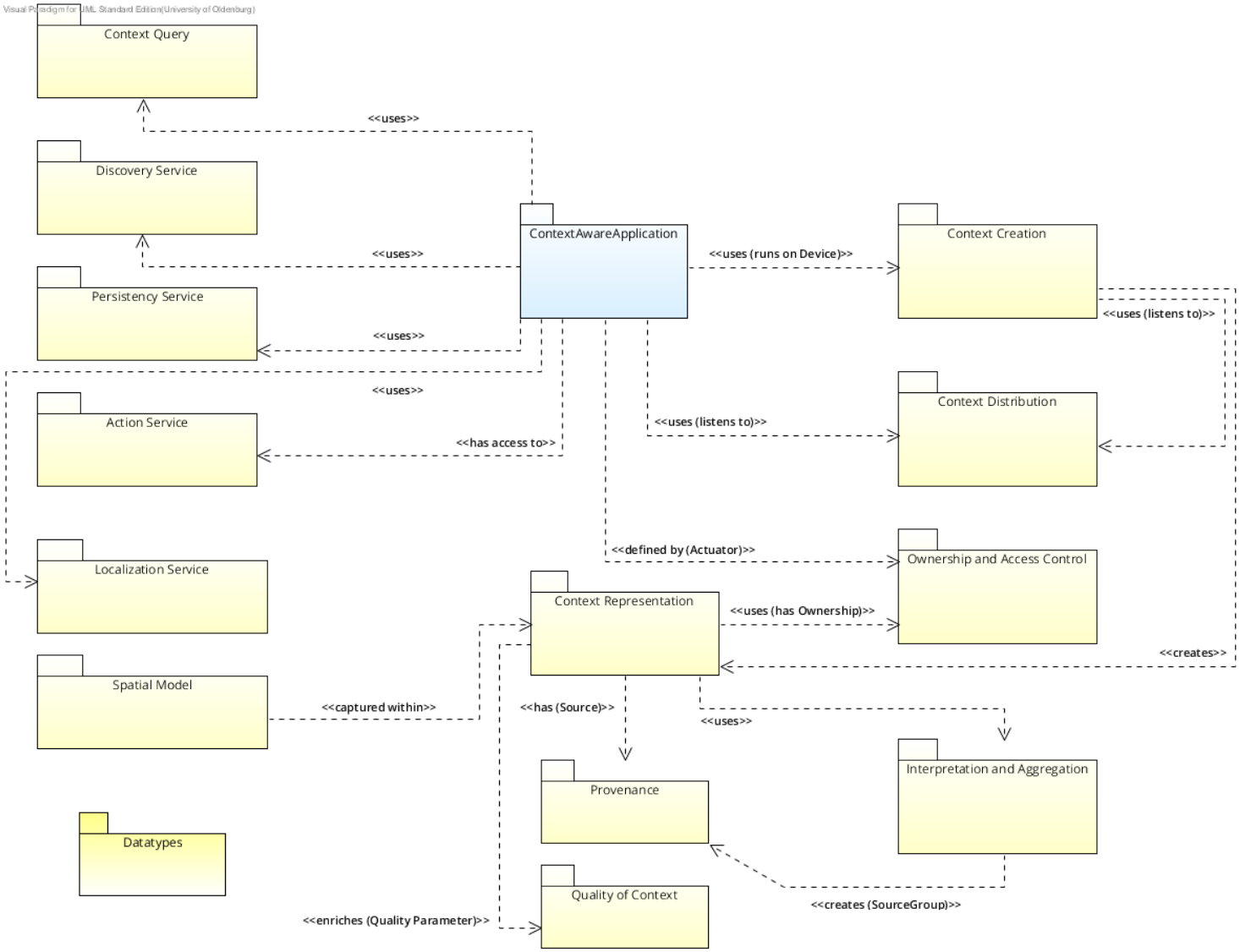


Abbildung 6.3: Das im Rahmen dieser Arbeit entwickelte Kontext-Modell, mit Bestandteilen und Beziehungen zwischen den Bestandteilen

6.2.1 Einbeziehung von kontextbewussten Anwendungen und spezifische Datentypen

Der zentrale Aspekt hinsichtlich die Benutzung des Kontext-Modells ist die Präsenz der Klasse “ContextAwareApplication” als “Hauptmerkmal” im Modell. Dadurch wird eine kontextbewusste Anwendung (engl. Context-Aware Application) repräsentiert. Diese Entität ist in einer direkten oder indirekten Weise mit allen anderen Komponenten des Modells verknüpft und wird normalerweise auf einem mobilen Gerät betrieben, welches in der Ziel-Umgebung eingebettet ist. Die Anwendung kann anhand technologischen Komponenten, die im Laufe des Paragraphs 6.2.3 eingegangen werden, mit anderen Anwendungen aus der Ziel-Umgebung verschiedene Kontext-Informationen austauschen. In dieser Sinne kann ein Netzwerk gebildet werden, die verteilt eine Anpassung an Kontext ausführen könnte.

Weitere Bestandteile des Kontext-Modells sind Datentypen, die Konzepte wie eine eindeutige Identifikation der einzelnen dargestellten Klassen sowie die Einkapselung Zeit-bezogener Sachverhalte enthalten. Die Klasse für die eindeutige Identifikation kapselt eine ID, einen optionalen Namen, sowie einen optionalen, eindeutigen URI ein. Was die zeitbezogenen Sachverhalte angeht, bestehen diese aus einem Zeitstempel (mit z.B. UTC, Coordinated Universal Time, als einzige Zeitzone) sowie einem Zeitintervall für die Beschreibung ablaufender Verläufe.

In den folgenden Abschnitten wird auf die Bestandteile des Modells einzeln eingegangen. Zudem wird ein entsprechender Bezug zu den Anforderungen an ein Kontext- Modell hergestellt.

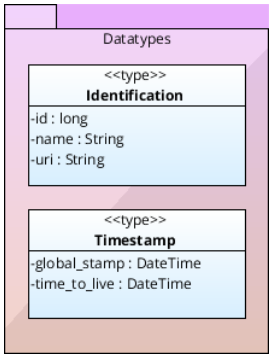


Abbildung 6.4: Datentypen des Kontext-Modells

6.2.2 Modell

In diesem Abschnitt werden die Bestandteile des Kontext-Modells eingeführt, welche Funktionen wie Datenerfassung und -beschreibung umsetzen können. Als Startpunkt wird die Repräsentation von Kontext-Information im Rahmen des Kontextes vorgestellt.

Erfassung, Interpretation und Aggregation von Kontext-Informationen

Ein Hauptthema des entwickelten Kontext-Modells ist die Erfassung von Kontext-Informationen (in der Fachsprache auch als “Fakten” bekannt) aus der umgebenden, auf dem Frontend vorhandenen Sensorik. Zu diesem Zweck ist die Idee der Erstellung von spezifischen *Widgets* aus [DA00] übernommen – ein Widget repräsentiert eine Kapselung der Funktionalität eines Sensors, welche durch eine Schnittstelle repräsentiert wird (ISensor). Die Schnittstelle ist für einen bestimmten Sensor eindeutig und kann beispielsweise Methoden rund um das Auslesen der Datenströme enthalten. Darüber hinaus lassen sich Widgets zu einer Hierarchie organisieren, sodass die Möglichkeit entsteht, z.B. eine bestimmte Kontext-Information anhand eines Durchlaufs der Hierarchie zu erfassen, jeweils mit Teil-Informationen von jedem Knoten (diese Prozedur wird als “Sensor Fusion” bezeichnet). Zudem wird an dieser Stelle auch das eigentliche Frontend oder die Client-Anwendung beschrieben. Dies geschieht durch die Klasse *Device*, auf der die beschriebenen Entitäten “Sensor” und “Widget” ein-

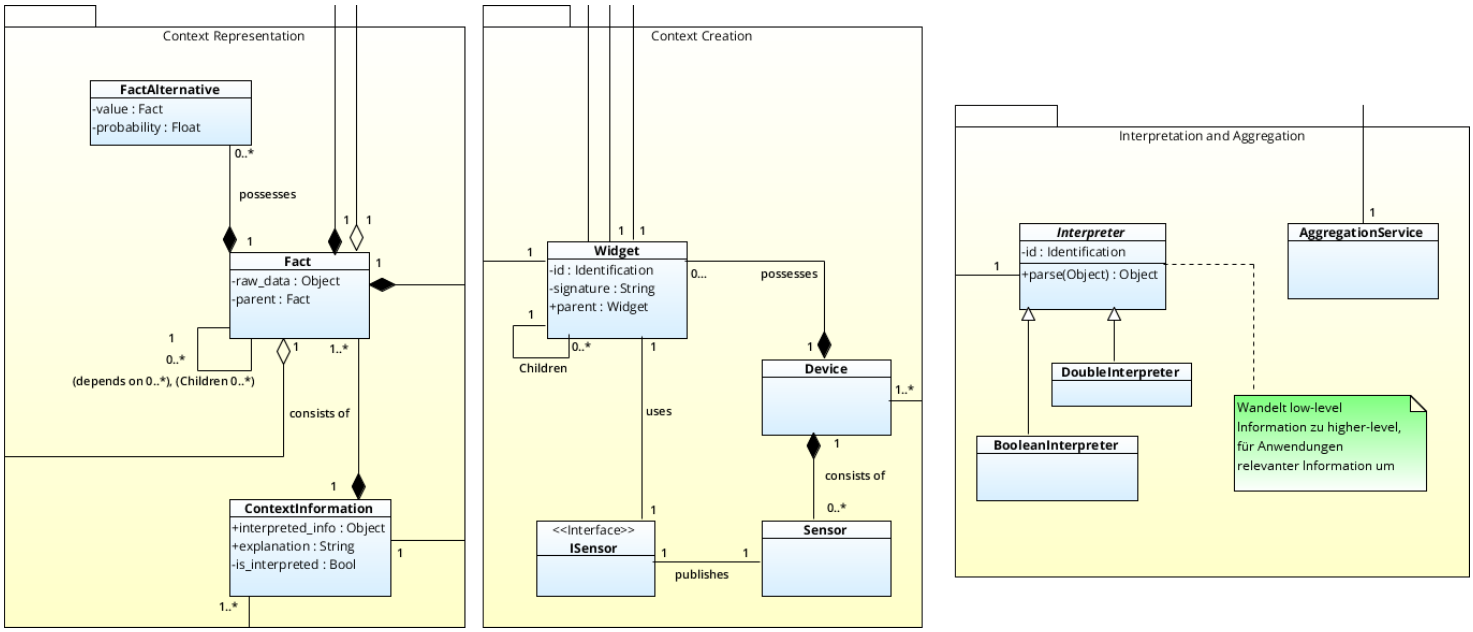


Abbildung 6.5: Teil des Kontext-Modells: Erfassung, Interpretation und Aggregation von Kontext-Informationen

gesetzt werden. Schließlich kann ein erfasstes Faktum auch “alternative” Fakten enthalten, die z.B. den selben Sachverhalt aus der Sicht eines anderen Sensors beschreiben.

Weiterhin erforderlich für die Erfassung von Sensoren-Daten ist eine einheitliche Repräsentation der (rohen) Daten. Innerhalb dieses Modells wird die Bezeichnung “Faktum” (engl. fact) benutzt und in der entsprechenden Klasse repräsentiert. An dieser Stelle besteht wiederum die Möglichkeit, die einzelnen Fakten zu einer Hierarchie zu gliedern, um z.B. eine zeitliche Reihenfolge oder einen Abhängigkeitspfad zwischen Knoten abbilden zu können. Eine Verknüpfung mit dem räumlichen Modell ist hier auch nennenswert: Fakten können zu einer bestimmten Lokation zugeordnet werden. Während ein Faktum die rohe, vom Sensor ausgelesene Information modelliert, enthält die Klasse *ContextInformation* die eigentliche “gereinigte”, abgeleitete Form der Daten. Zugleich lässt sich eine “Erklärung” der Ableitung in der genannten Klasse speichern, die z.B. die Auswahl der geeigneten Fakten und deren Interpretierung (s.u.) kurz “begründet” bzw. annotiert.

Wie oben aufgeführt lassen sich die erhaltenen oder ausgelesenen Fakten vom ihrer rohen Form in eine ausgelegte Form umwandeln – der Vorgang der *Interpretation*. Dafür sind Services vorhanden, wie die verallgemeinerte Klasse namens *Interpreter*. Mithilfe der konkreteren Ausprägungen bzw. der Unterklassen lassen sich rohe Daten zu üblichen Datentypen “zwingen” (engl. type coercion): beispielsweise kann ein konkretes DoubleInterpreter eine erfasste Zahl als eine Distanz zwischen zwei Objekte des Kontext-Modells umwandeln bzw. interpretieren. Diese Information hat dann für die kontextbewusste Anwendung eine klare Bedeutung, im Vergleich zum ursprünglichen, rohen Wert.

Ein zusätzlicher Dienst für den Umgang mit interpretierten Datenbeständen bzw. Kontext-Informationen ist der *Aggregationsdienst*. Hiermit lassen sich verschiedene Kontext-Informationen, die einen bestimmten für die untersuchte Domäne relevanten Sachverhalt darstellen, zusammenfassen bzw. gruppieren.

Auf diesen Aspekt der Aggregation wird im Absatz 6.2.2 näher eingegangen. Im selben Abschnitt wird auch ein bedeutsames Konzept des Kontext-Modells vorgestellt: die Möglichkeit, erfasste Kontext-Informationen mit Metadaten zu versehen.

Erfassung von Metadaten

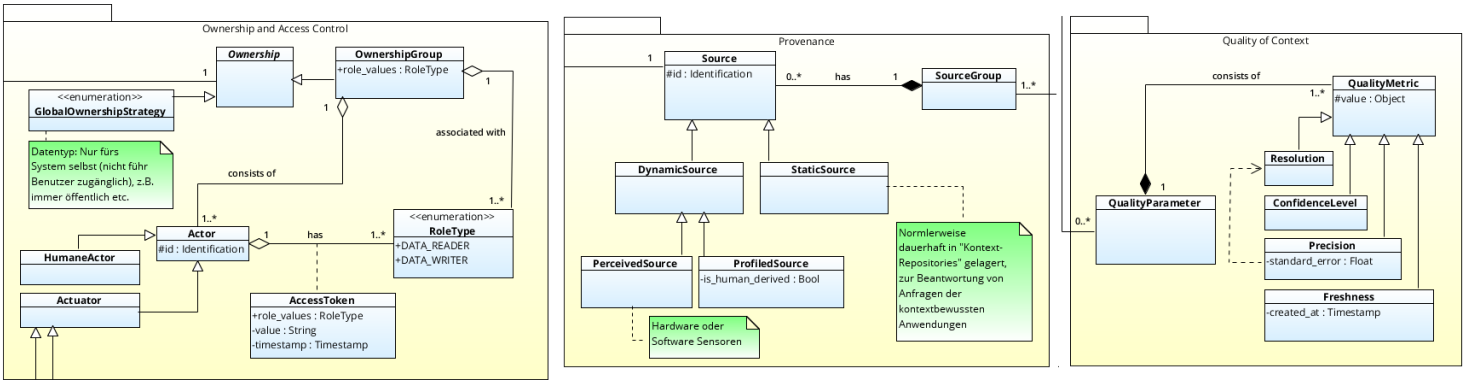


Abbildung 6.6: Teil des Kontext-Modells: Erfassung von Metadaten

In der Abbildung 6.6 sind die Bestandteile der integrierten Verwaltung von Metadaten innerhalb des Kontext-Modells abgebildet. Als erstes wird ermöglicht, Metadaten über den Besitzer der Kontext-Informationen zu erfassen – dazu wird eine abstrakte *Ownership* Klasse definiert, die entweder durch eine globale, für die Anwendungsdomäne spezifische Strategie (z.B. alle Daten sind nicht öffentlich verfügbar) oder durch eine Gruppe von Aktoren (s.u.) ausgeprägt werden kann.

Die tatsächliche Ausprägung des Besitzes wird durch die Definition von Aktoren ermöglicht, die entweder menschlich sind (*HumaneActor*) oder aus einer maschinellen Quelle herkommen (*Actuator*). Zusätzlich zu der Spezifikation von Besitz-Entitäten werden auch Mechanismen zur Zugangskontrolle im Kontext-Modell definiert.

Dafür sind zuerst verschiedene *Rollen* wie ein Lese- oder ein Lese/Schreibe-Rolle vorhanden, die den Zugang zu den Kontext-Informationen regeln. Die Rolle kann entweder direkt zu einem Aktor zugeordnet werden oder kann für eine bestimmte Gruppe von Aktoren angewendet werden. Weiterhin erlaubt das Modell die Definition von Zugriffsinformationen (engl. Access Token) für jeden Aktor, sodass der berechtigte Zugriff zu einem gewissen Zeitpunkt auch ablaufen kann. Der hierfür verwendete Mechanismus basiert auf einem Zeitstempel, z.B. ab dem Zeitpunkt in dem das Token als gültig im System markiert worden ist, läuft die Zeit bis zur definierten Gültigkeit schon ab. Der eigentliche Zugriff im System kann z.B. anhand eines festen Wertes wie ein kryptographisches Hashwert (eine private Schlüssel, die z.B. mit dem Server-System geteilt ist) erfolgen.

Weiterhin enthält das Kontext-Modell das Konzept der “Herkunft” (engl. provenance) der einzelnen Kontext-Informationen. Dabei ist die eindeutige Identifizierung (z.B. anhand der IP-Adresse) der Sensorik, die die Kontext-Informationen liefert, vorgesehen. Datenquellen können, wie genannt, identifiziert werden und lassen sich zudem in Gruppen von Datenquellen (engl. source groups) abbilden. Eine Datenquelle kann auf verschiedenen Art und Weisen bezeichnet werden und zwar: die Quelle ist entweder *dynamisch* (ein sich stets verändernder Wert) oder *statisch* (eine dauerhaft gespeicherter Wert, die z.B. bei der Konfiguration des Kontext-Modells für die Ziel-Anwendungsdomäne eingege-

ben bzw. gespeichert worden ist). Zudem lassen sich dynamische Quellen in *wahrgenommene* (engl. perceived) und *profilierte* Quellen unterteilen. Im Fall der wahrgenommenen Quellen handelt es sich um Hardware- oder Software-basierte Sensoren, die als Datenquellen agieren. Auf der anderen Seite sind profilierte Quellen Daten, die entweder von einem menschlichen Aktor eingegeben worden sind oder von der kontextbewussten Anwendung selber angelegt wurden.

Die letzte Kategorie von erfassbaren Metadaten ist die der Qualitätsdaten (engl. "Quality of Context"). Hiermit lassen sich verschiedene Metriken eingeben, die zu einer Kontext-Information zugeordnet werden. Die Metriken sind Teil einer Klasse namens *Qualitätsparameter*. Die im Kontext-Modell vorgesehenen Metriken beschreiben z.B. Konfidenzintervalle (beispielsweise Wahrscheinlichkeitswerte), die Genauigkeit (z.B. mithilfe des Standardfehlers) und die Resolution (z.B. eine Distanz oder eine räumliche Begrenzung) einer Sensor-Messung und letztendlich auch die so gesehene "Frische" einer Messung (die abgelaufene Zeit seit die Messung zustande gekommen ist) – hierzu können auch einzelne Messungen mit "Ablaufzeiten" vorgesehen sein.

Einbeziehung von Geodaten durch das Lokationsmodell

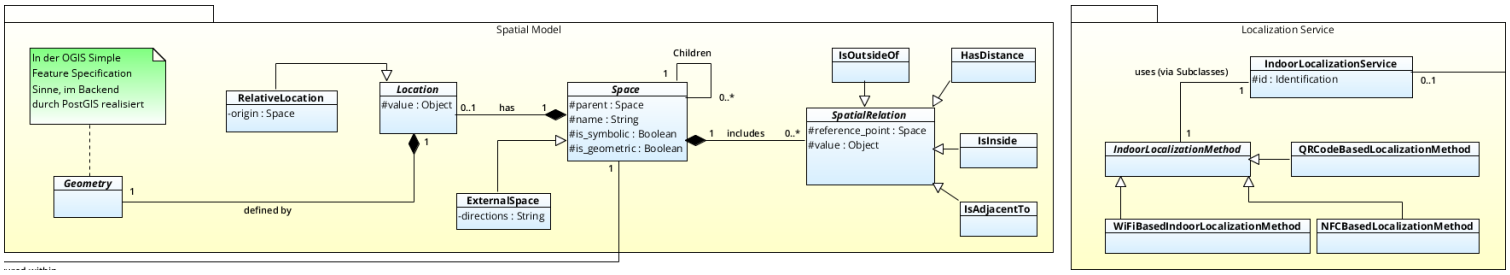


Abbildung 6.7: Teil des Kontext-Modells: Das räumliche Modell (engl. Spatial Model)

Als erster Bestandteil des Kontext-Modells wird das räumliche Modell betrachtet. Es handelt sich um ein hybrides Modell (vgl. [BD05]), welches symbolische sowie geometrische Daten repräsentieren kann. Weiterhin lassen sich verschiedene Typen von Beziehungen zwischen den einzelnen räumlich begrenzten "Objekten" (z.B. innerhalb der "Home Sharing" Domäne werden Räume und einzelne Gegenstände verortet) erfassen, wie z.B. Distanz-Beziehungen oder die Repräsentation einer Adjazenz-Beziehung. Die Möglichkeit, eine implizite Hierarchie zwischen den Objekten zu definieren, wurde im Modell auch dargestellt – ein Objekt kann in einer "Vater-Kind" (eine Baumstruktur) Beziehung zu einem anderen Objekt stehen.

Während die symbolischen Objekte durch einen Namen identifiziert werden können, wird im Fall von geometrischen Objekten eine präzise Lokation für ein Objekt vermittelt. Die Ausprägung dieser Lokation befolgt den internationalen Standard namens "Simple Features for SQL" Spezifikation [2] der Open Geospatial Consortium (OGC), womit mehrere Geometrien repräsentiert bzw. gespeichert werden können.

Verknüpft mit der Definition des räumlichen Modells ist die Bereitstellung von Methoden zur Indoor-Lokalisierung. Dieses Thema wird in diesem Modell durch einen Dienst realisiert, welcher eine der spezifizierten Methoden zur Lokalisierung benutzen kann. Wie in vorherigen Kapiteln auf-

[2] `http://www.opengeospatial.org/standards/sfa`

geführt, wurde im Rahmen der prototypischen Entwicklung nur eine Methode umgesetzt, deren Implementierungsdetails dem dedizierten Paragraph 7.4.3 zu entnehmen sind.

6.2.3 Service-Schicht

Im Folgenden werden die wesentlichen Dienste des Kontext-Modells im Detail vorgestellt.

Bereitstellung von Diensten zur Persistenz, Entdeckung und Feedback-Generierung bei gegebenen Kontext-Informationen

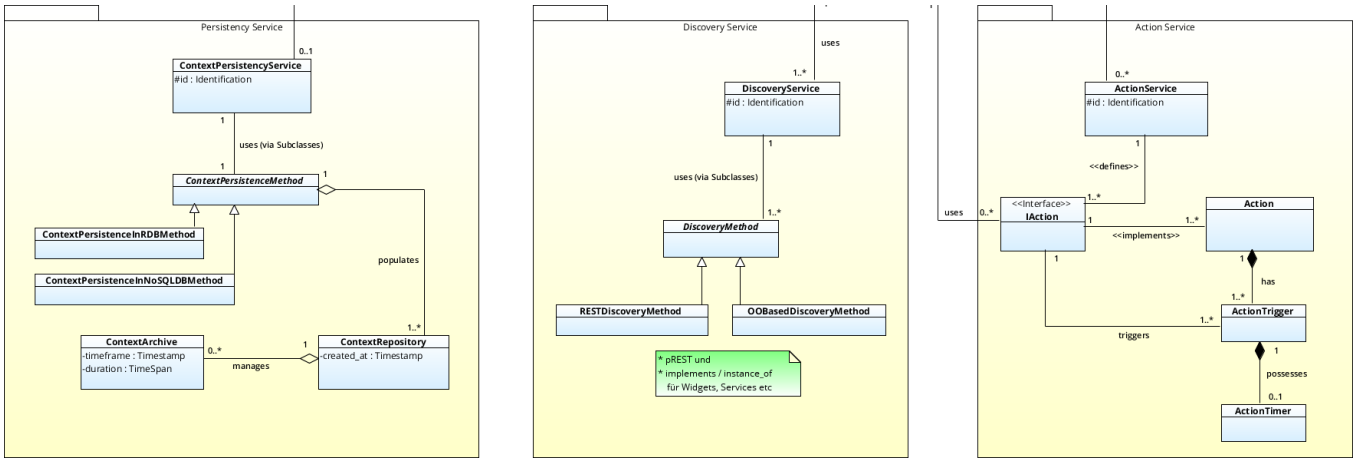


Abbildung 6.8: Teil des Kontext-Modells: Dienste zur Persistenz, Entdeckung (Discovery) und Feedback-Generierung

In der Abbildung 6.8 werden mehrere Dienste oder Services dargestellt, die ein Teil des Kontext-Modells sind. Diese Dienste lassen sich normalerweise von einem Server-System ausführen und stellen diverse Funktionalitäten bereit: zum einen wird die Möglichkeit gegeben, bestimmte Kontext-Informationen zu persistieren, in ein relationales oder nicht-relationales Datenbanksystem. Der damit entstehende Datenbestand (auch unter der Bezeichnung “Context Repository” bekannt) kann in verschiedene “Archive” verlagert werden, sodass verschiedene “Schnappschüsse” der gespeicherten Informationen erstellt werden können. Dieser Aspekt ist relevant, da die Kontext-Informationen normalerweise nur für ein begrenztes Zeitintervall gespeichert werden (sofern sie überhaupt gespeichert werden).

Weiterhin befasst sich der “Entdeckungs”-Dienst mit der möglichst automatischen Erweiterung der zunächst statisch gelagerten Kontext-Informationen bezüglich der Umgebung, in der das Kontext-Modell eingesetzt wird. Diese Erweiterung erzielt das Hinzufügen von neuen Widgets (s.u.) oder Services, die nur zur Laufzeit in die Umgebung eingefügt worden sind. In diesem Fall ist eventuell eine Aktualisierung der laufenden Software erforderlich, da die Software höchstwahrscheinlich mit den unerwarteten, neuen Diensten nicht umgehen kann. Für die Bereitstellung dieser Funktionalität wurde das Kontext-Modell mit zwei verschiedenen Methoden der “Entdeckung” versehen: ein auf pREST basierender Mechanismus (siehe auch 2.7.1) sowie ein objektorientierter Mechanismus, welcher die

dynamischen Fähigkeiten der Programmiersprache Ruby benutzt (für Implementierungsdetails siehe dazu auch 7.3.5).

Der letzte Dienst repräsentiert ein allgemeines Verfahren zur Definition von Anpassungen bzw. "Reaktionen" auf allgemein auftretende Ereignisse im Kontext-Modell. Hiermit wird über Ereignisse diskutiert, welche ausgelöst werden können und somit eine Aktion vom System erfordern. Durch dieses Konzept lassen sich Reaktionen oder allgemeines Feedback modellieren bzw. darstellen wie z.B. eine Mitteilung im Frontend, wenn ein bestimmter QR-Code eingescannt worden ist. Darüber hinaus könnte die Reaktion ausführlicher aufgestellt sein – es können automatische Aktionen durchgeführt werden, nicht nur auf dem auslösenden System, sonder auch auf anderen Systemen (damit soll ein Format der Anpassung bzw. Reaktion sowie ein Kommunikationskanal zwischen den wahrscheinlich technologisch heterogenen Systemen geschaffen werden).

Anfrage und Verteilung von Kontext-Informationen

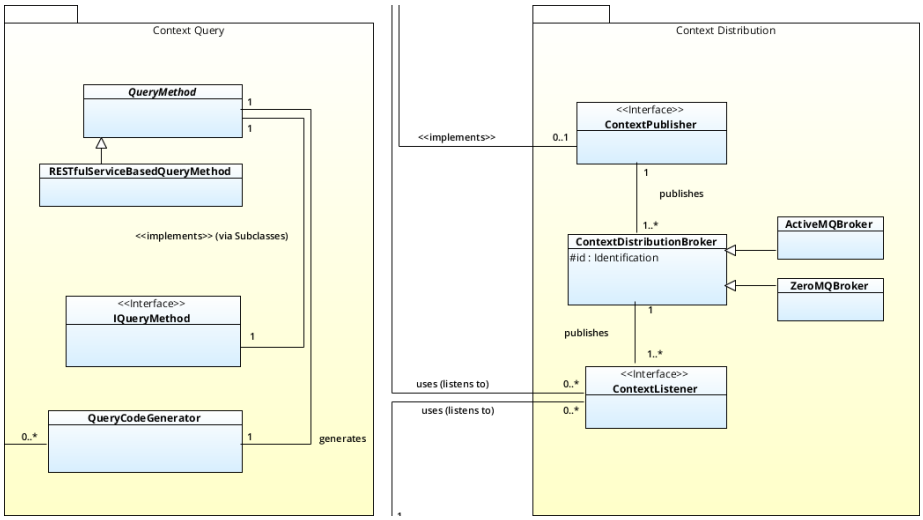


Abbildung 6.9: Teil des Kontext-Modells: Anfrage und Verteilung von Kontext-Informationen

Nachdem die Kontext-Informationen aus der Umgebung oder mithilfe der lokalen, auf dem mobilen Gerät vorhandenen Sensorik erfasst worden sind, wird im Kontext-Modell die Möglichkeit gegeben, diese Informationen anzufragen. Dieser möglicher Anwendungsfall ist insbesondere im Szenario der "Smart Home"-Umgebung relevant. In diesem Szenario wird normalerweise von der Existenz mehrerer kontextbewusster Anwendungen ausgegangen. Diese Anwendungen können auch "zusammenarbeiten", indem sie Informationen aus mehreren Quellen durch Anfragen "verlangen". Um dieser Aspekt zu ermöglichen, lassen sich verschiedene Anfragemethoden definieren, die eine Schnittstelle bereitstellen und mithilfe von Quelltext-Generierungsmethoden erstellt werden können. Eine weitere Beschreibung einer Generierungsmethode, die auf REST Web Services basiert, folgt im Abschnitt 7.3.5.

Eine grundlegende Anfrage könnte die Lokation eines `InternalAsset` verlangen, durch z.B. eine HTTP Anfrage wie `GET /location/Chair/relative/Sofa`. Die Semantik dieser Anfrage ist die folgende: die Lokation (geographisch oder symbolisch) des Assets mit dem Namen "Chair" wurde verlangt, relativ zu einem anderen Gegenstand namens "Sofa".

Bei der Distribution oder Verteilung von erfassten Kontext-Informationen lassen sich dieselben Technologien einsetzen (bzw. das Netzwerkprotokoll namens AMQP, welches insbesondere für Echtzeit-Anwendungen geeignet ist), die auch für das Konzept des Kommunikationskanals im Paragraph 5.2.3 aufgeführt worden sind. Im Paragraph 7.3.4 werden zudem die Implementierungsdetails diskutiert. Nennenswert bleibt an dieser Stelle auch die Verallgemeinerung des Verteilungsvorganges: es können weitere Technologien eingesetzt werden, die entweder einen direkten Pub-Sub Kommunikationsmechanismus umsetzen oder dies mithilfe eines Zwischendienstes (engl. Middleware) realisieren. Die "Abonnenten" können im Sinne des Kontext-Modells andere kontextbewusste Anwendungen aus derselben Umgebung wie die "produzierenden" Anwendungen sein.

Publisher-Subscriber, oft als PubSub bezeichnet, ist ein allgemeiner Mechanismus, welcher zum einen die Erstellung von beliebigen "Nachrichten" oder Datenbeständen von einer Anwendung erlaubt und zum anderen die entsprechende Verteilung der veröffentlichten Datenbestände an beliebige "Abonnenten" bzw. Anwendungen ermöglicht.

Nachdem die Bausteine des Kontext-Modells definiert worden sind, wird ein kurzer Überblick über eine mögliche Nutzung des Kontext-Modells anhand eines vereinfachten Anwendungsfalles gegeben.

6.3 Beispielhafte Instanziierung des Kontext-Modells

In der Abbildung 6.10 wird das entworfene Kontext-Modell in einem beispielhaften Szenario einer "Smart Home" Umgebung eingesetzt. Hierfür wurde eine kontextbewusste Anwendung (`SH1`, Platzhalter für "Smart Home 1") definiert, die von einem mobilen Gerät (`HTC_Desire`) betrieben wird.

Die Umgebung wird durch das Lokationsmodell bestimmt und enthält mehrere Räume wie `Room_1` und `Room_2`, die Teil einer Wohnung (`Apartment_1`) sind. Die Wohnung befindet sich in einem Gebäude (`Building_1`), dessen auf GPS-basierende Koordinaten bekannt sind. Darüber hinaus werden verschiedene räumliche Beziehungen definiert wie `Distance_to_Room_1`, die die geschätzte Distanz zwischen den zwei Räumen erfasst.

Fest verankert im System sind Merkmale des Besitzes: das System selber verwendet ein globales Schema für den Besitz, `SH_Sys1_GlobalOwnership`, welcher mit einem Zeitstempel und einer Signatur versehen ist. Dieses `AccessToken` wird dem anderen Akteur (dem Bewohner) im System mitgeteilt, sodass ein Zugriff auf die vom System bereitgestellten Gegenstände (s.u) möglich ist.

Die vorher modellierten Gegenstände im System umfassen mehrere Fakten: es gibt ein virtuell modelliertes Sofa (`IntelligentSofa`), welches sich im Raum2 befindet. Dieses Sofa wird als "Smart Object" betrachtet und stellt eine über REST-verfügbare Schnittstelle zu Funktionen wie `getDescription()` bereit. Die statische, vorher gegebene Lokalisierung des Sofas ist ein Beispiel für die Klasse `ContextInformation`.

Weitere Kontext-Informationen im System werden dynamisch erfasst: auf dem mobilen Gerät ist ein Sensor zur Vermittlung der aktuellen umgebenden Helligkeit eingebaut. Dieser Sensor wird benutzt, indem er neue Informationen in Form von Fakten zur Verfügung stellt. Ein Faktum wie `BrightnessFact1` wird von einem "Interpreter" wie `BrightnessLuxInterpreter` interpretiert und ergibt dadurch eine profilierte Informationsquelle.

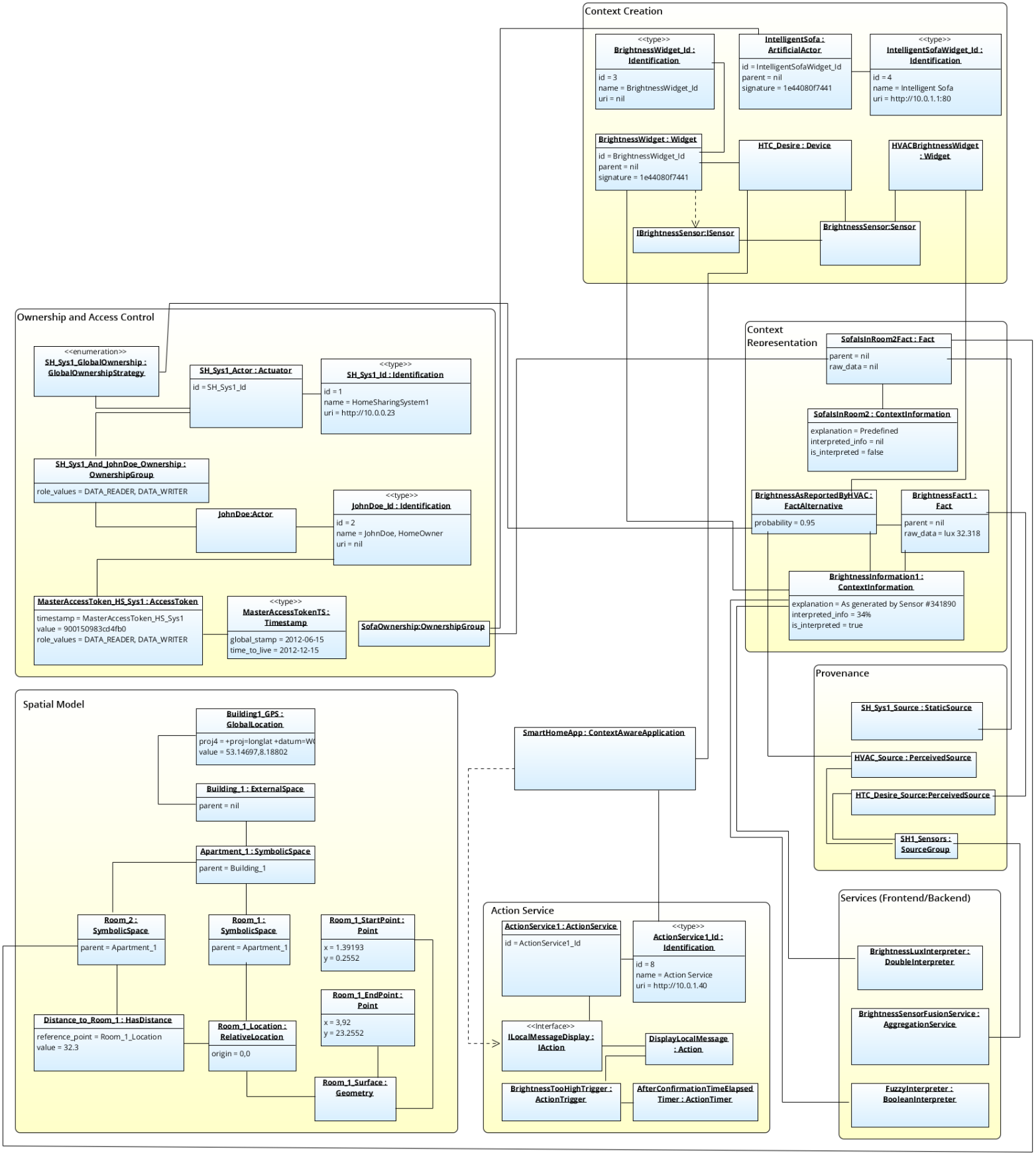


Abbildung 6.10: Veranschaulichung beispielhafter Instanzen des Kontext-Modells

Weiterhin befindet sich im Haus ein System zur Erfassung der Lichtintensität in jedem Raum. Dieser Sensor wird auch benutzt um alternative Fakten zu erstellen. Diese beiden Informationen werden mit einer gewissen Wahrscheinlichkeit zu einer einheitlichen Darstellung in Form einer Kontext-Information fusioniert.

Eine Veränderung des umgebenden Lichtes wird dem mobilen Gerät mitgeteilt, indem eine Benachrichtigung mittels des Dienstes `ActionService1` erstellt und eventuell angezeigt wird.

Schließlich werden Metadaten wie z.B. die Herkunft der einzelnen Kontext-Informationen durch Klassen wie `LightPerceptor_Source` und `HTC_Desire_Source` erfasst.

6.4 Zusammenfassung

In diesem Kapitel wurde einer der zentralen Beiträge dieser Arbeit vorgestellt – das Kontext-Modell. Mit diesem Konzept können verschiedene konkrete Szenarien (s.u.) umgesetzt werden, die ein "Netz" von kontextbewussten Anwendungen und von in der Umgebung eingebetteten Sensorik beschreiben können. Dieses Netz kann zusammen mit der benötigten Darstellungen von Kontext-Informationen zu der Generierung von kontextbezogenen Anpassungen führen kann.

Bestandteile und Konzepte dieses Modells wurden prototypisch eingesetzt, zur Umsetzung eines "(Smart) Home Sharing" Szenarios. Die dafür entwickelte Anwendung wird im nächsten Kapitel aus Sicht der Implementierung vorgestellt.

7 Umsetzung des Prototyps

In diesem Kapitel wird vorgestellt, wie die Umsetzung der beschriebenen Konzepte in Form eines Prototyps realisiert ist. Die entstehende Softwarelösung bzw. der Prototyp setzt sich aus den zwei genannten Teilen zusammen, dem Backend sowie dem Frontend, auf welche in diesem Kapitel näher eingegangen wird. Als Erstes wird aber ein kurzer Überblick über das Konzept "Quelltextqualität" gegeben.

7.1 Allgemeine Merkmale der Quelltextqualität

Unter Berücksichtigung der Merkmale der Quelltextqualität wurde bei der Entwicklung der Softwarelösung darauf geachtet, dass die folgenden allgemeinen Prinzipien der Qualität durchgehend eingesetzt wurden:

- **Dokumentation**: Die Dokumentation wurde mithilfe von Javadoc[1] im Fall des Frontends sowie ein Javadoc-ähnliches System namens YARD[2] im Fall des Backends erstellt. Dokumentiert wurden Klassen, einzelne Methoden sowie weitere erklärungsbedürftige Attribute bzw. Felder der Klassen. Die Verwendung von Javadoc und Javadoc-ähnlichen Dokumentationswerkzeugen verschafft den Vorteil, dass die Vorgänge zur Erstellung von statischen Webseiten, die die Softwarelösung anhand der eingebetteten Dokumentationstexte vorstellt, automatisiert sind. Darüber hinaus wird versucht, Kodeblöcke mit eigenen Erklärungen in Form von ein- oder mehrzeiligen Kommentaren zu versehen. Dies dient zur Erhebung der Lesbarkeit und allgemeinen Verständlichkeit der benutzten Programmiertechniken.

- **Logging**: Zusätzlich zu der Dokumentation wurde darauf geachtet, sämtliche Logging-Ereignisse so häufig wie möglich zu verwenden. Durch die Bezeichnung "Logging" wird die Dokumentation von Ereignissen rund um das entwickelnde System zusammengefasst. Diese Mitteilungen lassen sich verschiedenen Warnungsstufen zuordnen und können in mehreren Formaten exportiert werden. Ein häufiger Anwendungsfall von Logging ist die Ausgabe eintretender, meist unerwarteter System- sowie Werkzeug-Fehlermeldungen und -Ereignisse. Diese Ausgaben müssen im Hintergrund behandelt bzw. gelagert werden, sodass der Nutzer nicht beeinträchtigt oder verwirrt wird. Die verwendeten Logging-Werkzeuge sind im Fall des Frontends mit der Android-Plattform mitgeliefert, während im Fall des Backends wurde das Werkzeug namens "Log4r" (eine Umsetzung der Log4j Konzepten aus der Java-Welt in Ruby) umgesetzt und bzw. erweitert wurde.

- **Behaviour-Driven Development (BDD[3])**: Im Fall des Backends wurden Prinzipien rund um die verhaltensgetriebene Softwareentwicklung eingesetzt. Bei der Verwendung der verhaltensgetriebenen Softwareentwicklung wird geprüft, ob die Ziele (z.B. die Anforderungen) des Softwareprodukts erreicht werden können. Zu diesem Zweck wird eine *Spezifikation* (engl. specification, Kurzform "spec") erstellt, in der das erwartete Verhalten des Quelltextes kurz beschrieben wird. In ihrer rein textuellen Form beschreibt eine Spezifikation das Verhalten mithilfe von "**Vorausgesetzt** dass [...] erfüllt ist, beim **Auftritt dieses Ereignisses** muss dies **folgendermaßen behandelt werden**" (engl. "Given – When – Then") Sätzen. Hiermit können auch die Stakeholders mit bei

[1] Ein in Java eingesetzt aber an weiteren Sprachen angelehntes Format für die textuelle Dokumentation vom Quelltext
[2] YARD (Yay! A Ruby Documentation Tool), `http://www.yardoc.org/`
[3] Einführung in BDD: `http://dannorth.net/introducing-bdd/`

der Erstellung solcher Spezifikationen miteinbezogen werden. Aus Sicht des Softwareentwicklers werden im Fall von BDD einzelne *Beispiele* des erwarteten Verhaltens des Quelltextes erstellt, die auch als Akzeptanz-Tests oder Integrationstests (engl. acceptance und integration test) bekannt sind. Ein Unterschied zwischen TDD (Test-Driven Development, testgetriebene Softwareentwicklung) zu BDD ist, dass bei TDD normalerweise die Struktur, Organisation und Robustheit der Softwarelösung getestet werden, während eine TDD-Spezifikation die Softwarelösung als eine Black-Box betrachtet und nur die Erfüllung einer gewissen Funktionalität (oder eines Verhaltens) erwartet.

- **Verwendung von open-source Bibliotheken und Projekte**: Alle bei der Umsetzung des Softwareproduktes eingesetzten Technologien bzw. Produkte sind im Internet frei verfügbar und für präzisere Zwecke anpassbar. Dieses Kriterium stellte sich als Hauptmerkmal für die Auswahl der Werkzeuge dar, die im Laufe der Entwicklung verwendet worden sind. Die Möglichkeit einer Anpassung war essentiell im Fall von einigen Werkzeugen, die beispielsweise hinsichtlich ihrer Fehlerausgabe oder allgemeinen Funktionsweise anhand des frei verfügbaren Quelltextes untersucht worden sind. Hiermit ist auch die Gelegenheit verbunden, einige kleinere, für diese Arbeit benutzten Software-Teile als open-source "Produkte" veröffentlichen zu können.

Nachdem in einer zusammengefassten Weise die einzelnen bei dem Entwicklungsvorgang beachteten Merkmale zur Sicherung einer erhöhten Quelltextqualität beschrieben worden sind, wird der Fokus auf die tatsächliche Umsetzung der Bestandteile der Softwarelösung gelegt. Zu diesem Zweck wird zuerst die für die Umsetzung des "Home Sharing" Szenarios entwickelte Ausprägung des Kontext-Modells (hier als "Datenmodell" gekennzeichnet) vorgestellt.

7.2 Datenmodell für Home Sharing

Während in Kapitel 6 die Definition eines allgemein anwendbaren Kontext-Modells gegeben wurde, muss eine weitere Definition einer konkreten Ausprägung bzw. Umsetzung dieses Modells eingebracht werden. In dieser Arbeit wurde zugleich ein Datenmodell erstellt, welches die in Kapitel 3 beschriebenen User-Stories unterstützt. Eine visuelle Darstellung dieses Modells erfolgt anhand der Abbildung 7.1.

Neben den genannten, in Kürze zu erläuternden Bestandteilen des Datenmodells befinden sich in dem dargestellten Modell außer den impliziten (s.u.) Verknüpfungspunkten mit dem verallgemeinerten Kontext-Modell auch tatsächliche Verknüpfungen: die Klassen `Location` und `Source`. Die Einbeziehung der `Location` Klasse liegt im Bedarf begründet, Kontext-Informationen zu erzeugen und zu bearbeiten, die auch in der Wohnung verortet sind. Weiterhin wurde die Klasse `Source` in diesem Modell verknüpft, um den Unterschied zwischen vordefinierten, statischen, vom System anfänglich bereitgestellten Daten (siehe auch 7.3.1) und dynamischen, "auftretenden" Datensätze ausdrücken zu können.

Die genannte Art von "impliziten" Verknüpfungspunkten mit dem grundlegenden Kontext-Modell bezieht sich darauf, dass versucht wurde, die Komplexität der entwickelten Softwarelösung zu verringern. Unter diesem Gesichtspunkt handelt es sich um implizites Wissen, das im Datenmodell enthalten ist. Die räumlichen Daten werden nicht durch einzelne Fakten wie im Paragraph 6.3 im Fall der beispielhaften Instanziierung dargestellt, sondern sie werden durch eine Assoziation zwischen den teilnehmenden Klassen (bzw. Relationen aus Sicht des Datenbankmodells) ausgeprägt. Außer-

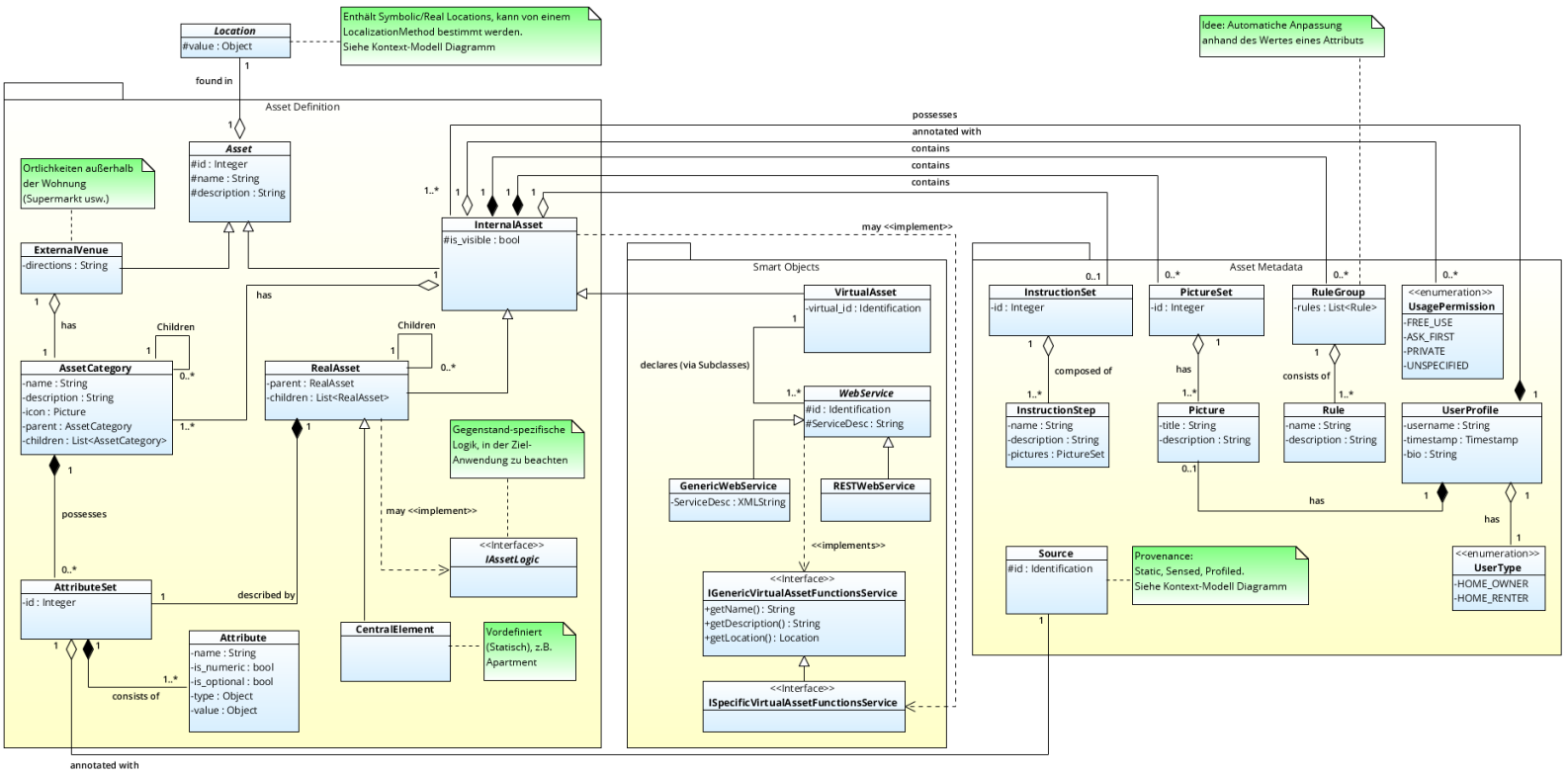


Abbildung 7.1: Überblick des Datenmodells, mit folgenden Bestandteilen: Asset Definition (a), Virtual Assets (b) sowie Asset Metadata (c)

dem lässt sich das Verfahren hinsichtlich der Absicherung des Kontext-Modells vom unberechtigten Zugang (Die Rolle der `AccessToken` Klasse wurde im Laufe des Abschnittes 6.2.2 behandelt) mithilfe eines Private-Key Verfahrens absichern, wie im Abschnitt 7.3.5 beschrieben wird. Darüber hinaus übernimmt das Datenbanksystem die Verwaltung und Zuordnung von Rollen(Das Konzept der Zugangsrollen wurde im Absatz 6.2.2 eingeführt), wie im Paragraph 7.3.1 vorgestellt wird.

7.2.1 Erfassung von Gegenständen (a)

Die erste unterstützte Funktionalität des Datenmodells für Home Sharing ist die Erfassung von Gegenständen aus der Wohnung. Hierfür wird die Basis-Klasse `Asset` zur Verfügung gestellt, welche sich in Gegenstände innerhalb der Wohnung (`InternalAsset`) oder Örtlichkeiten außerhalb der Wohnung (`ExternalVenue`) spezialisiert. Die Verbindung mit dem räumlichen Modell wird durch eine Assoziation dargestellt – ein `Asset` ist einer bestimmten geometrischen oder symbolischen Lokation zugeordnet. Des Weiteren können im Fall von `ExternalVenue` Klassen auch Wegbeschreibungen annotiert werden. Zudem ermöglicht ein `InternalAsset` die Ausblendung bzw. nicht-Veröffentlichung eines bestimmten Gegenstandes und die Erfassung verschiedener Attribute (siehe unten).

Weiter kann ein `InternalAsset` in ein `RealAsset` oder in ein `VirtualAsset` (siehe nächsten Paragraph) unterteilt werden. Die Modellierung als `RealAsset` erlaubt die Darstellung verschiedener Gegenstände in einer hierarchischen Struktur und die Zuweisung einer bestimmten Kategorie oder mehrerer Kategorien (z.B. in Form verschiedener *Tags*) mittels der Klasse `AssetCategory`. Darüber hinaus wurde die Schnittstelle `IAssetLogic` modelliert, sodass ein Hinweis darauf gegeben wird, dass ein Gegenstand auch mit einer bestimmten Logik in derjenigen Anwendung, die das Datenmodell umsetzt, "eingeplant" werden kann. Weiterhin besteht im Rahmen des "Home Sharing" Szenarios der Bedarf, ein zentrales Element (`CentralElement`) statisch modellieren zu können – in diesem Fall wird das zentrale Element durch eine Wohnung repräsentiert.

Die Instanzen der Klasse `InternalAsset` sowie die Instanzen der Klasse `AssetCategory` unterstützen die Definition von beliebigen Eigenschaften anhand der Klasse `AttributeSet` und implizit der Klasse `Attribute`. Durch diesen Mechanismus können Eigenschaften wie z.B. die Wohnfläche für jede Art von Raum oder die Mobilität eines bestimmten Gegenstands ausgedrückt werden. Eine Eigenschaft (ein `Attribut`) kann einen Namen und eine Beschreibung tragen und kann zugleich hinsichtlich der Optionalität und des Typs (numerisch oder textuell) angepasst werden.

Im Fall von `InternalAssets` können verschiedene Metadaten erfasst werden, welche im übernächsten Paragraph erläutert werden.

7.2.2 Einbeziehung von Smart Objects (b)

Die im Paragraph 2.2.1 eingeführten Konzepte des "Smart Objects" Forschungsfeldes werden in das Datenmodell für Home Sharing einbezogen, in welchem die weitere Spezialisierung der vorher erläuterten `InternalAsset` Klasse, namens `VirtualAsset` modelliert ist. Hiermit wird es ermöglicht, zumindest auf einer konzeptuellen Ebene (s.u) verschiedene zusätzliche, domänenspezifische virtuelle Funktionalitäten an einem Gegenstand zu vergeben.

Die Bezeichnung "eine Art von Web Services" zeigt die Erweiterbarkeit des Prinzips: es können entweder generische Web Services (`GenericWebService`), bei denen eine WSDL Beschreibung vorliegt, oder REST Web Services (siehe auch Abschnitt 2.7), bei denen eine spezifische Art von Beschreibung (siehe dazu Abschnitt 7.3.5) vorliegt, modelliert werden.

Bezüglich der erwarteten Funktionalität eines `VirtualAssets` werden zur Verringerung der Komplexität keine Aussagen in diesem Modell getroffen. Vorgesehen ist die Bereitstellung einer grundlegenden Funktionalität, die einige für das "eingekapselte", anreicherte Objekt relevante Daten zurückliefert. Beispielsweise liefern die Anfragen `getName`, `getDescription` und `getLocation` grundlegende Informationen über den letztendlich modellierten `Asset`. Darüber hinaus besteht die Möglichkeit, eine Schnittstelle zu definieren bzw. vom eingekapselten `InternalAsset` umsetzen zu lassen, die für den Gegenstand eine spezifische Funktionalität (über einen Web Service, wie beschrieben) bereitstellt. An dieser Stelle können z.B. fortgeschrittene Programmlogik ablaufen, wie beispielsweise eine Anfrage an verschiedener Parameter direkt vom eingekapselten Gegenstand.

7.2.3 Erfassung von Metadaten (c)

Eine weitere in dem Datenmodell enthaltene Funktionalität ist die Beschreibung verschiedener Daten über die einzeln modellierten Gegenstände der Umgebung. Hiermit können folgende Metadaten für jeden Gegenstand erfasst werden:

- **Bedienungsanleitungen, Hinweise zum Gebrauch** (`InstructionSet`, `UsagePermission`) : Mithilfe der Klasse `InstructionSet` können mit dem Gegenstand assoziierte Bedienungsanleitungen erfasst werden. Eine Bedienungsanleitung (`Instruction`) wird durch einen Namen, eine Beschreibung und ein dazugehöriges, optionales Bild ausgeprägt. Weiterhin können Hinweise zum Gebrauch erfasst werden, indem theoretische Einschränkungen des Gebrauchs gemacht werden. Beispielsweise beschreibt "Frage zuerst" (ASK_FIRST) die Situation, in der der Mieter dezent darauf hingewiesen wird, dass vor dem Gebrauch des spezifischen Gegenstandes nachgefragt werden sollte, ob dies möglich wäre.

- **Bilder** (`PictureSet`) : Durch diese "Metadaten" können Bilder zu den einzelnen Gegenständen assoziiert werden. Diese Informationen können zu einer erleichterten Assoziierung der digital vorgestellten Informationen mit dem Gegenstand aus der realen Welt bzw. aus der Umgebung dienen. Bilder werden in Fotogalerien gruppiert und lokal (heißt auf dem Server-System) gespeichert.

- **Regeln** (`RuleGroup`) : Hiermit werden Regeln erstellt, die bei oder vor der Benutzung des entsprechenden Gegenstandes zu beachten sind. Diese Informationen werden zur Zeit nur in einer textuellen Form erfasst und entsprechend dargestellt, jedoch könnte eine weitere Entwicklung auch den Fall vorsehen, dass statische, vorhandene Regeln eine automatische Wirkung bei der Benutzung haben. Beispielsweise könnte dadurch ermöglicht werden, dass jeder Gegenstand, welcher mit der Regel "Bitte nicht benutzen" versehen ist, während oder nach der Mietphase wieder auf die Einhaltung dieser Regel überprüft wird. Hilfreich an dieser Stelle ist eine Liste mit allen Gegenständen, die geprüft werden sollten.

Innerhalb des Diagramms wurden die Klassen `UserProfile` und `UserType` auch als Teile des "Asset Metadata" Softwarepakets einbezogen. Diese "Maßnahme" wurde eingeführt, sodass das Konzept des Besitzes (im Rahmen des Kontext-Modells, Paragraph 6.2.2) im Datenmodell auch verfügbar ist. Somit modelliert die `UserProfile` ein minimales Profil über den Wohnungseigentümer und dessen Gäste. Ebenfalls enthalten ist der Zeitstempel (Attribut `timestamp`), womit die Zugangskontrolle geregelt werden kann (siehe auch 6.2.2). Weiterhin erfasst die `UserType` Klasse die zwei Klassen von Benutzern aus dem Home Sharing Szenario: den Wohnungsmieter (HOME_GUEST) sowie den Wohnungsbesitzer (HOME_OWNER).

Zusammenfassend enthält das entwickelte Datenmodell für Home Sharing die für die Umsetzung der in Kapitel 3 vorgestellten User-Stories benötigten Bestandteile. Das Datenmodell wird von der UML-Notation zu einer relationalen Datenbankschema umgewandelt, wie in den folgenden Abschnitten, die die zentrale Komponente bzw. das Backend behandeln, vorgestellt wird.

7.3 Backend

Das Backend setzt die Persistenzschicht des Kontext-Modells, verschiedene Dienste rund um das Kontext-Modell sowie die technische Verbindung zum Kommunikationskanal um. Diese Unterkomponenten werden ausführlich in den kommenden Abschnitten eingeführt und beschrieben.

7.3.1 Persistzenzkonzept des Kontext-Modells

Das entworfene Kontext-Modell muss in einer relationalen Datenbank persistiert werden, zur Gewährleistung der Integrität und Sicherheit der gespeicherten Daten, die meistens persönliche Daten darstellen. Zu diesem Zweck wurde das verbreitete Datenbankmanagementsystem namens PostgreSQL in der Version 5.1 eingesetzt. Zudem wurde für die normenkonforme Speicherung und Bearbeitung von geographischen Daten das Werkzeug mit dem Namen "PostGIS" angewendet – ein Plugin für PostgreSQL, welches die Verwaltung von geographischen Daten ermöglicht.

Von UML Diagramm zur Datenbankschema

Wie vorgestellt wurde das Kontext-Modell zunächst in Form von UML Diagrammen entworfen. Um diese Darstellung in Form eines Datenmodells entwerten zu können, müssen diejenige darin beschriebene Klassen und deren Beziehungen zuerst in Datenbank-spezifische Konstrukte umgewandelt werden. Dieser Vorgang entspricht in der MDA-Terminologie etwa "Model Transformation". Im Raum dieser Softwarelösung wurde ein Werkzeug für die Automatisierung dieses Vorgangs entwickelt.

Als Eingang nimmt das Software eine Excel-Datei (Microsoft Excel ist ein Softwareprodukt für den Umgang mit Tabellenkalkulationen und ist Teil der Microsoft-Office-Suite) entgegen, die die entsprechende UML-Diagramme des Kontext-Modells darstellen. Die UML-Diagramme sind mittels der UML-Modelleierung Software namens Visual Paradigm entworfen und exportiert worden. Auf der Ausgangsseite lassen sich Ruby-Klassen generieren, die ihre in der Diagramm modellierten Assoziationen und Attributen beibehalten. Das Ausgangsformat entspricht dem vom "DataMapper" ORM (s.u.) benutzten Format.

Der nächste Schritt zur Erreichung der Repräsentation des Datenmodells in einer relationalen Datenbank ist von einer Initialisierung der Datenbank (bzw. die Erstellung der entsprechenden Relationen) repräsentiert. Zuerst werden einige einführende Details über die Verbindung mit dem Datenbanksystem gegeben.

Kommunikation mit der Datenbank und Initialisierung dessen

Die Kommunikation mit der Datenbank erfolgt mittels selbstgeschriebenen Wrappers einer Datenbankverbindung. Benutzt wurde das mit dem "DataMapper"[4] ORM mitgelieferte Verbindungsmodell, sowie ein "schlankes" Wrapper über das Verbindungsmodell des "pg" Gems. Eine Verbindung lässt sich entweder als der Datenbankadministrator (der so genannte "root" oder "superuser" Benutzer) oder als der Administrator einer bestimmten Datenbank, oder als ein Nutzer mit eingeschränkten Rechten dessen erstellen.

[4] `http://www.datamapper.org`

Die genannten Verbindungsmodelle bauen auf einer TCP-Verbindung auf und können bei Bedarf mittels TLS/SSL gesichert werden. Außerdem besteht die Möglichkeit bei beiden Modellen, SQL-Befehle manuell einzugeben, eine Möglichkeit die demnächst bei der Initialisierung verwendet werden. Schließlich bleibt eine Verbindung mit der Datenbank bis der Schließung dessen offen, in einer "Keep-Alive" Weise.

Nachdem eine Verbindung erstellt ist, lassen sich entweder ORM-spezifische Befehle oder wie genannt "rohe" SQL-Befehle zu dem Datenbanksystem übergeben. Im Fall der Initialisierung werden SQL-Befehle der Typ "DDL" verwendet, um die Datenbank für die Benutzung (von der Wohnungsmieter sowie -eigentümer Anwendunge) vorzubereiten. Data Definition Language (DDL) ist eine Sprache mit einer Programmiersprache-ähnlichen Syntax, womit Datenstrukturen insb. Datenbankschematas definiert werden können. Die Steuerung des Initialisierung Vorgangs erfolgt per Kommandozeile – notwendig dafür ist nur der interne Name der Datenbank, wie z.B. `HomeSharingDB1`; wenn aber keinen Namen eingegeben wird, wird automatisch ein generischer Name angewendet.

Die Initialisierung wird dadurch ausgeprägt, dass mehrere DDL Befehle über die Verbindung ausgeführt werden, womit die ggf. bestehende Datenbank gelöscht und neu erstellt wird. Weiterhin erfolgt eine erneute Erstellung der genannten, pro Datenbank verfügbaren Rollen – der Datenbankadministrator und die eingeschränkten Datenbankbenutzer (ein Datenleser-Benutzer und ein Daten-Lese/Schreiben-Benutzer). Wie der Name es schon eindeutet, behält der Datenleser nur Lese-Rechte, während der Lese/Schreiben-Benutzer dazu Schreiben-Rechte auf dem entsprechenden Datenbank behält. Letztendlich kann der Datenbankadministrator die erstellte Datenbank verwalten (das heißt, Benutzer, Relationen und andere Elemente einfügen, bearbeiten oder löschen).

Bevor die Beschreibung fortgesetzt wird, müssen einige Details hinsichtlich der Umsetzung der im Abschnitt 5.2.2 aufgestellten Sicherheitkonzepte gegeben werden, weil dieser Vorgang bei dem Anlegen von neuen Datenbank-Benutzern verwendet wird.

Sicherheitskonzept – Umgang mit Passwörter

Das grundlegende Konzept für den Umgang des Backends mit den benötigten Passwörter basiert auf folgende Voraussetzungen:

1. Die (Basis)-Passwörter bzw. Passwort-Dateien sind nur auf dem Server-System zugänglich
2. Die Passwort-Dateien sind nur dem Backend-Systemadministrators (z.B. der Wohnungseigentümer) zugänglich
3. Das Backend-System ist mit einem Administrator-Passwort versehen

Während die erste Voraussetzung mit der Organisation des Quelltextes im Fall der Backend-Anwendung erfüllt ist (durch die Handhabung des `config/secrets` Ordners), müssen die anderen aufgelistete Elementen entweder vom Wohnungseigentümer (wenn das Server-System auf einem eigenen Server installiert wird) oder vom Dienstleister umgesetzt werden . Die Mechanismen der Zugangskontrolle am Betriebssystem-Ebene sind dafür zuständig – man kann z.B. Rechte nur dem Systemadministrator vergeben.

Die auf dem Server gelagerten Passwörter bzw. die entsprechenden Dateien werden vom Backend ausgelesen, angepasst und benutzt. Die genannte Anpassung wird in Form eines einfachen Ersetzen

eines Platzhalter-Textes ausgeführt – z.B. in dem "Satz" `$Pass-is_HERE/` wird den Platzhalter "HERE" durch den entsprechenden Passwort ersetzt.

Der nächste Schritt in der Bereitstellung der Datenbank-Umgebung ist das Laden und die letzte Vorbereitung der Modelle, die in der Datenbank als Relationen gespeichert werden.

Ladevorgang und Vorbereitung des Datenmodells

In diesem Schritt werden die von dem Komanndozeile-basierten Programm gelieferten und ggf. noch manuell verarbeiteten Ruby-Klassen geladen. Mittels Ruby werden erstmal die einzelnen Dateien rekursiv aus dem Dateisystem geladen (mithilfe des `require` Befehls). Somit sind alle in der Abbildung 7.1 dargestellten Klassen dem Backend-System bekannt und die letzten Vorbereitungen können anfangen.

Nachdem alle Modelle (in der DataMapper Terminologie auch als "Ressourcen" bekannt) geladen sind, müssen die "abgeschlossen" bzw. überprüft werden. Der Befehl `DataMapper.finalize!` ist dafür zuständig. Dieser Zwischenschritt ist benötigt, um diejenige deklarierten Modelle nach Fehler wie z.B. eine nicht unterstützte Kardinalität (beispielsweise "0..3" anstatt "0..n") zu untersuchen sowie die mit den Assoziationen verbundenen Getters und Setters zu initialisieren.

An dieser Stelle ist der Umgang mit dem Datenmodell für die Formulierung von Befehlen wie z.B. die Erstellung von Instanzen bereit. Die Ausführung der Befehle setzt voraus, dass die Migration (d.h. die entsprechende Darstellung der Modelle als Relationen in der Datenbank) schon abgeschlossen bzw. durchgeführt ist. Dieser Schritt wird im kommenden Paragraph erläutert.

Initielle Migration und Befüllung der Datenbank

Der "Seeding" Vorgang muss zuerst über eine Datenbankschema verfügen – zur Zeit wurde nur die Datenbank und deren Rollenverwaltung durchgeführt worden. Für die Erstellung einer Schema (meist im engl. als ein "migration" bekannt) müssen weitere DDL Befehle übergeben werden, wodurch die einzelnen Relationen erstellt werden. Dieser Schritt erfolgt fast automatisch dank dem genannten "DataMapper" ORM, mittels der `DataMapper.automigrate!` Befehl.

Die Auswirkung dieses Befehls ist es, dass diejenige Klassen, die mit Eigenschaften (Attributen) und Assoziationen (Beziehungen) versehen und als DataMapper "Ressourcen" markiert sind, sich in der Datenbank wiederfinden werden. Das ORM kapselt dann dieser Prozess (die Ausführung von DDL Befehle wie z.B. `CREATE TABLE`) ein, abstrahierend von der händischen Übergabe von DDL-Befehlen und die eventuell damit verbundene Fehlerbehandlung.

Das Konzept einer "Migration" berührt am meisten die einzelnen Schritte, die erfolgen müssen, um einen gewissen Zustand der Datenbank bzw. des Datenbasis zu erreichen. Beispielsweise lässt sich die Erstellung einer einzigen Relation auch als ein "Migrationsschritt" ansehen. Vorteilhaft bei diesem Konzept ist die Möglichkeit, eine ausgeführte Migration zurücksetzen zu können – damit wird den umgekehrten Weg einer Migration beschrieben. In der entwickelten Softwarelösung wird die ganze Erstellung der zum Datenmodell gehörenden Relationen als eine einzige Migration gesehen, die durch eine erneute Initialisierung behoben werden kann.

Nachdem die Relationen erstellt sind, lassen sich diejenige ursprünglichen Datenbestände dadrin speichern – wie herausgestellt, ist dieser Prozess durch den Bezeichner "seeding" beschrieben. In diesem Fall werden die ursprünglichen Elementen des Datenmodells in der Datenbank mittels DML

Befehle wie z.B. `INSERT INTO ... VALUES ...` befüllt. Data Manipulation Language (DML) ist eine Sprache mit einer Programmiersprachen-ähnlichen Syntax, die für die Erstellung, Entfernung und Aktualisierung von Datenbestände in einer Datenbank verwendet wird.

Es werden nämlich die `AssetCategory` und andere Relationen wie CentralElement (siehe auch Abschnitt 7.3.2) mit einem ursprünglichen Datenbasis befüllt. Durch diese ursprüngliche Befüllung wird die weitere Modellierung der Umgebung so gesehen "freigegeben" – der Wohnungseigentümer kann die vorhandene Menge an Kontext-Elementen wie z.B. die AssetCategories sehen und eventuell erweitern.

Dadurch sind einige der technischen Vorbedingungen behandelt, die für die Benutzung bzw. Befüllung des ursprünglichen Datenmodells erforderlich sind. Die Beschreibung der eigentlichen Benutzung des Kontext-Modells mithilfe der mobilen Anwendungen wird im Absatz 7.4 in Detail vorgeführt.

Umsetzung des "Data Liberation" Vorgangs

Ein zentraler Prinzip des "Data Liberation" Vorgangs ist die Bereitstellung der exportierten Daten in ein möglichst offenes, portables Format, um die Interoperabilität des Ergebnisses gewährleisten zu können. In diesem Fall wird es von einem allgemeinen Datenexport der Datenbank gesprochen – das übliche Format ist eine SQL-Datei, die aus mehreren DDL-Befehlen wie `CREATE TABLE` sowie `INSERT INTO` besteht. Dadurch bleibt nur die Erfüllung der Interoperabilität – im Fall des PostgreSQL Datenbanksystems müssen einige Aspekte mitberücksichtigt werden, wie z.B. die Vermeidung des eigenen "COPY" (schnellere, PostgreSQL-spezifische Variante vom `INSERT INTO`) Befehls.

Die Analyse vorhandener Lösungen zu einem Ruby-unterstützten Datenbank-Export haben gezeigt, dass entweder die Werkzeuge nur für UNIX-ähnliche Betriebssysteme anwendbar sind (wie im Fall des "Backup"[5] Gems) oder dass sie nur mit einer spezifischen ORM (in diesem Fall, ActiveRecord, bekannt als Teil des "Ruby on Rails" Web Framework[6]) auszuführen sind (der Fall bei dem "pg_dumper"[7] Gem).

Zu diesem Zweck wurde ein eigenes Kommandozeile-basiertes Ruby Programm entwickelt, welches das mit PostgreSQL mitgelieferte Werkzeug `pg_dump` aufruft und den Datenexport ausführt. Die Funktionsweise ist die folgende: mit der Übergabe des Namens der Datenbank (siehe auch 7.3.1) werden die Zugangsparameter bestimmt (Benutzername, Name der zu exportierenden Datenbank, welche die Datenmodell enthält). Den Pfad bis zum `pg_dump` wird ermittelt (ggf. wird nach einer manuellen Angabe nachgefragt, wenn den Pfad nicht automatisch ermittelt werden kann) und der Datenexport wird ausgeführt.

Als Ergebnis wird eine SQL-Datei zurückgegeben, welche der Zustand der Datenbank zu dem Zeitpunkt der Ausführung widerspiegelt. Optional kann diese Datei mit dem GZip Verfahren komprimiert werden. Die Datei kann von anderen Datenbanksystemen benutzt werden oder zu einer PostgreSQL Datenbank automatisch mithilfe des `pg_restore` Werkzeuges wieder importiert werden.

[5] `https://github.com/meskyanichi/backup`
[6] `http://www.rubyonrails.org`
[7] `https://github.com/mikz/pg_dumper`

7.3.2 Statische, vordefinierte Elemente des Datenmodells

In diesem Abschnitt wird kurz die Prozedur zur ursprünglichen Befüllung des Datenbestandes erläutert. Für diesen Vorgang wurden Ruby-Programme entwickelt, die die Daten auslesen, interpretieren, in einem Datenbank- bzw. ORM-geeigneten Format umwandeln und letztendlich die interpretierten Daten als feste Instanzen des Datenmodells in der relationalen Datenbank speichern.

AssetCategory

Die erste Kategorie von statischen, vordefinierten Elementen des Datenmodells ist von der Klasse `AssetCategory` (siehe auch 7.2) repräsentiert. Diese Datensammlung wurde händisch von dedizierten Quellen wie ein Online-Werkzeug für die Erstellung von Grundrissen[8] oder ein Online-Werkzeug für die Erstellung von verschiedenen Diagrammen-Typen , welche auch die Erstellung von Grundrissen ermöglicht[9] zusammengefasst.

Die Darstellung der gesammelten Daten wurde in einer formatierten Text-Datei realisiert, zur Erleichterung der Bearbeitung. Die Vater-Kind Beziehungen zwischen den einzelnen Kategorien bzw. "Knoten" wird durch die Einrückung bestimmt. Trennzeichen unterscheiden die verfügbaren Attribute wie Name, Beschreibung oder für die Darstellung benutztes Symbol. Ein Parser wurde für diese vereinfachte Datei-Struktur geschrieben. Nachdem die Daten ausgelesen sind und damit die Struktur bzw. die Vater-Kind Beziehungen festgestellt sind, werden die Daten zu DataMapper-entsprechenden Instanzen der Klasse `AssetCategory` umgewandelt und in der Datenbank gespeichert.

Derselbe Vorgang wird im Fall der "externen" Kategorien (d.h. die Örtlichkeiten *außerhalb* der Wohnung, `ExternalVenue`) eingesetzt, mit dem Unterschied dass kein Parser erforderlich ist, es wurde ein JSON-Parser für Ruby verwendet. Die Quelle der Örtlichkeiten ist die API der populäre Dienst für Check-Ins und lokale Vorschläge namens Foursquare[10], welche die Taxonomie der erfassbaren Örtlichkeiten veröffentlicht. Eine Application Programming Interface (API) ist eine Spezifikation der Schnittstellen, die für die Kommunikation zwischen Software-Komponenten verwendet werden können.

Um eine transparente Darstellung der auswählbaren Kategorien im Frontend zu ermöglichen, wird bei der Umwandlung der ausgelesenen Daten zu DataMapper-Ressourcen eine Vereinbarung gemacht. Zwar wird als Vater-Knoten für die hiermit entstehende Baum-Struktur die vorher erfassten Kategorie, die "Gegenstände" außerhalb der Wohnung beschreibt, benutzt. Weitere Informationen sind im Abschnitt 7.4.1 beschrieben.

CentralElement

Der zentrale "Gegenstand" wird durch eine Wohnung ausgeprägt und lässt sich durch ein Kommandozeile-gesteuertes Programm automatisch erstellen. Hierfür sind Parameter der `UserProfile` erforderlich (siehe auch 7.2.3), wie z.B. einen Benutzername oder einen kurzen Profil-Satz.

Die damit entstehende Instanz der `CentralElement` Klasse wird in der Datenbank gespeichert und kann danach benutzt werden. An dieser Stelle findet eine "Anpassung" des vorher befüllten Datenbestandes: die generisch benannten Elementen wie z.B. *Apartment/House* oder *Building* werden

[8] `http://www.floorplanner.com`
[9] `http://www.gliffy.com`
[10] `http://www.foursquare.com`

aktualisiert bzw. umbenannt, anhand der Eingaben zum eigenen Profil. Somit werden die Namen und dazugehörigen Beschreibungen bestimmter Kategorien mithilfe des eingegebenen Benutzernamens angepasst. Beispielsweise werden im Fall des Benutzernamens *Larry23* folgende Anpassungen durchgeführt: “Larry23’s Apartment/House” und “Larry23’s Building”.

In diesem Fall wird versucht, eine grundlegende aber wahrnehmbare Personalisierung der End-Anwendung (nämlich das Frontend) auszuführen. Die Zielsetzung wäre, die benötigten Anpassung des Mieters am neuen Kontext (die Wohnung) einigermaßen zu erleichtern. Mit diesem Vorgang der Befüllung der `CentralElement` Relation anhand der Kommandozeile-gesteuerten Angaben zum Profil sind weitere “Seed”- bzw. Befüllungsvorgänge verknüpft – dieser Aspekt wird im Paragraph 7.3.3 behandelt.

AttributeSet

Im Fall der `AttributeSet` Klasse wurden ein paar beispielhaften Attribute vorhandener Kategorien vergeben. Der Vorgang startet mit dem Auslesen der dafür zuständige Datei, welche in einem YAML-Dateiformat bereitgestellt wird. YAML (Yet Another Markup Language) ist eine vereinfachte Auszeichnungssprache (engl. markup language), die vielen üblichen Datenstrukturen serialisieren kann. Ein mit Ruby mitgelieferten Softwarepaket ist bei dem Auslesen der YAML-Datei benutzt. Die Umwandlung erfolgt entsprechen danach, wo z.B. nach einer bestimmten Vereinbarung gesucht wird – ein optionaler Attribut ist im Text mit einem Asterisk, *, markiert.

Nachdem wird versucht, die vorhandene Wert als einen bekannten Datentyp wie z.B. Integer oder Float zu interpretieren. Wenn diese Umwandlung nicht möglich ist, wird die angegebene Wert für eine allgemeine Zeichenkette (String) gehalten. Die Umwandlung ist fertig sobald die ausgelesenen Schlüssel und dazugehörigen Werten wie z.B. “anzahl Stühle: 2” für die Kategorie “Tisch” in einen Datenmodell-konformen Form gebracht sind. Als letztes erfolgt die Speicherung in der Datenbank.

Nachdem den ursprünglichen Datenbestand in der Datenbank gespeichert ist, müssen sich gewisse Konfigurationsvorgänge ausgeführt werden, eine Thematik die in den kommenden Abschnitte behandelt wird.

7.3.3 Konfiguration des Server- und Anwendungszugangs

Die aufwendige Aufstellung- und Installationsvorgänge sind aus der Welt der “Smart Home” Technologien bekannt, wo sich der Aufwand der Konfiguration als eine erhebliche Herausforderungen zur breiteren, kommerziellen Einführung der Technologie herausstellt [RQBA09].

Dafür ist eine der Zielsetzungen der entwickelten Softwarelösung, eine möglichst schnelle und einwandfreie Konfiguration des Servers bzw. des Backend-Systems anzubieten. Zu diesem Zweck wurden Kommandozeile-basierten Werkzeuge entwickelt, die dieser Vorgang der ersten Konfiguration automatisieren. Diese Werkzeuge werden als nächstes vorgestellt.

Das erste Werkzeug ist für die Aktualisierung der IP-Adresse des Servers zuständig. Der Ziel dieses Werkzeuges ist, die erste registrierte lokale IP des aufrufenden Rechners zu übernehmen und diese in den Konfigurationsdateien des Frontend- und Backend-Systems zu aktualisieren. Damit wird sichergestellt, dass das Backend auf der vom Frontend angesprochenen IP-Adresse lauscht und dass das Frontend eine Verbindung mit der richtigen IP-Adresse versuchen wird. Eine mögliche Erwei-

terung dieses Programms kann auch angesehen werden – zur Zeit funktioniert diese kleine Anwendung nur im Fall der *privaten* Klasse von IP-Adressen, nämlich: `10.0.0.0/8`, `172.16.0.0/12`, oder `192.168.0.0/16`, welche normalerweise für den privaten, "häuslichen" Verbrauch eingesetzt sind. Im Hinblick darauf muss die Funktionalität dieses Programms manuell reproduziert werden, wenn die Anwendung z.B. für Testzwecke in einem "beruflichen" Netzwerk getestet werden sollte.

Zusätzlich wurde ein anderes Kommandozeile-gesteuertes Programm entwickelt, welches für die im Abschnitt 7.3.1 genannte ursprüngliche Migration und Befüllung der Datenbank zuständig ist. Die Aufgabe dieses Programms ist nur die Sammlung bzw. das Laden der involvierten Klassen und die Ausführung der Funktionalität dieser Klassen.

Zudem unterstützt ein weiteres Kommandozeile-gesteuertes Programm die Verteilung des Zugangswertes (nämlich die Private Schlüssel) für die REST Services, welche im Abschnitt 5.2.2 behandelt wurde. Hiermit wird diesen Wert in dem Quelltext der mobilen Anwendungen aktualisiert. Darüber hinaus werden Werte aktualisiert, die für die künftige Distribution (oder engl. deployment) der Anwendung relevant sind (siehe auch 7.4). Die Ausführung dieses Programms kann als eine Vorbedingung für die erfolgreiche Distribution der Anwendungen angesehen werden. Damit wird sichergestellt, dass alle erforderlichen Einstellungen durchgeführt worden sind. Als nächstes werden weitere Aspekte der Distribution der Anwendungen vorgestellt.

Distribution der Anwendung für Wohnungsmieter

Der nächste Schritt bis hin zur Nutzung der entwickelten mobilen Anwendungen ist die Distribution der Anwendungen als Android-konforme `.apk` Dateien. Die normale Distribution einer Android Anwendung erfolgt auf der Google "Market" Plattform (oder inzwischen Google "Play Store") Plattform, wo Anwendungen und die dazugehörige `.apk` Datei heruntergeladen und lokal, auf dem End-Gerät installiert werden können. Zusätzlich steht eine Konfigurationsoption bereit, die diese "Limitierung" aufhebt und dadurch die Installation von Anwendungen auch aus "unvertrauten" Quellen zulässt.

Der Unterschied zwischen eine "unvertraute" und "vertraute" Anwendung wird durch die Herkunft der .apk-Datei etabliert: um eine Anwendung auf dem Market veröffentlichen zu können muss die .apk "Distributionsatei" signiert werden. Diese Authentifizierung ist nötig, um sicherzustellen dass die Anwendungen, die von einem Entwickler oder einer Softwareentwicklungsfirma veröffentlicht sind, tatsächlich aus dieser Quelle auch herkommen. In dieser Hinsicht wird normalerweise dasselbe Zertifikat (oder Signatur) für alle vom demselben Entwickler veröffentlichte Apps verwendet [RLB10].

Das Distributionsmodell dieser Anwendung setzt voraus, dass die entstehende, zu installierende .apk-Datei *immer* als eine unvertraute Quelle vom Android angesehen wird, da die Datei nicht aus dem Market heruntergeladen wurde. Diese Limitierung wird auch auf der offiziellen Seiten[11] über die Distribution einer Anwendung erläutert – eine weitere Limitierung wird an der Stelle auch eingeführt: manche Telefonanbieter sperren die genannte Konfigurationsoption. Aufgrund dieser möglichen Begrenzungen in dem Distributionsmodell könnte die Distributionsphase beeinträchtigt werden – eine Evaluation mit mehrerer Nutzer könnte diese Hypothese (ob die Installation der Anwendung durch Telefonanbieter behindert werden könnte) überprüfen. Auf der genannten Seite, die Hinweise zu der Distribution von Android-Anwendungen gibt, werden auch Möglichkeiten eingegangen, die die .apk-

[11] `http://developer.android.com/distribute/open.html`

Datei per E-Mail oder eine Webseite verteilen. Dies stellt sich als eine praktische Variante und könnte auch im Fall vom alternativen Distributionsmodell (s.u.) angewendet werden.

Des Weiteren kann ein anderes Distributionsmodell berücksichtigt werden: es können die benötigten Zugangsdaten (siehe auch 7.3.3) mittels einem genannten Kommunikationskanal wie E-Mail oder eine gesicherte Webseite verschickt werden. Dieser Schritt impliziert dennoch eine manuelle Eingabe vom Wohnungsmieter – ein Punkt, der mit dem erzielten Prinzip "Zero-Configuration" (Handhabung möglichst automatisierter Konfigurationsvorgänge, im Vergleich zu dem Konfigurationsaufwand, der mit "Smart Home" Lösungen normalerweise verknüpft ist – siehe auch 3.2.2) kollidiert. Hinsichtlich dieses Aspekts wird in dieser Arbeit das erste Distributionsmodell eingesetzt.

Ein Vorteil der von dieser Softwarelösung verwendeten Herangehensweise ist die Verbesserung des genannten Sicherheitsgefühls, welches vom Wohnungseigentümer und -mieter gespürt werden sollte und in der Grundprinzipien des "Home Sharings" oder der allgemeinen "Sharing Economy" verwurzelt ist. Die Distribution ist dann wird dadurch in einem "geschlossenen Kreis" ausgeführt und die Sicherheit der personenbezogenen Daten wird eindeutig. Dieser Punkt ist im Rahmen einer Mitarbeit mit einem weiteren Dienstleister noch zu beachten, sodass die erzielte Sicherheit bestehen bleibt.

Die tatsächliche Laufumgebung bzw. worauf die Anwendungen laufen, kann unterschiedlich sein: im Normalfall bringt der Wohnungsmieter sein eigenes mobiles Gerät mit und lässt die Installation der Softwarelösung darauf laufen oder in einem weniger wahrscheinlichen Fall wird dem Mieter ein mobiles Gerät vom Wohnungseigentümer zur Verfügung gestellt. Wie bereits genannt bleibt die Distribution nur eine Frage eines ursprünglichen Vertrauens. Natürlich lässt sich die Einstellung, auch "fremde" Anwendungen laufen zu lassen, nach der Installation bzw. Distribution der entwickelten Softwarelösung zurücksetzen.

Trotzdem könnte auch eine hybride Lösung zur Distribution verwendet werden: der Wohnungseigentümer kann die .apk-Datei selber erstellen bzw. anhand eines automatisierten Programms erstellen lassen, welches die benötigten Schritte (Kompilierung, Signierung) ausführt und dann per E-Mail zu dem Mieter sendet und darauf hinweist, dass die Konfigurationsoption für unvertraute Quellen eingestellt werden soll. Damit wird der End-Benutzer in dem Prozess aktiv miteinbezogen (siehe auch das Konzept der "Nutzerbefähigung" aus Paragraph 4.3.5) und es wird Zeit beim Start der Mietphase gespart. Weiterhin könnten die Zugangsdaten auch per QR-Code eingelesen werden.

Nachdem die Aspekte der Konfiguration und Distribution der mobilen Anwendung eingeführt sind, wird das Thema des Kommunikationskanals im kommenden Abschnitt vorgestellt.

7.3.4 Kommunikationskanal

Das Konzept des Kommunikationskanals hat als technische Grundlage das AMQP-Protokoll, welches ein Kommunikationsprotokoll darstellt, wie im Absatz 5.2.3 erläutert. Das AMQP-Protokoll stellt nur einen Grundriss bereit, worauf weitere Anwendungen in Form von AMQP-fähigen Serverprodukten namens Brokers laufen können. Die meist verwendete Serverlösung für AMQP wird vom frei verfügbaren RabbitMQ Broker repräsentiert.

Die Entwicklung von RabbitMQ begann im Jahr 2006, als der erste Entwurf der AMQP-Spezifikation veröffentlicht wurde. RabbitMQ verwendet die Programmiersprache Erlang als technologische Grund-

lage und ist als eine open-source Alternative zu kommerziellen MQ Implementierungen wie IBM WebSphere MQ oder Amazon Simple Queue Service gegeben.

Erlang ist eine relativ kompakte Programmiersprache, die zugleich ein Laufzeitsystem und eine umfassende Bibliothek für die Entwicklung verteilter, hochverfügbarer Systeme zur Verfügung stellt. Eine alternative Bezeichnug für Erlang ist *Erlang/OTP*, bei der das Akronym OTP für The Open Telecom Platform steht.

Die Auswahl von RabbitMQ kann folgenderweise begründet werden: aus verfügbaren open-source Message-Oriented Middleware Softwarelösungen wie ActiveMQ, ZeroMQ oder Apache Qpid setzt nur Apache Qpid den AMQP-Standard um [VW12]. Darüber hinaus verfügt RabbitMQ über eine umfangreiche Dokumentation inklusive Fachbücher.

Ein weiteres Argument für RabbitMQ sind die praktischen, mitgelieferten Erweiterungen des AMQP-Protokolls. Beispielsweise können unzustellbare Mitteilungen persistiert werden; weiter lässt sich der Empfang einer Mitteilung von einem Verbraucher zu dem aufrufenden Erzeuger bestätigen. Eine ausführliche Beschreibung des Funktionsumfangs von RabbitMQ sowie weitere Erklärungen zum AMQP-Protokoll sind in [VW12] vorhanden.

Queuing-Strategie für die Umsetzung des Kommunikationskanals

Die generelle Strategie, die verwendet wurde, um die Funktionalität des Kommunikationskanals bereitzustellen ist in der folgenden Auflistung wiedergegeben:

(a) **Dauerhafte Queues**: Es wird jeweils eine dauerhafte Queue für jede teilnehmende Rolle (Wohnungseigentümer und -Mieter) definiert. Die Queues können die im Sinne dieser Herangehensweise auch als “Inboxes” oder als “Foren” für jeden Teilnehmer angesehen werden. Hierbei werden dauerhafte Mitteilungen persistiert, indem aktuelle sowie vorherige Mitteilungen abgerufen werden können. Hier können auch Mitteilungen gespeichert werden, die zu dem Sendezeitpunkt nicht empfangen werden konnten (die “Offline” Art von Mitteilungen). Eine dauerhafte Queue kann nur auf Anfrage gelöscht werden.

(b) **Kurzlebige Queues**: Darüber hinaus lassen sich nicht dauerhafte Queues erstellen, die für die Lebensdauer einer Instant Messaging Sitzung behalten werden. Eine Sitzung entsteht zwischen einem oder mehreren Wohnungsmietern und dem entsprechenden Wohnungseigentümer oder umgekehrt. Die Entfernung der nicht dauerhaften Queues wird normalerweise dann ausgeführt, wenn keine “Abonnenten” mehr (d.h. Erzeuger und Verbraucher) auf dem entstehenden “Kanal” lauschen. In diesem Fall könnte auch ein Mechanismus definiert werden, der die “ausgelaufenen” Queues in eine dauerhafte Form von Queues umwandelt, jeweils mit eindeutigen Namen (z.B. anhand des Zeitstempels).

Für die erste aufgelistete Art von Queues kann die genannte erneute Zusendung der Mitteilung anhand einer RabbitMQ-spezifischen Erweiterung[12] zum AMQP-Standard erfolgen: Message Receipt Confirmation (auch als Publisher Acknowledgements bekannt). Weiterhin könnten die nicht zustellbaren Mitteilungen in der Benutzeroberfläche beispielsweise hervorgehoben werden, um zu zeigen dass sie während einer Zeitspanne von Inaktivität angekommen sind.

[12] `http://www.rabbitmq.com/extensions.html`

Die Entwicklung der “nicht dauerhaften” Art von Queues ist mehr mit der nicht-relationalen Datenbank namens Redis verbunden, wie im kommenden Paragraph erläutert wird.

Redis ist ein “Server für Datenstrukturen”, der ein In-Memory Datenspeicherungsmodell für eine schnelle Anfragebearbeitung verwendet. Im Vergleich zu anderen “key-value store” Datenbanksystemen ist Redis in der Lage, komplexere Datenstrukturen zu persistieren, wie Maps (Hashes/Dictionaries), Mengen, sortierte Mengen und Listen. Weitere Details sind in [MO11] zu finden.

AMQP-Modell und .Architektur für die Realisierung des Kommunikationskanals

Konkret wurde das folgende Modell für den Kommunikationskanal in der AMQP-Terminologie umgesetzt, zusammen mit dem genannten “NoSQL”, echtzeitfähigen Datenbanksystem namens Redis:

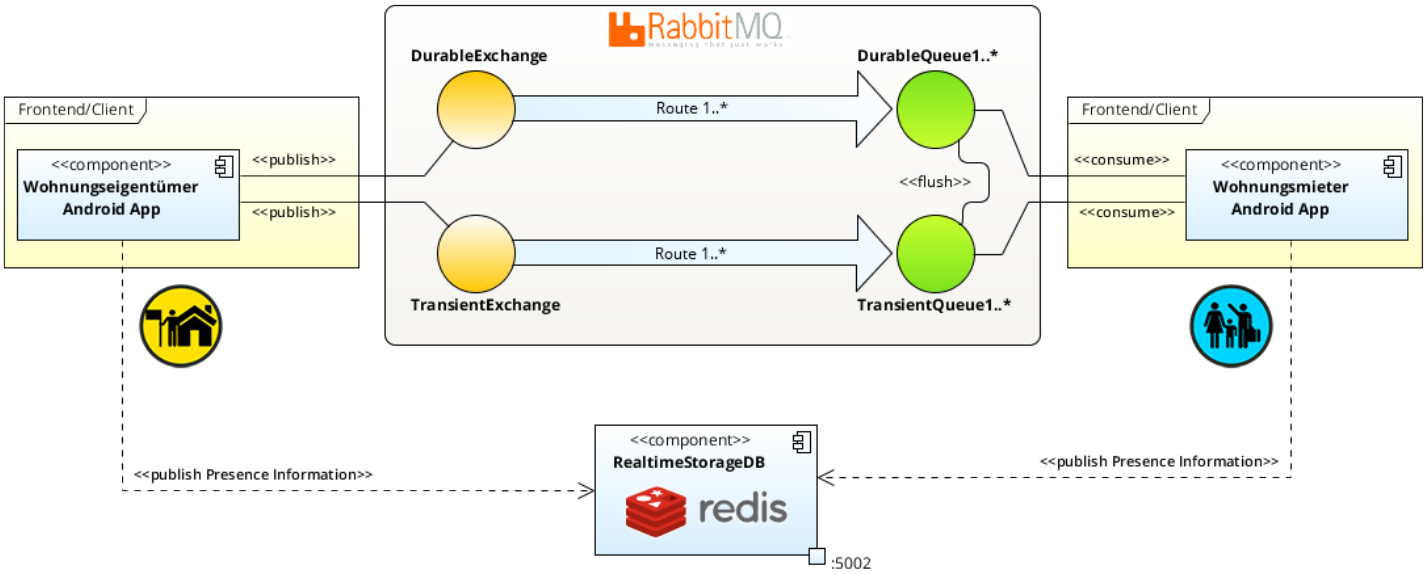


Abbildung 7.2: Überblick des Kommunikationskanals, in der AMQP-Terminologie, mit dauerhaften und kurzlebigen Queues sowie anwendungsspezifischen Erweiterungen.

In der Abbildung 7.2 wird das verwendete AMQP-Modell für den Kommunikationskanal dargestellt. Die Anwendungen können die Rollen wechseln, sodass die Wohnungseigentümer Android App auch in der Lage ist, Mitteilungen zu empfangen und umgekehrt. Es wurde jeweils ein Exchange implementiert, der die dauerhaften (d.h. “Offline” oder persistierte) Mitteilungen und entsprechend die nicht langlebigen Mitteilungen (d.h. Mitteilungen des Typs “Instant Messaging”) zu einer Queue weiterleitet bzw. routet.

Die Skalierbarkeit der Lösung kann durch die Existenz mehrerer Queues begründet werden, obwohl für den Basis-Anwendungsfall nur eine Queue pro Mieter erstellt wird. In weiteren Fällen wie z.B. eine höhere Anzahl von Mietern, die einzelne Kommunikationen mit dem Mieter durchführen möchten, bleibt die Architektur bestehen. Ein weiterer Aspekt der Implementierung ist die Darstellung der “flush” Operation. Hiermit wird der Umgang mit Queues definiert, die von einer nicht dauerhaften

Form (z.B. eine Chat-Sitzung) zu einer dauerhaften Form (z.B. die Archivierung einer Konversation) per "Anfrage" umgewandelt werden sollen.

Des Weiteren wird das nicht-relationale Datenbanksystem "Redis" für die Speicherung und bzw. Anfrage des aktuellen Online-Zustandes eines Benutzers (z.B. Online oder Offline) eingesetzt. Eine weitere Entwicklung des "Präsenz" Konzeptes wäre die Speicherung verschiedener Metadaten über die Präsenz, wie z.B. Ereignisse wie "tippt gerade" (engl. is typing) oder "zuletzt gesehen am .. Zeit" (engl. last seen at).

Die technische Infrastruktur für die Realisierung des Kommunikationskanals ist das `amqp` Ruby Gem, welches nicht nur eine Schnittstelle zum AMQP-Netzwerkprotokoll und zu den dazugehörigen Broker-Systemen bereitstellt, sondern nutzt auch spezifische Erweiterungen von Brokers (wie z.B. die RabbitMQ-Erweiterungen) nutzt.

Zusammenfassung

In diesem Abschnitt wurde die Umsetzung des Kommunikationskanals vorgestellt. Zu beachten ist jedoch die prototypische Gewährleistung eingeschränkter Funktionalitäten. Es wurde nur ein beispielhaftes Szenario der Übersendung von einfachen Nachrichten ("Instante Nachrichten" oder IM) zwischen Wohnungseigentümer und -mieter getestet (d.h. nicht evaluiert).

Der Grund für diese Entscheidung ist die ubiquitäre Präsenz und Nutzung anderer Kanäle für die Kommunikation (z.B. Google Talk / IM-Anbieter, E-Mail oder SMS) von den Benutzern. Andererseits besteht der Vorteil des Konzeptes zum Kommunikationskanal darin, dass eine sichere Kommunikation hinsichtlich der Vertraulichkeit (siehe auch 1.5.5) und Teilnehmerauthentizität (siehe auch die Erklärungen aus Paragraph 7.3.3 und 7.3.5) entstehen kann.

Im folgenden Abschnitt wird ein anderer Teil der Implementierung vorgestellt, welche das Datenmodell für die Client-Anwendungen anhand einer REST-basierten Schnittstelle zur Verfügung stellt.

7.3.5 REST Implementierung

Ein wesentliches Kriterium, das während der Suche nach einem geeigneten Framework für die Umsetzung einer REST API gestellt wurde, ist die "Leichtigkeit" der Lösung. Es wurde ein Framework gesucht, welches die Rack Schnittstelle (siehe auch 5.2.1) umsetzt und ein geeignetes DSL (Domain Specific Language, eine domänenspezifische Sprache, womit Lösungen für ein bestimmtes Problemfeld möglichst kompakt definiert werden können) für die Beschreibung der einzelnen Methoden (die so genannten "Routen") bereitstellt.

Ein erster Gedanke war das Web Framework namens "Sinatra": die zur Verfügung gestellte DSL ist kompakt, jedoch wird Sinatra als ein voll entwickeltes Framework für die Umsetzung Web-basierter Anwendungen (Web Applications, Kurzform "Webapps") angesehen und nicht "nur" als ein Framework für die Definition eines REST API. In dieser Hinsicht wurde die Entscheidung getroffen, eine nicht unbedingt bekannte[13] aber mächtige Softwarelösung für die Definition von REST APIs namens "Grape"[14] anzuwenden.

[13] Grape hat 1900 Watchers und 162 Forks auf Github, während Sinatra wesentlich populärer ist, mit 4160 Watchers und 615 Forks

[14] `http://intridea.github.com/grape/`

Grape ermöglicht die Definition einer REST API durch die Bereitstellung eines “Micro-Frameworks”. Nachdem eine API definiert ist, kann diese Spezifikation auf einem Rack-kompatiblen Web Server installiert und den Client-Anwendungen angeboten werden. Grape unterstützt häufig verwendete Konzepte, die mit der Definition einer REST API verbunden sind, wie z.B. verschiedene Repräsentationen oder Formate: Die Anfrage an eine URI wie `http://localhost/api/index.html` liefert z.B. eine andere Antwort als eine Anfrage an die `http://localhost/api/index.json`) URI. Weiterhin unterstützt Grape die Versionierung einer API, durch die Einbettung der aktuellen Version in die URI (`http://localhost/api/v8/etc`) – alternativ können HTTP Headers oder Parameters verwendet werden.

Weiterhin können verschiedene Namensräume (engl. namespace) definiert werden, in dem die beschriebenen Routen ihre Einzigartigkeit behalten – z.B. bei der Definition der Namensräume “foo” und “bar” werden die Anfragen “GET /foo/my-action” und “GET /bar/my-action” dem entsprechenden Namensraum weitergeleitet bzw. geroutet.

Nachdem die benutzte technologische Grundlage für die Bereitstellung von REST APIs kurz eingeführt ist, werden in den kommenden Abschnitten Implementierungsdetails über die Umsetzung der Kommunikation zwischen dem Backend und dem Frontend gegeben.

Definition einer Basis-API

Ein erster Schritt für die Gewährleistung der Allgemeinheit im Fall der zu entwickelnden REST API ist die Definition einer Basis-API, die allgmein gültige Einstellungen feststellt. Diese Einstellungen werden per Ruby “Einbeziehung” (engl. inclusion, Ruby Schlüsselwort `include`) in die jeweiligen spezialisierten APIs integriert.

Die genannten Einstellungen berühren REST- und HTTP-spezifische Themen, wie z.B. die Benutzung einer einheitlichen Versionnummer für API Methoden (s.o.), die Benutzung einer Default-Repräsentation oder die Feststellung eines Default `Content-Type`. Darüber hinaus werden in der Basis-API verschiedene Grape-spezifische Einstellungen ausgeführt wie die “stumme” Beseitigung eintretender Exceptions durch Logging (siehe auch 7.3.6).

Die Basis-API enthält auch eine Basis-Funktionalität, die von erbenden APIs benutzt werden können: eine Methode zur Serialisierung (s.u.) und ggf. Komprimierung des übergebenen Inhalts. Weiterhin ist die Basis-API für erweiterte Konfigurationen zuständig, wie z.B. die Installation einer Rack-Middleware für die Implementierung des HMAC-basierten Sicherheitskonzeptes (siehe auch den kommenden, dedizierten Paragraph).

Die genannten Einstellungen werden dank einem Feature von Ruby automatisch durchgeführt bei der Vererbung der Klasse `BaseAPI` von einer spezialisierten Klasse. Wie bei einer normalen Vererbung, bleiben die Einstellungen der Vater-Klasse bestehen, während die spezialisierte Klasse eine eigene REST API definiert. Weitere Details zur Spezialisierung werden in einem kommenden Abschnitt gegeben werden.

Definition einer erweiterten API für die Einkapselung beliebiger Bestandteile des Datenmodells

Die Spezialisierung der `BaseAPI` Klasse wird durch die Bestandteile des Datenmodells (siehe auch 7.2) ausgeprägt und ist in der Basis-Klasse `BaseModelAPI` zusammengefasst. Zum Beispiel

erlaubt die `AssetCategory` Klasse, Teil des Datenmodells, die Definition einer Route für eine `GET /tree` Anfrage, welche die Baum-Struktur der einzelnen Kategorien von Gegenständen zurückliefert. Die Art und Weise, in der dies ermöglicht wird, liegt im Fokus dieses Abschnittes.

Der erste Schritt ist das Laden der Modell-Spezifikation – ein `rake`-Task namens `model_rest` steht dafür bereit und erwartet als Parameter die Namen der einzelnen Datenmodelle. Rake (Ruby make, ähnlich zum `make` Werkzeug aus der Programmiersprache C) ist ein Werkzeug zur Verwaltung von Software-Aufgaben (Tasks), die normalerweise routinemäßig ausgeführt werden müssen wie z.B. die Durchführung von Tests oder die Erstellung eines Builds.

Die Funktionalität dieses Tasks ist die Erstellung einer `BaseModelAPI` Klasse, die mittels Metaprogrammierung (s.u.) und Grape eine REST API für das übergebene Datenmodell definiert. Anschließend wird ein Web Server gestartet (siehe auch 7.3.5), welcher die spezifischen Routen dieses Datenmodells zur Verfügung stellt.

Der grundlegende Ansatz der Metaprogrammierung ist die Definition eines bestimmten Quelltextes, der einen anderen Quelltext *selbst erzeugt* (die "Programmierung von Programmierung"). Durch diese Technik kann ein Quelltext definiert werden, welcher besser und flexibler auf unerwartete Ereignisse reagieren kann. Weitere Informationen über das Konzept der Metaprogrammierung werden in der einflussreichen Arbeit "Metaprogramming Ruby" [Per10] gegeben.

Metaprogrammierung wird an dieser Stelle benutzt, um die angeforderten Datenmodelle und deren Routes in eine "monolytische" REST API einzubetten, die Anfragen für alle involvierten Datenmodelle bearbeiten kann. Darüber hinaus werden Hilfsmethoden bereitgestellt, die z.B. für das Logging bestimmter Ereignisse oder die Etablierung einer Verbindung mit der Datenbank zuständig sind. Durch Metaprogrammierung wird das automatische Laden der Datenmodell-spezfischen Routen wie z.B. `GET /spatial_relations` (eine Route für das Datenmodell `Space`, das `SpatialRelations` enthält) gewährleistet. Die Routen, die per Default in der `BaseModelAPI` Klasse verfügbar sind, berühren auf den mit DataMapper mitgelieferten Methoden zur Selektion eines bestimmten Datensatzes (eine Instanz in der Relation): `all` oder `count`.

Durch die Definition der Routen und der Instanz-Logik (Methoden, die für eine Instanz des Datenmodells verfügbar sind), wird eine bessere Verwaltung des Quelltextes ermöglicht. In seiner finalen Form wird ein Datenmodell folgendermaßen in Ruby definiert:

```
# External requires
require "dm-is-tree"

# Internal resource references
require_relative "internal_asset"
require_relative "attribute_set"

# Internal requires
require "./model/logic/data/asset_definition/real_asset_logic"
require "./model/REST/data/asset_definition/real_asset_routes"

class RealAsset < InternalAsset

  # Mark as a DataMapper Resource
  include DataMapper::Resource

  # Include Logic, extend REST Routes (Class level)
  include RealAssetLogic
```

```
  extend  RealAssetRoutes

  # Auto-generated property ID
  property :id, Serial, :key => true
  property :parent_id, Integer, :required => false

  # Single Table Inheritance
  property :type, Discriminator

  # Associations -- has
  has n, :attribute_sets # described by

  # Associations -- belongs
  require_relative "asset_category"
  belongs_to :asset_category, :through => Resource # has_category
  belongs_to :user_profile # possesses

  # Tree
  is :tree, :order => :name

end # class RealAsset
```

Als nächstes wird das Übertragungsformat vorgestellt, welches von dem Backend- und Frontend-Anwendungen als Kommunikationsgrundlage benutzt wird.

Das “Payload” – MessagePack

Das Konzept der “Repräsentation” (siehe auch 2.7) wird in der Praxis meistens als “Payload” (dt. Nutzdaten) bezeichnet. Ein Payload repräsentiert das Format der im Rahmen einer Kommunikation übertragenen Daten – beispiele dafür sind XML, JSON oder MessagePack. Während XML normalerweise mit “alten” Web Services assoziiert werden, ist JSON für ein “Aushängeschild” für die Verbreitung REST-basierter Web Services gesehen. Ein Payload unterstützt normalerweise zwei mögliche Operationen: eine Serialisierung und eine entsprechende Deserialisierung der zu übertagenden bzw. übertragenen Daten.

Ein nicht äußerst bekanntes aber effizientes Payload ist MessagePack[15] (Kurzform MsgPack). Das Team hinter der Entwicklung dieses open-source Projektes gibt bekannt, dass MessagePack bis auf zehn Mal so schneller als JSON sei und fasst dieses Ergebnis als das Tagline “It’s like JSON. but fast and small.” zusammen. Unabhängige Studien[16] zeigen dass die Aussagen des MessagePack Teams zwar begründet werden können, jeweils nicht unbedingt in der angegebenen Maßzahl – es wird nur beschrieben, dass MessagePack im Endeffekt immer noch JSON ist (Rückwärtskompatibel), jedoch mit einer effizienten, binären Kodierung.

Die Alternativen zu MessagePack, die in der zitierten Artikel vorgestellt wurden, haben den Nachteil, dass sie eine streng typisierte Anschauung der Daten erfordern – nachteilig bei der Verwendung einer dynamischen Programmiersprache wie Ruby. Andererseits erfordert den Umgang mit einem nicht typisierten eine Umwandlung (engl. typecasting) im Fall von streng typisierten Programmiersprachen – MessagePack bietet in Java verschiedene Möglichkeiten zur Automatisierung dieses Vor-

[15] http://www.msgpack.org

[16] Artikel “Protocol Buffers, Avro, Thrift & MessagePack”, Erscheinungsdatum: 01. Aug 2011, http://www.igvita.com/2011/08/01/protocol-buffers-avro-thrift-messagepack/

gangs mittels Techniken der Reflection; eigene Tests auf einem mobilen Gerät haben diesen Vorgang als nicht optimal beurteilt. Die gefundene Lösung war die "manuelle" Umwandlung der Daten, anhand Konventionen, was die verwendeten Datentypen angeht, zwischen Backend und Frontend. Des Weiteren werden Ruby SDKs nicht von allen Alternativen angeboten.

Trotz dieser "Marketing-Mitteilung" der Geschwindigkeit gegenüber JSON, setzt MessagePack ein kompaktes, binäres Format für die schnelle Übertragung von Mitteilungen im Rahmen einer Punkt-zu-Punkt (bzw. System-zu-System) Kommunikation. Dieses Konzept wird in Form verschiedener "Bindings" oder SDKs umgesetzt: MessagePack kann in eine erhebliche Anzahl von Programmiersprachen eingesetzt werden, welche die Interoperabilität dieses Produktes zeigt. Jedoch leidet MessagePack (wie viele andere open-source Projekte) unter der "Erkrangung" einer nicht unbedingt detaillierten Dokumentation.

Obwohl verfügbar und behandelnd den meisten Anwendungsfälle, betrachtet die Dokumentation voraussichtlich veraltete Versionen des Produktes. Ein Hinweis ist vom Team an dieser Stelle gegeben – um Beispiele der Nutzung sehen zu können, müssen die geschriebenen Testfälle betrachtet werden. Dies implizit, dass die aktuelle Dokumentation wie gesagt veraltet ist. Trotz dieser Versuch, sich aus der Testfälle zu informieren, werden in den zahlreichen Tests jeweils nur Serialisierungen von vereinfachten Datentypen durchgeführt.

Aus diesem Grund hat die Benutzung von MessagePack einige Probleme bei der Entwicklung des Frontends ergeben – im Fall der Ruby Version des MessagePack Gems können übliche Datenstrukturen (Hashes/Maps, Arrays) per eine Zeile Quelltext serialisieren (`.to_msgpack`) bzw. deserialisieren (`MsgPack.unpack(..)`).

Die Zusammenfassung dieses Paragraphs kann folgendermaßen lauten: MessagePack stellt eine binäre Serialisierungsmethode, die eine bessere Komprimierung der Datenübertragung gewährleisten kann und wird meistens als eine Erweiterung zu einer bestehenden JSON-Serialisierung eingesetzt. Es werden verschiedene Programmiersprachen unterstützt und es gibt eine einigermaßen benutzbare Dokumentation. Darüber hinaus habe eigene Tests gezeigt, dass eine weitere Komprimierung der binären Übertragung weitere Gewinne bezüglich der Geschwindigkeit leisten kann.

Nachdem die zu verschickenden bzw. empfangenen Daten serialisiert bzw. deserialisiert worden sind, muss eine Absicherung der Übertragung hinsichtlich der Nachrichtenauthentizität und der Datenintegrität (siehe auch Paragraph 1.5.5) erfolgen. Dieses Thema wird im folgenden Abschnitt eingegangen.

Überprüfung der Nachrichtenauthentizität und Datenintegrität mittels HMAC

Zur Unterstützung der genannten Sicherheitsziele wird das "rack-authenticate"[17] Gem benutzt, jeweils mit spezifischen Änderungen bezüglich der Performanz und allg. Debugging-bezogenen Konzepten. Wie vorher genannt, unterstützt Rack die Benutzung verschiedener "Middleware" Komponenten – diese werden normalerweise zwischen der Anfrage, das Web Anwendung und die Antwort zur Anfrage platziert aber sie können z.B. auch den gesamten Vorgang abbrechen. Der Zusammenhang von Middleware-Komponenten und die grundlegende Architektur von Rack wird in der Abbildung 7.3 (angelehnt an dem Artikel "Rack Middleware Examples"[18]) veranschaulicht.

[17] `https://github.com/seomoz/rack-authenticate`

[18] Erscheinungsdatum: 04. Juni 2010, `http://ephemera.karmi.cz/post/663716963/rack-middleware-examples`

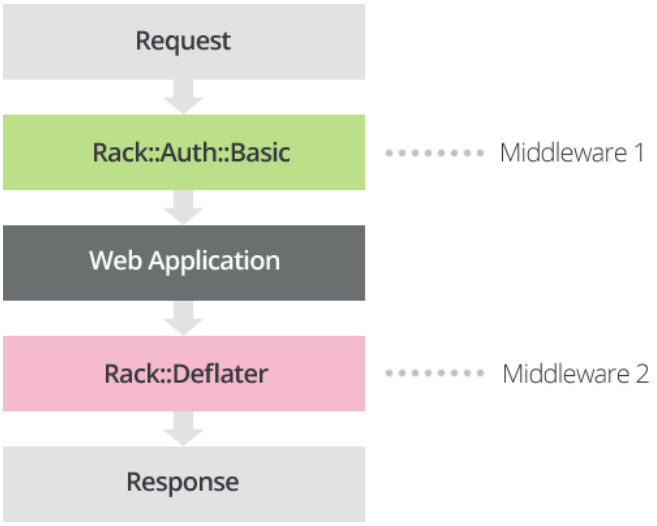


Abbildung 7.3: Überblick beispielhafter Einbindungen von Rack Middleware in der Rack Architektur – als Authentifizierung- oder Komprimierung-Maßnahme

Eine Middleware-Komponente kann auch als ein Filter angesehen werden, die z.B. die ursprüngliche Anfrage abfangen und bearbeiten kann oder die formulierte Antwort einer Anfrage weiter bearbeiten kann. Darüber hinaus kann der gesamte Vorgang unterbrochen werden, wenn z.B. eine Authentifizierung fehlschlagt. Darüber hinaus können Rack Middleware Komponenten für Caching, Überwachung der Performanz und ähnliche Konzepte eingeführt werden.

Der letzte Anwendungsfall (Vorgang unterbrechen) wird im Fall des `rack-authenticate` Gems ausgenutzt: gemäß dem standardisierten Verfahren[19] zur HMAC wird die Bearbeitungskette aus der Abbildung 7.3 abgebrochen und eine `401 Unauthorized` HTTP Antwort zurückgeliefert, falls eine Überprüfung (s.u.) fehlschlägt.

Die Überprüfung einer HMAC-Signatur berücksichtigt die Handhabung verschiedener Parameter wie die private Schlüssel und den dazugehörigen Bezeichner (in diesem Fall der Name der kontextbewussten Anwendung, Klasse `ContextAwareApplication`), das Datum, an dem die Anfrage formuliert wurde, sowie im Fall von `POST` Anfragen, die Handhabung des MD5 Wertes. Anhand dieser Parameter wird zur hauptsächlichen Überprüfung der privaten Schlüssel gelungen – in dieser Phase wird der Bezeichner benutzt, um das in der Datenbank gespeicherten Wert (die private, von Frontend und Backend geteilte Schlüssel) auszulesen. Der letzte Schritt bei der HMAC Authentifizierung ist die erneute Berechnung des Hashwertes anhand der ausgelesenen privaten Schlüssel. Wenn das entstehende Wert mit dem übermittelten übereinstimmt, wird die Anfrage als authentifiziert bezeichnet und weiter in die Bearbeitungskette zugelässt.

Die genannten Erweiterungen zu dem `rack-authenticate` Gem wurden temporär in der Entwicklungsphase im Quelltext durchgeführt, um die Überprüfungsphase der HMAC-Signatur einzuteilen, zu dem Zweck, eine eventuelle fehlerhafte Überprüfung identifizieren zu können. Darüber hinaus wurde eine Erweiterung der Hashwert-Berechnung (s.o.) durchgeführt, in dem die performantere, mit Ruby 1.9 mitgelieferte Bibliothek `OpenSSL` für die Berechnung des Hashwertes angewendet wurde.

[19] RFC 2104: `http://www.ietf.org/rfc/rfc2104.txt`

Ursprünglich wurde eine externe Bibliothek namens `ruby-hmac` für die Berechnung der SHA1-256 Hashwertes angewendet, welche eine wesentlich geringere Performanz zeigt[20].

Nachdem die wesentlichen Aspekte der Umsetzung der REST API vorgestellt wurden, werden in den kommenden Abschnitten Details gegeben über die Aufstellung der API mithilfe eines Web Servers sowie die Beschreibung der verfügbaren Routes als die Beantwortung einer `OPTIONS` Anfrage.

Beschreibung und Entdeckung von REST Web Services

Ein grundlegendes Prinzip des Grape mini-Frameworks ist die *Kompilierung* von Routen, ein Prinzip welches auch anderen Web Frameworks zugrunde liegt. Der Name dieser Technik deutet an, dass die definierten Routen eines REST APIs bevor die Aufstellung (engl. deployment) auf einem Web Server "abgeschlossen" werden müssen. Das heißt, es können außer der "abgeschlossenen", finalisierten Routen keine andere Route behandelt. Zu diesem Zweck werden alle deklarierten Routen einer REST API in einer Rack-konforme Anwendung zusammengefasst. Durch die Kompilierung der Routen werden diese stets statisch (Routen können auch anhand von variablen Parametern wie z.B. `POST /model/:action/:id` definiert werden. Diese Route besagt, dass die Parameter namens "action" und "id" aus der ankommenden Anfrage auszulesen sind), was die Bedeutung von "auswertbar" enthält.

Die "Auswertung" der Routen wird auch von einem Grape-Plugin[21] benutzt. Dieses Plugin dient als ein Dokumentations-Werkzeug für die Beschreibung einer mit Grape definierten REST API und kann mit dem "Swagger"[22] Werkzeug zum Testen von Web Services zusammenarbeiten. Anhand des Prinzips der Routen-Kompilierung wurde in dieser Arbeit eine Erweiterung zur Beschreibung und eventuelle Entdeckung vorhandener REST Web Services, definiert, welche im Folgenden beschrieben wird.

Durch Metaprogrammierung wurden in vorhandenen REST APIs (bzw. `BaseAPI` sowie `BasicModelAPI`) grundlegende HTTP Methoden zur Beschreibung der Funktionalität der entsprechenden API eingeführt. Beispielsweise liefert die Anfrage `OPTIONS /` eine JSON-kodierte Array mit dem folgenden Format zurück:

```
[version: "v1", method:"GET", path:"/:version/set/visited(.:format),
version: "v2", method:"DELETE", path:"/:version/:id/:recurse(.:format)]
```

Hiermit wurde durch das Hash erkenntlich gemacht, welche Routen für die ausgewählte Namespace verfügbar sind. Dabei enthalten sind alle benötigten Informationen: die Version der API, die zu benutzende HTTP Methode sowie das erwartete Format der URI.

Die Entdeckung vorhandener Services könnte mittels derselben vorgestellten Lösung erreicht werden: durch wiederholte Formulierungen von `OPTIONS` Anfragen (anfänglich an einen Dienst-Gateway und darauffolgend gezielt an eine Namespace von Interesse) könnten theoretisch neue durch REST Web Services veröffentlichte Dienste entdeckt werden.

[20] Artikel "OpenSSL::HMAC vs ruby-hmac Benchmarks", Erscheinungsdatum: 04. Dez 2008, `http://blog.nathanielbibler.com/post/63031273/openssl-hmac-vs-ruby-hmac-benchmarks`
[21] `https://github.com/tim-vandecasteele/grape-swagger`
[22] `http://swagger.wordnik.com/`

Entdeckung neuer Services und Widgets

Eine Weise, in der die Entdeckung neu angebundener oder vorhandener Services (im Fall des Backends) erfolgen könnte, basiert auf der Fähigkeit einer Programmiersprache bzw. eines Programms, eine "Reflexion" (engl. reflection) durchzuführen. Durch diese Prozedur kann die vorhandene Struktur eines Programms (d.h. im Fall einer objektorientierten Programmiersprache einer Klasse) untersucht werden und damit ihr Verhalten (die Werte, Metadaten, einzelne Felder und Methoden) zur Laufzeit verändert werden.

Damit lassen sich z.B. Subklassen einer spezifischen Klasse (oder im Fall von strikt typisierten Sprachen, einer spezifischen Schnittstelle) zur Laufzeit ermitteln. Diese Herangehensweise konnte wie im Fall der vorher vorgestellten REST-basierten Entdeckung fortlaufend durchgeführt werden, bis das erzielte Ergebnis oder ein Grenzfall entdeckt ist. Ein Beispiel dieser Methode ist im Datenmodell vorhanden, wo das Frontend eine Anfrage für die Auflistung der Subklassen der Klasse `SpatialRelation` formuliert (siehe auch Absatz 7.4).

Die Ausführung von Reflection-basierten Verfahren ist sowohl im Fall des Backends als auch im Fall des Frontends verfügbar – hiermit können z.B. neue, zur Laufzeit noch nicht bekannte Widgets, die auf dem mobilen Gerät laufen, entdeckt werden. Weiterhin könnte dieser Typ von Entdeckung auch als eine REST API im Fall des Backends veröffentlicht werden, z.B. mit der Anfrage `GET /app/HomeSharing1/available_widgets/subclass/IMyWidget`.

Veröffentlichung der REST API

Die Bereitstellung bzw. Veröffentlichung der vorher beschriebenen REST APIs erfolgt mithilfe eines Rack-kompatiblen Web Servers. Die beschriebene "Web Anwendung" und dessen Routen werden nach außen anhand einer festgestellten IP-Adresse (siehe auch 7.3.3) und eines Ports zur Verfügung gestellt. Die Konfiguration ist aus einer Datei gelesen, wie weiter im Paragraph 7.3.6 beschrieben wird.

Bei der Auswahl eines Web Servers wurde darauf geachtet, dass die Softwarelösung die Fähigkeit behält, mehrere gleichzeitige Verbindungen anzunehmen, zur Gewährleistung der Performanz bei einer erhöhten Anzahl von Benutzern (Skalierbarkeit). Die aus der Welt der "Ruby on Rails" Web Application Framework bekannten "minimalistischen" Web Server-Lösungen (WEBRick oder Mongrel) unterstützen diese Anforderung nicht. Weitere Lösungen[23] wie Passsenger oder Unicorn sind für fortgeschrittene Zwecke, wie die Veröffentlichung von Web-basierten Enterprise-Software geeignet.

Der richtige Ausgleich zwischen mitgelieferten Features und Leichtgewicht wird von dem Web Server namens "Thin"[24] geliefert, welcher sich als ein schneller, leicht bedienbares Web Server für Ruby beschreibt. Thin fasst Ideen aus mehreren Technologien für Ruby-basierte Web Entwicklung in einer vielseitigen Software-Lösung zusammen.

Thin wird in Form eines Ruby Gems veröffentlicht und bietet eine vereinfachte Benutzung an. Um eine Instanz des Web Servers zu starten werden nur die genannten Parameter (IP-Adresse und Port) und die entsprechende Rack-konforme Anwendung benötigt. Damit entsteht aus der Perspektive des Betriebssystems, worauf der Server betrieben wird, ein neuer Prozess, welcher den Web Server hostet bzw. der übergebenen Adresse und dem Port lauscht. Thin ermöglicht auch eine Art des "internen"

[23] `https://www.ruby-toolbox.com/categories/web_servers`
[24] `http://code.macournoyer.com/thin/`

Loggings (ein “debug” Modus), in dem einzelne Anfragen, Antworten und Reaktionsszeiten in der Konsole ausgegeben werden.

7.3.6 Sonstige Funktionalität des Backends

Im Folgenden werden einige Details über die sonstige von der Backend-Komponente angebotene Funktionalität gegeben.

QR-Code Generierung

Eine sonstige Funktionalität des Backends wird von der Unterstützung der Indoor-Lokalisierung repräsentiert. Hiermit wird ein Eingangswert (z.B. eine String, die eine URI kodiert) entgegengenommen und mit Hilfe des Ruby Gems namens `barby`[25] eine Darstellung des Wertes als QR-Code als Antwort zurückgeliefert.

Das Format der Antwort ist ein PNG-Bild, in einer mittleren Größe. Kodiert wurde das übergebene Wert (meistens eine URI, die für andere Anwendungen oder Szenarien keinerlei Bedeutung tragen sollte, wie z.B. `/real_asset/42` – hiermit wird kodiert, dass der QR-Code ein `RealAsset` des Datenmodells, mit der ID 42 repräsentiert).

Weitere Entwicklungen können an dieser Stelle genannt werden, wie z.B. die direkte Untersützung des PDF-Formats, welches vom genannten Gem auch unterstützt wird. Dieser Anwendungsfall könnte auch für die Implementierung des REST-Prinzips für verschiedene Repräsentationen derselben Route (siehe dazu Paragraph 2.7) verwendet werden. Weiterhin könnte die Speicherung der generierten Datei, die der QR-Code enthält, direkt auf dem Server entstehen.

Diese Funktionalität des Backends wird für die Generierung von QR-Codes eingesetzt, die eine vereinfachte Art von Indoor-Lokalisierung bereitstellen, wie im Paragraph 7.4.3 weiter erläutert wird.

Logging

Das Backend setzt die vorher aufgeführten Konzepte des Logging-Verfahrens (siehe auch Paragraph 7.1) um, indem das genannte Werkzeug namens `Log4r` um einige Funktionalitäten erweitert wurde.

Die Funktionalitäten erlauben die Vererbung von Logging-Eigenschaften einer Vater-Klasse (z.B. Pfad der zu speichernden Datei) zu der neuen Kind-Klasse. Darüber hinaus wurden einige Vereinbarungen in die Implementierung des Logging-Verfahrens eingebettet wie z.B. die automatische Speicherung der Logging-Datei mit der Dateimaske `<NameDerKlasse>.log.xml` unter dem Ordner `log` in der Projektstruktur. Des Weiteren wurden Kurzformen der benötigten Methoden zum Logging in jeder aufrufenden Klasse mittels grundlegender Metaprogrammierungs-Verfahren definiert, sodass die Erstellung eines neuen Eintrags in die Logging-Datei einfach und kurz erfolgt.

Konfigurationsdateien

Zur Erleichterung der Eingabe sämtlicher Konfigurationsparameter wie z.B. Zugangsdaten, IP-Adressen und dazugehörige Ports wurde ein Mechanismus entwickelt, der den Vorgang des Ausle-

[25] `https://github.com/toretore/barby`

sens und die Benutzung dieser Konfigurationsdateien automatisiert. Als Format für die Speicherung der einzelnen Dateien wurde YAML ausgewählt, aufgrund guter Lesbarkeit und vereinfachter Lese-Mechanismen.

Zugleich wurden einige Vereinbarungen definiert, die z.B. die Präsenz einer Konfigurationsdatei per Default unter dem Ordner `config` in der Projektstruktur, mit der jeweiligen Dateimaske `<NameDerKlasse>.yaml` regeln. Diese Vereinbarungen ermöglichen eine automatische Auswertung einer Konfigurationsdatei: es wird gezielt nach der entsprechenden Konfigurationsdatei in der Projektstruktur gesucht, mit einer Toleranz von höchstens einem Niveau.

Bei der Einbeziehung einer Konfigurationsdatei in der Klasse `My::Test::Cls` wird beispielsweise unter dem Pfad `config/my/test/` nach einer Datei namens "cls.yaml" gesucht. Wenn die Datei YAML-konform ist, lassen sich die ggf. gespeicherten Einstellungen mittels der Methode "get_configuration" in der Klasse `Cls` abrufen. Die genannte Methode wird per Metaprogrammierung in der `Cls` Klasse dynamisch definiert.

7.3.7 Zusammenfassung

In diesem Unterkapitel ist die Umsetzung der wesentlichen Funktionalitäten der "Backend" Komponente aus der Softwarelösung ausführlich vorgestellt worden. Damit ist die technische Infrastruktur bereitgestellt, die für die Umsetzung des Hauptkonzeptes dieser Arbeit erforderlich ist. Die Integration des Backends in die gesamte Softwarelösung erfolgt mit der Umsetzung und Bereitstellung von mobilen Anwendungen für den Wohnungseigentümer sowie für die Wohnungsmieter. Die Umsetzung der "Frontend" bzw. der mobilen Komponente ist das Thema, welches im kommenden Absatz behandelt wird.

7.4 Frontend

Die weiteren Systemen, die im gesamten Konzept der Softwarelösung eine Rolle spielen, sind von den mobilen, auf der Android-Plattform entwickelten Anwendungen repräsentiert. In diesem Abschnitt werden die zwei Anwendungen aus Sicht ihrer Umsetzung vorgestellt.

7.4.1 Architektur

Die Architektur des Frontends orientiert sich an dem klassischen MVC (Model–View–Controller) Architekturmuster. Hiermit werden Views (Aktivitäten und Layouts) dargestellt, die deren Verhältnisse in Controllers (z.B. ein Controller für die Kommunikation mit dem Backend) einkapseln. Für die lokale, temporäre Persistierung der Übertragungsdaten werden leichte Modelle eingesetzt, die nur Werte der einzelnen "richtigen" Modelle enthalten (in der SOA-Terminologie werden diese Typen von Objekten als Value Objects, VOs angesehen). Somit werden in den Modellen nur Funktionen rund um die Serialisierung der Daten angeboten.

Controller

Ein erster wichtiger Bestandteil der Architektur wird von den Controllers repräsentiert. Die Aufgaben eines Controllers bewegen sich im Umfeld der Daten-Kontrolle: in einem Controller werden bestimmte Vorgänge abgewickelt und dessen Ergebnisse weiter zu den Views eingereicht. Dabei verwendet der Controller die für den Zweck der behandelten Aufgabe relevanten Modelle.

Im konkreten Fall wurden folgende Controllers definiert:

- **BackendCommunicationControllerBase**: In diesem Controller werden Vorgänge für die Kommunikation mit dem Backend definiert. Eingekapselt wird eine synchrone und asynchrone Art der Kommunikation, mittels der Bibliotheken *android-async-http*[26] sowie die in Android SDK eingebettete Bibliothek namens *HTTP Client*[27] aus dem Apache-Projekt.

  Bereitgestellt von diesem Controller sind Fehlerbehandlungen, die Aktualisierung des Aktivitäts-Symbols im "Hauptmenü", die Generierung und Einbettung von HMAC Signaturen sowie entsprechend die Zusendung der HTTP Anfragen.

  Eine weitere Entwicklung im Rahmen dieses Controllers wäre die Untersuchung anderer Bibliotheken zur Steuerung von ausgehenden HTTP Anfragen (engl. Requests), wie z.B. die leichtgewichtige, vom Google Android Team entwickelte Bibliothek namens `HttpUrlConnection`. Dieser alternative Weg könnte aus Sicht der Performanz getestet werden – eine Hypothese ist, dass die Optimierung des Vorgangs anhand von Android-spezifischen Konstrukten stattfindet.

- **Model-Spezifische Controllers**: Des Weiteren wurden spezifische Handlers von REST-Anfragen für jedes Model (s.u.) entwickelt, die die Ereignisse (z.B. Anfrage schlug fehl, Anfrage mit folgendem Ergebnis [erfolgreich] beendet usw.) dem View (über das Model) mitteilen. Jeder spezifische Controller erbt von der genannten `BackendCommunicationControllerBase` und setzt je nach Anfrage-Typ (eine binär, MessagePack-getriebene oder eine normale Datenübertragung) eine Schnittstelle um – entweder `BackendAsyncBinaryHTTPResponseHandler` oder `BackendAsyncHTTPResponseHandler`.

- **AppPreloadController**: Dieser Controller wird für die Überprüfung spezifischer Vorbedingungen bei dem Start der jeweiligen Anwendung angewendet. Die zu überprüfenden Vorbedingungen sind folgende:

  - Überprüfen, ob die Anwendung auf dem End-Gerät zum ersten Mal läuft. Hiermit können z.B. spezifische Maßnahmen ergriffen werden, wie die Darstellung einer Anleitung zur ersten Anwendung oder die Darstellung eines Formulars zum Eingeben eines minimalen sozialen Profils (siehe auch 3).
  - Überprüfen, ob Wi-Fi aktiviert ist oder nicht. Wenn noch nicht aktiviert, wird diese Wahl automatisch betätigt.
  - Überprüfen, ob eine Wi-Fi Verbindung erfolgreich etabliert werden konnte. Falls die Verbindung nicht hergestellt werden konnte, wurde der Nutzer zum Android-spezifischen Menü weitergeleitet, das für die Wi-Fi-Verbindung zuständig ist. Diese Wahl wird getroffen, um sicherzustellen dass eine Verbindung mit dem lokalen Netzwerk erfolgreich hergestellt werden kann.

[26] `http://loopj.com/android-async-http/`

[27] `http://developer.android.com/reference/org/apache/http/client/package-summary.html`

Diese Überprüfungen basieren auf zwei Schnittstellen, nämlich der `IOnPrerequisiteCompletedListener`, welcher eine Ruckruffunktion (engl. callback) bereitstellt für den Fall, in dass die Überprüfung zu Ende gekommen ist. Außerdem steht derentsprechende `IOnPrerequisiteUpdatingListener` zur Verfügung, um das View und implizit den Benutzer über den aktuellen Fortschritt des Überprüfungsvorgangs informieren zu können. Eine Überprüfung-Klasse erbt von der Basis-Klasse `PrerequisiteVerificationBase`.

Weitere Controllers sind nicht eingeführt, diese werden eventuell nach einem erneuten Refactorisierungsdurchlauf benötigt, um z.B. den Umgang der Anwendung mit lokal vorhandener Sensorik zentral steuern zu können (durch z.B. ein `HardwareSensorController`).

Model

Abgesehen davon, dass die gesamte Architektur der Softwarelösung (vorgestellt in Kapitel 5) sich aus verschiedenen Systemen zusammensetzt, werden bestimmte Aspekte des Kontext-Modells in verschiedenen Orten gespeichert, angezeigt und/oder bearbeitet. Im Fall des Datenmodells (siehe auch Abschnitt 7.2) werden die einzelnen Modelle im Frontend erstellt und befüllt, während die entsprechende Persistierung dieser Modelle im Backend erfolgt. Der umgekehrte Weg gilt auch: die persistierten Modelle lassen sich aus dem Backend bzw. aus der Datenbank lesen und zu dem Frontend übertragen.

Für die beiden Szenarien (Erstellung und Speicherung bzw. Lesen und Darstellen) muss zuerst eine Darstellung des Modell-Zustands entstehen. Hierbei werden leichtgewichtige Modelle erstellt, die im Frontend nur ein “Snapshot” des Modells zu einem gewissen Zeitpunkt anbieten, zusammen mit der Möglichkeit, diesen Zustand entweder zum Backend zu verschicken oder entsprechend auszulesen. Die “schlanke” Aufstellung der Modellen bietet den Vorteil, dass eine Serialisierung und bzw. eine Deserialisierung der Daten (wie im Abschnitt 7.3.5 erläutert) mit einer verbesserten Geschwindigkeit ausgeführt werden kann.

Eine konkrete Ausprägung eines Modells wird in der folgenden Auflistung gegeben:

```
package edu.uni_ol.msc.pitu.model;

// NOTE: .. Imports weggelassen ..

public class Asset implements MessagePackable
{
  // region Public Attributes

  public Integer ParentId;
  public boolean IsAtRootLevel;

  public String  Name;
  public String  Description;

  public boolean IsVisible = true;
  public String  UsagePermission;

  public List<AssetAttribute> Attributes;
  public List<AssetRule> Rules;
  public List<AssetInstruction> Instructions;
```

```
// endregion

// region Constructors

/**
 * Default Constructor, initializes members.
 */
public Asset()
{
  Attributes   = new ArrayList<AssetAttribute>();
  Rules        = new ArrayList<AssetRule>();
  Instructions = new ArrayList<AssetInstruction>();
}

// endregion

// region MessagePackable Implementation

@Override
public void readFrom(Unpacker pUnpacker) throws IOException
{
  // NOTE: .. Wegen Datei-Größe nicht aufgelistet ..
}

@Override
public void writeTo(Packer pPacker) throws IOException
{
  // Prepare result
  Map<String, String> out = new HashMap<String, String>();

  // Put common attributes -- Parent, IsAtRoot
  if (ParentId != null)        out.put("Parent", ParentId.toString());
  out.put("IsAtRootLevel",     Boolean.toString(IsAtRootLevel));

  // Name and description
  if (Name != null)            out.put("Name", Name);
  if (Description != null)     out.put("Description", Description);

  // Usage permission, is visible
  if (UsagePermission!= null)  out.put("UsagePermission", UsagePermission);
  out.put("IsVisible",         Boolean.toString(IsVisible));

  // Add lists
  if (Attributes.size() > 0)   collectionToMsgPack("Attributes", out);
  if (Rules.size() > 0)        collectionToMsgPack("Rules", out);
  if (Instructions.size() > 0) collectionToMsgPack("Instructions", out);

  // Write result as MsgPack'd map
  MsgPackUtils.writeStringMapTo(pPacker, out);
}

@SuppressWarnings("unchecked")
private void collectionToMsgPack(String pKey, Map<String, String> pDestinationMap)
{
  // Retrieve List in question using Reflection
  List<IToStringExAble> sourceList =
    (List<IToStringExAble>) AssortedUtils.getValueOfFieldNamed(pKey, this);
```

```
    // When found,
    if (sourceList != null)
    {
      // Output the items as given by a ToStringEx() call
      StringBuilder out = new StringBuilder();
      for (IToStringExAble currentItem : sourceList)
      {
        // toStringEx() uses a simple tokenizing Method to serialize Attributes in a
        // comma-separated Value String (since nesting two Maps in MsgPack did not work)
        String stringExResult = currentItem.toStringEx();
        if (stringExResult != null)
        {
          out.append(stringExResult);
        }
      }

      // Update destination map
      pDestinationMap.put(pKey, out.toString());
    }
  }

  // endregion
}
```

Die Art und Weise in der diese leichtgewichtigen Modelle benutzt werden, wird als nächstes im "View" Paragraph behandelt.

Darüber hinaus werden Ausschnitte des Kontext-Modells auch im Frontend umgesetzt. Dies geschieht zuerst in Form von Services wie bereits im Absatz 7.4.1 erläutert aber auch als leichtgewichtige Modellen, in derselben "schlanken" Art wie kürzlich dargestellt. Die Aufgabe dieser Art von Modellen ist, ein "Snapshot" eines Vorgangs wie z.B. die Erfassung von Sensoren-Daten zu modellieren, sodass diese Daten (der Zustand des Modells) weiter zum Backend übertragen oder lokal zwischengespeichert werden können.

Eine weitere Art von eingesetzten Modellen ist von einem beispielhaften Modell repräsentiert, welches den Zustand der lokalen Datenbank (bei Android, SQLite[28]) modelliert. Beispielsweise können hier Statistiken wie Anzahl der Benutzungen oder Präferenzen für die Anwendung gespeichert werden.

View

Gemäß des Android-Konzeptes sind Views in Form von Aktivitäten (engl. Activities) ausgeprägt. Die Gestaltung einer Aktivität entsteht mithilfe von Layouts, die ein "What you see is what you get" Konstrukt repräsentieren. Das heißt, ein Layout bestimmt die visuelle Wiedergabe, die dem Benutzer vorgestellt wird. Basierend auf den Konzepten zum Nutzungserlebnis (siehe auch Abschnitt 5.4) wurden verschiedene Komponenten entwickelt, die die Darstellung und entsprechende Benutzung der Anwendung erleichtern. Die folgende Auflistung fasst die entwickelten UI-Komponenten zusammen:

- Eine klappbare Liste, die die Grundriss-Komponente des Android Frameworks namens `ExpandableListView` um die Funktionalität zur Hinzufügung, Löschung und Bearbeitung von Items

[28] `http://www.sqlite.org`

erweitert. Die ursprüngliche Komponente wurde nur mit einer grundlegenden Funktionalität zur Darstellung von Items entwickelt.

- Verschiedene Hilfskomponenten, wie z.B. eine Combo-Box (als "Spinner" in der Android-Terminologie bekannt), die einen Titel, einen Untertitel und ein Symbol (Icon) unterstützt. Diese Komponente wird z.B. für die Darstellung der Privatsphäre-Merkmale wie `ASK_FIRST` in einer visuellen und textuell ausführlichen Form nützlich. Weitere Details über die Situationen, in denen diese Hilfskomponente nötig sind, werden im Paragraph 7.4.2) gegeben.

- Darüber hinaus wurde eine Komponente namens `SwipeView` (an einer open-source Komponente angelehnt) und deren Hilfsklassen (`SwipeViewPopulate` sowie `SwipeViewNavigate`) zur Navigation in einer verschachtelten Datenstruktur (wie z.B. eine Baum-Struktur) entwickelt. Diese Komponente ermöglicht mithilfe von Gesten wie z.B. einer "Swipe"-Geste eine Bewegung durch die implizit dargestellte Hierarchie. Der Anwendungsfall für diese Komponente wird im Paragraph 7.4.2 genannt.

- Weiterhin wurde eine Komponente für die Darstellung eines Buttons für das Dashboard (siehe auch 7.4.2) und eine dazugehörige Anzahl von "ungelesenen Ereignissen" bereitgestellt. Schließlich ist eine Komponente zum visuellen Feedback über den Fortschritt eines asynchronen Vorgangs nennenswert.

Eine Übersicht der einzelnen entwickelten Activities und ihrer dazugehörigen Funktionalität und Zielsetzung wird in 7.4.2 verschafft.

An dieser Stelle zu bemerken ist die Verwendung einer einheitlichen, an die aktuellen Entwicklungen des Android Designs (nämlich die Einführung des "Holo" Designs und der `ActionBar` Komponente[29] ab Android Version 3.0) angepassten visuellen Darstellung. Zu diesem Zweck wurde die Bibliothek namens `ActionBarSherlock`[30] angewendet. Dies liegt wesentlich darin begründet, fortgeschrittene Konzepte der neuen Versionen der Android Plattform wie z.B. `Fragments`[31] nutzen zu können, trotz der Entwicklung und dem Testen auf einer alten Version der Plattform (Gingerbread, 2.3.7). Darüber hinaus wurde versucht, durch die benutzten visuellen Elemente (wie z.B. Farben, Schriftart, Wohungseigentümer und -Mieter Symbole) ein Gefühl der "Markenentwicklung" (engl. branding) herbeizuführen.

Feststellung der zu kompilierenden Anwendung

Wie bei der Beschreibung des Backend-Systems genannt, wird ein automatisierter Mechanismus zur Vorbereitung der Aufstellung (engl. deployment) verwendet, die eine Quelltext-Datei und die dazugehörige Manifest-Datei (siehe auch 5.3.2) aktualisiert. Durch diese Aktualisierung werden visuelle Merkmale wie das Symbol der Anwendung in dem "App Drawer" (das Gitter zur Auswahl einer auszuführenden Anwendung oder App in dem Android mobilen Betriebssystem) oder die auf dem Dashboard (siehe auch 7.4.2) verfügbaren Buttons aktualisiert.

[29] Artikel "Say Goodbye to the Menu Button", Erscheinungsdatum: 26. Jan 2012, `http://android-developers.blogspot.com/2012/01/say-goodbye-to-menu-button.html`

[30] `http://actionbarsherlock.com/`

[31] Artikel "The Android 3.0 Fragments API", Erscheinungsdatum: 03. Feb 2011, `http://android-developers.blogspot.com/2011/02/android-30-fragments-api.html`

Neben der Manifest-Datei wird die Klasse `TargetSpecification` aktualisiert. Der Wert der booleschen Konstante namens `TARGET_IS_HOST` wird entsprechend überarbeitet. Dieser Wert ist für die Anpassungen der Anwendung an den aktuellen Benutzertyp (Wohnungseigentümer oder -mieter) in der ganzen Anwendung zuständig.

Nachdem die Hauptmerkmale der Architektur des Frontend-Systems genannt wurden, lassen sich die angebotenen Funktionalitäten dieses Systems aus dem kommenden Absatz entnehmen.

7.4.2 Funktionalität der Frontend-Anwendungen

Der Startpunkt in die Diskussion über die angebotenen Funktionalitäten der Frontend-Anwendungen (Wohnungseigentümer- bzw. Mieter-Anwendung) ist von der Wohnungsmieter-Anwendung repräsentiert.

Wohnungseigentümer-Anwendung

Die Wohnungseigentümer-Anwendung stellt dem End-Benutzer folgende zusammengefasste Android-Aktivitäten (und dementsprechende Funktionalitäten) bereit:

- `AssetCategoryChooseActivity`: Hiermit wird zuerst eine Anfrage ans Backend verschickt, die die Baum-Struktur der aktuell vorhandenen `AssetCategories` zurückliefert. Nach der Deserialisierung entsteht eine visuelle Darstellung der Baum-Struktur (Namen, Beschreibung sowie Symbol, wenn vorhanden) mithilfe der genannten `SwipeView` Komponente. Gesten für die Navigation und Betätigung einer Auswahl werden bereitgestellt: die Auswahl einer Kategorie (d.h. die Bestätigung, dass eine Instanz der Kategorie wie z.B. ein Balkon oder ein Rechner hinzuzufügen ist) erfolgt per ‘Long Tap”-Geste. Die Navigation “innerhalb” der Kategorie (d.h. die Ansicht der Kind-Knoten) erfolgt mittels einer “Tap”-Geste.

  Falls die ausgewählte Kategorie ein Vater-Knoten ist, wird nach der Absicht des Benutzers gefragt – ob eine neue Kategorie (Erweiterung des Datenmodelles) innerhalb der ausgewählten Kategorie erstellt werden sollte oder ob die bestehende Kategorie (z.B. die gesamte Wohnung oder das gesamte Gebäude in dem die Wohnung sich befindet) bearbeitet werden. Nach der Betätigung einer Auswahl gelangt der Benutzer zur nächsten Aktivität, welche im Folgenden behandelt wird.

- `AssetAddActivity`: Durch diese Aktivität werden Informationen über die zu erstellene bzw. zu erweiternde Instanz einer `AssetCategory` erfasst. Es werden drei Tab-Komponenten zur Verfügung gestellt, die in einem entsprechenden Absatz als nächstes beschrieben werden.

  Neben der Zuweisung eines Namen, einer Beschreibung, oder dem Hinzufügen eines oder mehrerer Photos stehen dem Benutzer zur Möglichkeiten zur Veröffentlichung der Instanz (z.B. des Gegenstandes) bereit. Darüber hinaus lässt sich eine Eingabe über die permissive oder nicht permissive Verwendung (`USAGE_PERMISSION`) der Instanz über eine Combo-Box-Komponente machen.

  Die Erfassung von Metadaten wird unterstützt durch mehrere `ExpandableListViews`, die jeweils die Veränderung der erfassten oder vorhandenen Attribute, Regeln sowie Anwendungshinweise ermöglicht. Ein genanntes Merkmal wie z.B. ein neues Attribut (im Datenmodell als `Attribute` gekennzeichnet) kann mithilfe eines Dialogfensters erstellt werden. Die genannten

Bearbeitungsmöglichkeiten (Löschen, Bearbeiten) sind in jeder Kategorie von genannten Metadaten vorhanden.

Schließlich lässt sich die Lokation der Instanz feststellen. Hierfür wird eine Liste der aktuell modellierten Räume (wenn vorhanden) bereitgestellt. Der Benutzer kann eine Auswahl betätigen und dadurch die textuell dargestellte Entscheidung "Dieser Gegenstand befindet sich innerhalb dieses Raumes" treffen. Eine weitere Entwicklungsmöglichkeit lässt sich an dieser Stelle benennen: anstatt einer statischen Lokation könnte eine Visualisierung der aktuell berechneten Lokation des Benutzers innerhalb der Wohnung vorgestellt werden.

Die Speicherung der modellierten Instanz erfolgt per Knopfdruck des "Save" Buttons. Der Benutzer wird über den Fortschritt der Speicher-Aktion visuell informiert.

- `SpaceAddActivity`: Durch diese Aktivität wird dem Benutzer ermöglicht, einen symbolischen Raum (vgl. Lokationsmodell im Paragraph 6.2.2) hinzuzufügen. Eine Anfrage ans Backend wird als Vorbereitungsschritt verschickt (währenddessen wird ein Ladesymbol dargestellt), um die aktuell verfügbaren Räume zu erhalten. Diese Räume sind im Fall von relativen Räumen (Lokation anhand einer räumlichen Beziehung) relevant.

  Dem Raum kann ein Namen gegeben werden, sowie eine freiwillige Anzahl von räumlichen Beziehungen zu einem anderen Raum. Hierfür wird eine `ExpandableListView` bereitgestellt, die Operationen auf vorhandenen räumlichen Beziehungen oder das Einfügen einer neuen Beziehung unterstützt.

  Die Speicherung des modellierten Raumes erfolgt per Knopfdruck des "Save" Buttons. Der Benutzer wird über den Fortschritt der Speicher-Aktion visuell informiert.

Darüber hinaus werden im Dashboard (s.u.) Möglichkeiten der Benutzer gegeben, einen QR-Code für eine bestimmte Instanz des Datenmodells zu erstellen oder eine Übersicht der modellierten Gegenstände zu erhalten.

Gemeinsame Grund-Funktionalitäten der Anwendungen

Die entwickelten Anwendungen teilen sich zwei visuelle Darstellungen: die `DashboardActivityBase` und die `SplashActivityBase`. Das Dashboard stellt ein Gitter dar, welches für die Auswahl einer bestimmten auszuführenden Aktivität zuständig ist. Des Weiteren ermöglicht das Dashboard die Darstellung einer Anzahl von "ungelesenen Ereignissen", die mit der jeweiligen dargestellten, auswählbaren Aktivität verknüpft sind. Beispielsweise kann durch diese visuelle Präsentation visuelles Feedback über die Anzahl von ungelesenen Mitteilungen aus dem Kommunikationskanal gegeben werden.

Darüber hinaus stellt die "Splash"-Aktivität ein Ladefenster dar, welches über den Fortschritt der "Preload"-Vorgänge (siehe dazu Abschnitt 7.4.1) informiert und den Benutzer bei der Benutzung der Anwendung visuell "begrüßt" (z.B. wird die Mitteilung "HomeSharingApp – Guest" dargestellt). Nachdem die "Preload"-Vorgänge erfolgreich abgeschlossen wurden, gelangt der Benutzer in die Ausführung der oben genannten `DashboardActivityBase` (bzw. ihre Spezialisierung, siehe unten).

Die genannten Aktivitäten stellen nur eine Vorlage dar, die durch die spezialisierten Klassen namens `DashboardActivityGuest` bzw. `DashboardActivityHost` erweitert wird. Bezie-

hungsweise werden in den Sub-Klassen die vorher genannten Aktivitäten des Wohnungseigentümers und die Aktivitäten des Wohnungsmieters zur Wahl in Form eines Dashboards vorgestellt.

Wohnungsmieter-Anwendung

Die Wohnungsmieter-Anwendung stellt dem End-Benutzer folgende zusammengefasste Android-Aktivitäten (und dementsprechende Funktionalitäten) bereit:

- `QRCodeScannerWrapperActivity`: Eine Aktivität zum Einscannen von QR-Codes wird dem Benutzer vorgestellt. Diese Aktivität wird im Vollbildmodus (engl. full screen mode) vorgestellt und verwendet die Bildaufnahme-Fähigkeiten des End-Geräts, indem die aktuell von der Kamera aufgenommene Szene dargestellt wird. Zugleich wird ein Hinweis zu der Funktionalität gegeben: der Benutzer kann diese Aktivität verwenden, um einen in der realen Welt (bzw. in der Umgebung, in der Wohnung) vorhandenen QR-Code einzuscannen. Weitere Details werden im Rahmen des zum Thema "Indoor-Lokalisierung" dedizierten Paragraphs gegeben.
- `RoomNavigateActivity`: Im Fall dieser Aktivität wird die im Rahmen des Paragraphs 7.4.1 eingeführte Komponente zur Navigierung einer hierarchischen Datenstruktur wiederverwendet. Die in diesem Fall angebotene Funktionalität ist eine grundlegende Navigation durch die Wohnung mithilfe der erfassten räumlichen Beziehungen. Dadurch lassen sich die modellierten bzw. erfassten Beziehungen navigieren. Falls Instanzen des Datenmodells (z.B. Gegenstände der Wohnung) in einem ausgewählten Raum vorhanden sind, werden diese in Form einer vereinfachten Auflistung vorgestellt. Die dafür zuständige Aktivität ist `AssetViewActivity`. Im Fall jeder Instanz werden die erfassten Daten (z.B. Namen) sowie die dazugehörigen Metadaten (z.B. Regeln) dargestellt.

Nachdem ein Überblick über die entwickelten Aktivitäten je nach Anwendung gegeben wurde, werden weiterhin erweiterte Funktionalitäten der beiden Anwendungen vorgestellt.

7.4.3 Sonstige Funktionalität des Frontends

In diesem Abschnitt werden die sonstigen von der Frontend-Komponente angebotenen Funktionalitäten zusammengefasst.

Indoor-Lokalisierung

Die bereits genannte Funktionalität zum Einscannen von erstellten QR-Codes (vgl. Paragraph 7.3.6) basiert auf einer open-source Bibliothek namens ZBar [32], die für das Einscannen verschiedener Typen von Codes eingesetzt wird. Bei der Ausführung der Aktivität werden Fähigkeiten der auf dem Endgerät vorhandenen Kamera genutzt, wie z.B. die Autofokus Funktion für einen schnelleren Vorgang zum Einscannen.

Nachdem der QR-Code ausgelesen und als gültig interpretiert wurde (die eingebettete Information sollte nur anhand der spezifischen im Rahmen dieser Arbeit entwickelten Annahmen wie z.B. eine

[32] `http://zbar.sourceforge.net/`

feste IP-Adresse und ein fester Port lesbar sein), gelangt der Benutzer zu der genannten Aktivität namens `AssetViewActivity`.

Die mögliche Weiterentwicklung der Funktionalität zur (automatischen) Indoor-Lokalisierung betrifft im Wesentlichen das Frontend, aufgrund der bei der Benutzung geschaffenen Mobilität. Hierfür können weiterführende Techniken zur Bestimmung einer Lokation (siehe dazu Absatz 2.5) eingesetzt werden, wie z.B. ein Wi-Fi basiertes Verfahren. Wie es schon genannt wurde, konnte hinsichtlich des geschätzten höheren technischen Aufwandes keine fortgeschrittene Technologie umgesetzt werden.

7.5 Zusammenfassung

Im Laufe dieses Kapitels wurde auf die geleistete Umsetzung der vorher eingeführten Konzepte im Detail eingegangen. Dies erfolgte im Hinblick auf die beiden Systemen (das Backend und entsprechend das Frontend). Während eine Umsetzung eines kompletten Systems zur Unterstützung des "Home Sharing" Szenarios nur teilweise geschafft wurde, zeigen die umgesetzten Anwendungen und die dazugehörige technische Infrastruktur ein gutes Potential für die Benutzung im Rahmen eines realen Szenarios.

Diese Aussage stützt sich auf erhobene Daten, die bei der durchgeführten Evaluation der genannten Systeme erfasst worden sind. Die Planung, Durchführung und Analyse der Evaluation wird als nächste Thematik in dieser Arbeit vorgestellt.

8 Evaluation

In diesem Kapitel wird die organisierte und durchgeführte Evaluation der prototypisch umgesetzten Softwarelösung vorgestellt. Im Folgenden wird die Bezeichnung “Prototyp” für beide mobilen Anwendungen (Wohnungseigentümer- bzw. Wohnungsmieter-Anwendung) benutzt.

8.1 Zielsetzung

Die Zielsetzung der Evaluation ist die möglichst präzise Beantwortung der Forschungsfrage aus Paragraph 1.4:

Wie muss ein Kontext-Modell gestaltet sein, dass es Erweiterungen und einen Umgang mit Echtzeit-Daten erlaubt, unter Beibehaltung gewisser Eigenschaften (wie die Allgemeinheit, Flexibilität oder ein erhöhtes Sicherheitsgrad) ?

Die nur konzeptionell entwickelten Konzepte wie z.B. der Umgang mit Echtzeit-Daten (wie z.B. “Datenströme”, die von lokalen oder in der Umgebung vorhandenen Sensorik erfasst werden) wurden in der Evaluation nicht berücksichtigt. Eine Vorstellung möglicher Methoden, die zu einer Lösung dieses Problems beitragen könnten findet im Paragraph 6.2.2 statt.

Auch als Echtzeit “Daten” können die Konzepte relativ zum Kommunikationskanal betrachtet werden. Diese nahmen wiederum an der Evaluation nicht teil, da der Konzept nicht für relevant genug im Rahmen der gesamten Arbeit gehalten wurde – eine Beschreibung der möglichen und teilweise auch durchgeführten Umsetzung der zum Kommunikationskanal bezogenen Konzepte erfolgte bereits im Paragraph 7.3.4.

In dieser Hinsicht wurden Aufgaben vorbereitet, die die *Allgemeinheit* und *Erweiterbarkeit / Flexibilität* des Datenmodells im Rahmen des “Home Sharing” Szenarios experimentell überprüfen können. Darüber hinaus wurden Metriken erhoben, die den Umgang der Probanden mit dem Prototyp (bzw. das dabei entstehende “Nutzungserlebnis”) bezüglich Kriterien, wie z.B. das Niveau der Frustration oder die mentale Last, ermessen können. Schließlich wurde versucht, eine allgemeine Meinung gegenüber der Nützlichkeit der Anwendung und der eventuellen Kritikpunkten zu erfragen. Die einzelnen Schritte dieses Vorgangs, von der Rekrutierung bis zu der Auswertung der Ergebnisse werden in den kommenden Abschnitten vorgestellt.

8.2 Anwerbung der Probanden

In [Tun09] wurde motiviert, dass eine Evaluation der Arbeitsbelastung (siehe auch Paragraph 8.5.3) gezielt mit einer Gruppe von Probanden durchgeführt sollte, die eine möglichst ähnliche Befähigung besitzt, wie die Nutzer, die das untersuchte System letztendlich bedienen werden. Dieser Aspekt kann auch folgendermaßen angesehen werden: bei dem Auswahlverfahren der Probanden sollen Testpersonen ausgewählt werden, die typisch für die untersuchte “Umgebung” des inspizierenden Systems sind (d.h. die benutzen ein ähnliches System schon oder haben Erfahrungen in der Sektor des Systems).

Während diese Idee sich eher an technologischen Problemstellungen richtet, lässt sie sich auch im Fall der aktuellen Evaluation benutzen: die ausgewählte Probanden sollen möglichst Interesse daran haben, die eigene Wohnung zu vermieten. Damit lässt sich die Art der Evaluation schon nennen: es

wurde ein *Feldstudie*-ähnliche Experiment durchgeführt, bei den jeweiligen Probanden vor Ort, “in situ”. Durch diese Lösungsansatz wurde versucht, die Authentizität des “Home Sharing”-Erlebnisses beizubehalten – durch diesen Ansatz lassen sich schon an dieser Stelle Fragen bezüglich des Vertrauens stellen (wie etwa “Warum würde ich eine unbekannte Person in meine Wohnung rein lassen?”). Diese Herangehensweise grenzt die Menge an möglichen Teilnehmer erheblich ein, nämlich zu dem eigentlichen Freundeskreis des Evaluationsbeauftragters. Diese Beschränkung wurde trotzdem nicht als negativ bewertet, wie bei der Zusammenfassung dieses Kapitels vorgestellt wird.

Die Anwerbung erfolgte per E-Mail und enthielt einige einführenden Merkmale über die Anwendung als Ganzes, sowie den Aspekt, dass die Evaluation *vor Ort* erfolgen muss. Darüber hinaus wurde das einzige Auswahlkriterium erläutert: es musste in der Wohnung eine WLAN-Verbindung vorhanden sein, sodass eine Verbindung zwischen Backend und Frontend möglich ist. Die WLAN-Verbindung steigerte auch die Mobilität der Probanden bei der Benutzung des Systems – somit wurde auch getestet, ob die Anwendung sich wirklich mobil in der Wohnung bedienen lässt. Schließlich wurde in der E-Mail erläutert, dass die Evaluation keine andere Technik (wie z.B. ein mobiles Gerät) erfordern würde. Es wurde zudem den Probanden empfohlen, die Zugangsdaten zu der WLAN-Verbindung möglicherweise aufzuschreiben, sodass die anfängliche Konfigurationszeit (s.u.) gering gehalten werden konnte.

Insgesamt wurden sechs Probanden gefunden und die Evaluationszeit betrug einzelne sechs Tage und dementsprechend sechs Evaluationstermine. Diese Termine fanden innerhalb des für die Ausführung der Evaluation eingeplanten und eingehaltenen Zeitintervalls von maximal 2 Wochen statt. Weiterhin erfolgt eine kurze, anonymisierte Vorstellung der Teilnehmer im kommenden Paragraph.

8.3 Profile der Teilnehmer

Die sechs Teilnehmer (ein Mann, fünf Frauen) können aufgrund des Alters in folgenden demographischen Kategorien eingestuft werden:

- **18-21 Jahre alt**: 1 Teilnehmer
- **22-25 Jahre alt**: 2 Teilnehmer
- **26-30 Jahre alt**: 1 Teilnehmer
- **31-35 Jahre alt**: 1 Teilnehmer
- **36-40 Jahre alt**: 1 Teilnehmer

Schließlich können folgende Profile der Teilnehmer hinsichtlich der jeweiligen Wohnungssituation und der ausgeübten Tätigkeit erstellt werden:

- Zwei Studentinnen, die in einer Wohnungsgemeinschaft wohnen
- Eine Studentin, die eine 2-Zi. Wohnung vermietet
- Ein erwerbstätiger Mann, der in einer 2-Zi. Wohnung wohnt
- Eine erwerbstätige Frau, die in einem Einfamilienhaus wohnt
- Eine erwerbstätige Frau, die in einem 2-Zi. Wohnung wohnt

Die aufgelisteten Aspekte spielten keine Rolle bei der Ausführung der Evaluation (d.h. kein Teilnehmer hatte Schwierigkeiten oder wurde aufgrund des Alters oder Wohnsituation benachteiligt) und wurden nur als Einblick in den Profile der Teilnehmer eingeführt. Im kommenden Paragraph werden weitere Details über die benötigten Schritte zum Aufbau der Evaluationsumgebung und die Ausführung der Evaluation gegeben.

8.4 Ablaufplan der Evaluation

Im Rahmen der Evaluation wurde ein Leitfaden-Dokument erstellt, welches zuerst die grundlegenden Ideen hinter dem Projekt und der Hintergrund dessen vorstellte. Das nächste behandelte Thema in dem Leitfaden ist die Vorstellung der technischen Infrastruktur (nämlich, dass zwei verschiedene Android Anwendungen evaluiert werden). Des Weiteren wird der Proband darauf hingewiesen, dass er/sie beide Rollen des “Home Sharing” Szenarios einnehmen wird.

Ein Ablaufplan der Evaluation ist weiterhin im Dokument vorgegeben, in dem erstens die jeweiligen Abläufe der Evaluation (wie z.B. die Einführung, die Durchführung der Aufgaben und die Beantwortung der Fragebögen) und das absolute Zeitlimit (auf höchstens 5 Minute pro Aufgabe festgestellt) beschrieben sind. Der nächste Hinweis auf dem Plan war die Auflistung verschiedener Anmerkungen, die zu beachten sind. Diese Anmerkungen wurden aus der “10 steps for the course of action” Dokument [Tog92] abgeleitet. Sie stellen allgemein im Laufe des Experiments geltenden Prinzipien vor, wie z.B. dass der Proband zu jeder Zeit mit dem Experiment aufhören darf oder dass alle bemerkte Fehler im System vom System selbst und nicht kommen vom Probanden selbst. Darüber hinaus wurde explizit erläutert, dass eine direkte Steuerung oder Beeinflussung der Aufgaben vom Evaluationsbeauftragter nicht möglich ist; jedoch wurde weiter erklärt, dass Fragen gerne beantwortet werden können.

Der Ablaufplan wurde dadurch beendet, dass alle Unklarheiten bevor der Ausführung der Aufgaben behoben werden sollen. Als nächstes Teil des Dokumentes wurden die Aufgaben vorgestellt, die zu lösen waren. Eine Auflistung der Aufgaben erfolgt im Anhang (A).

Die gestellten Aufgaben wurden so ausgewählt, dass sie die Funktionalitäten des Prototyps prüfen, mit der genannten Zielsetzung, Metriken und Freitext-Angaben zu erfassen, die zur eventuellen Beantwortung der Forschungsfrage dienen können. In [Tun09] wurde motiviert, dass im Fall von einem allgemeinen System, das im Rahmen einer Evaluation getestet werden soll, muss die Auswahl der Aufgaben anhand denjenigen Verfahren im System, die *häufig*, *kritisch* und *wirklich relevant* (engl. “real”) sind, erfolgen.

8.5 Ausführung der Evaluation

Der erste Schritt in der tatsächlichen Ausführung eines Evaluationstermins ist die ursprüngliche Einrichtung des Prototyps in der Wohnung. Dieser Schritt wird in dem nächsten Absatz näher eingegangen.

8.5.1 Einrichtung des Prototyps in der Wohnung

Einige Schritte wie z.B. die Initialisierung, Migration und Befüllung der Datenbank (siehe Paragraph 7.3.1) vor der Durchführung des jeweiligen Evaluationstermins erledigt worden. Damit wurde die ursprüngliche Konfiguration evaluiert, die normalerweise von dem Wohnungseigentümer während der Vorbereitungsphase (siehe Paragraph 3.1.1) ausgeführt werden sollte.

Die Evaluationsumgebung bestand in erster Linie aus der Wohnung, in der die Evaluation durchgeführt wurde und dessen spezifischen Bestandteile. Ein technisches Merkmal ist die Handhabung eines WLAN-Routers, das die minimale technische Infrastruktur für die Datenübertragung zwischen Frontend und Backend bereitstellt. Die erforderliche technische Apparatur bestand aus einem mobilen Gerät (ein HTC Desire, worauf Android in der Version 2.3.7 lief) und einem Server-System (in diesem Fall, ein Notebook). Beide Geräte unterstützen WLAN-Verbindungen.

Der erste Schritt in der Konfiguration des Prototyps und entsprechend der Laufzeitumgebung war die Erstellung einer WLAN-Verbindung. Nachdem die Verbindungen auf beiden Systemen erfolgreich etabliert wurden, wurde der benötigte Konfigurationsvorgang (siehe auch 7.3.3) per Kommandozeile vom Evaluationsbeauftragter ausgeführt. Damit wurde gesichert, dass die Referenz im Backend und Frontend auf dieselbe IP-Adresse des Server-Systems gesetzt wurde.

Während diese Einrichtung stattgefunden hatte, wurde der Proband darum gebeten, das mitgebrachte Dokument zum Evaluationsvorgang (bzw. der genannte Ablaufsplan) sorgfältig zu lesen und ggf. Fragen zu stellen, bevor die Bearbeitung der Aufgaben anfangen sollte.

8.5.2 Bearbeitung der Aufgaben der Wohnungseigentümer-Anwendung

Die Ausführung der Aufgaben beginnt wenn die Anwendung von dem “App Drawer” geöffnet ist. Es wird dabei ggf. die WLAN-Option angeschaltet und die Anwendung versucht, sich mit dem lokalen Netzwerk zu verbinden. Von diesem Punkt an sind die Aktionen der Probanden angefragt, die zu der eventuellen Lösung der gestellten Aufgaben führen können.

Nachdem die Aufgaben bezüglich der Wohnungseigentümer-Anwendung durchgeführt worden sind, mussten die im Paragraph 8.5.1 genannten Schritte zur Initialisierung erneut durchgeführt werden, dieses Mal für die Initialisierung der Wohnungsmieter-Anwendung. Die im vorherigen Abschnitt genannte Prozedur zur Öffnung der diesmal Wohnungsmieter-Anwendung werden wiederholt.

Schließlich werden diejenige für die Mieter gestellten Aufgaben durchhgeführt und somit fand die Ausführung der Aufgaben auf dem mobilen Gerät Ende. Die nächste vom Proband durchzuführende Aktion war die Ausfüllung der Fragebögen, welche im nächsten Paragraph behandelt wird.

8.5.3 Erfassung der Metriken

Für die Erfassung von Metriken bezüglich verschiedener mentalen Faktoren wie z.B. Frustration oder geistige Anforderung wurde das Fragebogen namens NASA-TLX [HS88] benutzt. Anhand dieses Testes kann festgestellt werden, aus welcher Parameter sich die selbst ermittelte Beurteilung der Arbeitsbelastung zusammensetzt.

Zu diesem Zweck werden die folgenden Parameter oder Faktoren der Arbeitsbelastung von dem dadurch multidimensionalen Fragebogen berücksichtigt:

- Geistige Anforderung (engl. Mental Demand)
- Körperliche Anforderung (engl. Physical Demand)
- Zeitliche Anforderung (engl. Temporal Demand)
- Aufgabenerfüllung (engl. Performance)
- Anstrengung (engl. Effort)
- Frustration (engl. Frustration)

Während die ersten drei Parameter die Anforderungen an dem Proband messen, beziehen sich die letzten drei Parameter auf die Auseinandersetzung der Probanden mit der gelösten Aufgabe. In dieser Hinsicht funktioniert dieses Fragebogen nur dann, wenn der Proband im Vorfeld eine festgelegte Aufgabe erfüllt hatte.

Die Bewertung des Testes erfolgt anhand einer Skala, die mit 20 Stufen versehen ist, jeweils mit einer "natürlichen" Ordnung der Stufen (0 wird als niedrigstes Ergebnis bewertet). Darüber hinaus werden die 6 genannten Parameter oder Faktoren mit 15 paarweisen Vergleiche gewichtet; die Vergleiche werden auf einer Skala von 0 bis 5 beurteilt. Schließlich lassen sich die Seiten bezüglich des NASA-TLX Fragebogens im Anhang (A) einzeln durchgehen, als Teil des Leitfadens.

Weiterhin wurde den Probanden eine Reihe von Fragen gestellt, die eine Antwort in einen textuellen Form erlangten. Die Fragen berührten die Thematik der Allgemeinheit bzw. Vollständigkeit des Datenmodells, verfolgt von einer Frage bezüglich der Erweiterbarkeit des Modells (im Sinne des behandelten Szenarios, z.B. die modellierten Gegenstände zu einem anderen Zweck als "Home Sharing" zu benutzen). Schließlich konnten Meinungen relativ zu dem allgemeinen Konzept, dessen Nützlichkeit und Anwendbarkeit, zusammen mit eventuellen Verbesserungsvorschläge geäußert werden. Dieser zweite Fragebogen gehört ebenso zum Anhang (A), als Teil des Leitfadens.

Nachdem die Organisation und Durchführung der Evaluation vorgestellt wurde, werden die dadurch feststellbare Ergebnisse im nächsten Abschnitt erläutert.

8.6 Analyse der Ergebnisse

Die erste Auswertung bezieht sich auf den Angaben der Probanden im Fall des NASA-TLX Fragebogens und wird in Abbildung 8.1 in tabellarischen Form zusammengefasst.

Die Gesamtergebnisse je nach Proband sind in der Abbildung 8.1 als jeweils ein Balken zu sehen. Das gesamte Ergebnis eines Probanden stellt sich nach der Gewichtung der eingegebenen Maßzahlen je nach Parameter oder Faktor (z.B. wird bei dem Vorgang der Gewichtung 3 Mal den Parameter "Frustration" ausgewählt, ergibt sich eine Gewichtung von 0.2 des Parameters, weil insgesamt 15 paarweise Vergleiche durchgeführt werden) fest. Damit setzt sich das gesamte Ergebnis des NASA-TLX Fragebogens durch die Addition der gewichtete Werte des jeweiligen Parameters.

Eine Interpretation der Ergebnisse kann durch die folgenden Aussagen zusammengefasst werden:

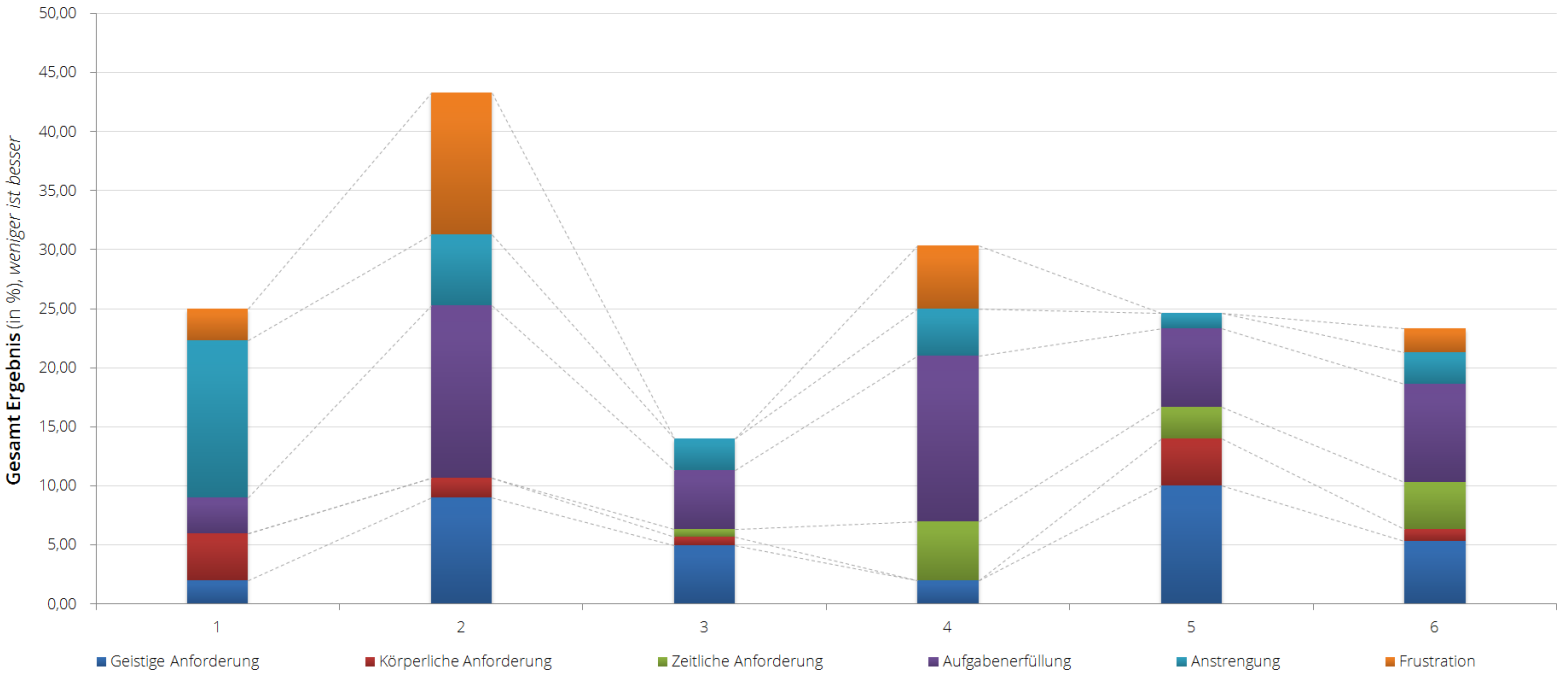


Abbildung 8.1: Auswertung des NASA-TLX Fragebogens

- Alle Probanden haben die Arbeitsbelastung jeweils als unter dem Wert von 50% eingeschätzt. Das heißt die Benutzung der Anwendung stellt keine erheblichen Schwierigkeiten.
- Trotz einem Ausreißer ist ein geringeres Frustationsniveau zu bemerken (ein eventuelles Zeichen der Bezeichnung “Calm Technology”, siehe auch 5.4.2), der Mittelwert der gewichteten Eingaben liegt beim Wert 4.40
- Die Zielsetzung der Probanden, alle Aufgaben zu lösen war als größtes Einflussfaktor von Probanden angesehen (aus alle Ergebnisse, repräsentiert dieser Faktor 66% der maximalen Ergebnisse), das Mittelwert der gewichteten Eingaben betrug 9.00
- Die Lösung der Aufgaben (Mittelwert der gewichteten Eingaben: 9.00) ist deshalb proportionell mit den geistigen Anforderungen (Mittelwert: 4.67) und die Anstrengung (Mittelwert: 5.73)
- Die zeitlichen und körperlichen Anforderungen wurden als die geringsten Einflussfaktoren eingesehen: der Mittelwert der zeitlichen Anforderungen lag bei 1.93, während die körperlichen Anforderungen mit dem Mittelwert 1.47

Des Weiteren wird die Analyse des zweitens Fragebogens vorgestellt. In dem zweiten Fragebogen wurden allgemeinen Eingaben in Freitext-Form von den Probanden verlangt. Die Ergebnisse zu den einzelnen Fragen (siehe dazu A) werden in der folgenden Auflistung zusammengefasst:

- Das allgemein vorgestellte Konzept wurde als hilfreich (teilweise auch *sehr* hilfreich) empfunden, bietend eine gut brauchbare Funktionalität (siehe unten). Außerdem wurde von einem Proband erwähnt, dass die Funktionalität auch im geschäftlichen Bereich angeobten werden könnte.

- Das vorgestellte Funktionalität des Prototyps könnte der Meinungen der Probanden nach folgendermaßen verbessert werden: durch die Bereitstellung eines Hausplans zur Raumeinteilung sowie durch die Möglichkeit, Raumdimensionen erfassen zu können.
- Darüber hinaus erwünschten sich 33% der Probanden, erweiterte Daten über die gesamte Wohnung erfassen zu können. Konkret wurde die Möglichkeit genannt, vor der Anreise zu wissen ob Haustiere in der Wohnung erlaubt sind. Weiter sind Hinweise zur Pflanzenpflege genannt worden; insbesondere im Fall von Miethäuser wurden Hinweise zur Flurreinigung erwünscht. Schließlich benannte ein Proband den Wunsch, dass Informationen zu Nachbarn und Bewohnern auch hilfreich sein könnten – mit der Erstellung eines minimalen sozialen Profils können z.B. Informationen zur Lärmempfindlichkeit erfasst werden. Weiterhin im Fall einer Wohngemeinschaft wurde erwähnt, dass die Erfassung der einzelnen zu den Mitbewohner gehörenden Zimmer als Räumlichkeiten mit diesem Konzept des Profils auch gut zusammenarbeiten könnte.
- Eine von 33% der Probanden eingegangene Thematik bezieht sich auf der eventuellen Erscheinung von Schaden in der Wohnung. Notfälle wie eine defekte Heizung oder allgemeine Fragen rund um die Versicherung sollen erfasst werden können. Darüber hinaus schlägt ein Proband vor, dass die Anwendung von einer Beibehaltung des Kontakts mit dem Wohnungseigentümer profitieren könnte. Diese Funktionalität wurde im Laufe der Evaluation nicht genannt; das vorgestellte Konzept zum Kommunikationskanal würde aber diese Anforderung erfüllen (siehe auch 1.5.4)
- Die Erweiterbarkeit des Prototyps wurde auch gut empfunden: 83% der Probanden wünschten sich eine Erweiterung der vorhandenen Kategorien, mit Beispiele wie z.B. ein Spiegel, einen Stecker (zusammen mit Anwendungshinweise bezüglich der maximal erlaubten Kraft beim Ausstecken eines Geräts) oder eine Kategorie für Dekorationsgegenstände (Vasen, Bilder, Blumen, Kerzen und Kissen), die zugleich mit Hinweise zur Umdekorierung (ob gestattet oder nicht) bereichert werden können. Einzeln benannten Geräte waren beispielsweise ein Eierkocher, eine Dunstabzugshaube oder ein Waffeleisen.
- Bezüglich der Frage zur Allgemeinheit der erfassten Daten wurden weiterhin interessante Aussagen von Probanden getroffen: es wurde die Möglichkeit einer strikten, privaten Nutzung erläutert, in dem man nur für sich selbst die Gegenstände modelliert, um die entweder bei Bedarf wiederfinden zu können oder diese einfach verwalten zu können. Des Weiteren können die erfassten Daten beim Verkauf der Wohnung helfen – eine automatische Ansicht der Wohnung konnte benutzt werden, um potentiellen Käufern ein genaueres Bild zu machen. Diese Idee kann mit der automatischen Generierung einer Anzeige für Vermietung oder Verkauft weiter verknüpft werden.

Weitere Anmerkungen dienten zur Evaluation der Benutzbarkeit des Prototyps: 33% der Probanden hätten sich eine Hilfsfunktion oder eine Gebrauchseinleitung (ein Tutorial) beim ersten Nutzen der Anwendungen gewünscht. Darüber hinaus sollte die Anwendung von einer zu jeder Zeit aufrufbarer Hilfsfunktion verfügen und in mehrerer Sprachen (z.B. Deutsch) übersetzt bzw. “lokalisiert” werden.

Schließlich wurde eine Aussage getroffen bezüglich der visuellen Artefakten zur Steuerung der Anwendung: um eine Kategorie auszuwählen (siehe auch 7.4) könnte ein Gitter (engl. Grid) eingesetzt werden; darüber hinaus stellte sich ein Proband eine Suchfunktion auch als hilfreich vor. Ein weiterer genannten Aspekt berührt die Problematik der Datensicherheit (siehe auch 1.5.5) der erfassten Daten: dieser Aspekt wurde als von großer Wichtigkeit empfunden, aufgrund der erwünschten Privatsphäre. Zudem wurde der Aspekt eines entfernten Zugriffs auf die Daten der Wohnung genannt, ein Punkt welcher in Abwägung zu der Aspekten der Privatsphäre des Mieters gebracht werden sollte.

8.7 Zusammenfassung

Am Ende der Analyse könnten die folgenden Aussagen hinsichtlich der aufgestellten Forschungsfrage getroffen werden:

Eigenschaft des Kontext-Modells	Unterstützung durch die prototypische Umsetzung
Erweiterbarkeit / Flexibilität	Das Kontext-Modell kann mit eigenen Kategorien und Gegenstände erweitert werden. Die Evaluation zeigte, dass die Erfassung mehrerer Metadaten, die in dem ursprünglichen Konzept nicht miteinbezogen sind, auch erwünscht ist. Eine weitere Entwicklung des Modells könnte in dieser Hinsicht die Funktionalität enthalten, weitere Metadaten zu erfassen.
Allgemeinheit	Das Kontext-Modell ist spezifisch für das "Home Sharing" Szenario aufgelegt. Die Evaluation zeigte, dass die damit erfassten Datenbestände auch für weitere Szenarien rund um die Wohnung angewendet werden können.
Erhöhtes Sicherheitsgrad	Das Kontext-Modell verfügt über Zugriffs- und Rechtekontrolle, die jeweils im Laufe der Evaluation implizit von den Probanden benutzt worden sind. Eine Frage bezüglich der von der Softwarelösung geleisteten Datensicherheit kann auch als Grundlage für den Bedarf einer umfassenden Sicherheitspolitik verwendet werden und zeigt zudem die Interesse der Benutzer an dieser Thematik.
Echtzeit-Fähigkeit	Das Kontext-Modell wurde nicht explizit anhand seiner Performanz getestet. Trotzdem zeigen Kennzahlen aus der Umsetzung eine gute Performanz (innerhalb von 3 Sekunden) des Web Servers bei der Behandlung einer ersten Anfrage ("cache miss") und weiterhin erheblich bessere Performanz (unter 1 Sekunde) im Fall der erneuten Behandlung derselben Anfrage ("cache hit"). Weiterhin kann das AMQP-Netzwerkprotokoll auch als "echtzeitfähig" angesehen werden, wie in Kapitel 5.2.3 motiviert.

Tabelle 8.1: Übersicht zur geleisteten Überprüfung der Forschungsfrage, hinsichtlich der Bestandteile und die Realisierbarkeit dessen

Zusammenfassend können die prototypisch entwickelten Anwendungen als für das vorgestellte Szenario des "Home Sharing" (und nicht nur) geeignet angesehen. Trotz einer nicht allzu hohe Anzahl an Probanden lässt sich das bekannte Diktum aus der Forschungsfeld namens "Usability Testing" einführen, welches besagt dass "*Observing 4-5 participants will uncover approximately 80% of the problems in a user interface that have a high likelihood of detection*" [MWT+02].

Diese Herangehensweise der kleinen Evaluationen wurde auch von Jakob Nielsen seit Jahren empfohlen[1] und behält seine Wahrheit immer noch behält[2]. Durch eine Evaluation der Gebrauchstauglichkeit (engl. usability) lassen sich auch bestimmte Merkmale über die Nützlichkeit der vorgestellten Konzepte – ein Ergebnis ist die Verbesserung *des Produktes* im Ganzen, trotz der Mangel an statistischer Aussagekraft.

Diese Hypothese hat sich im Fall der durchgeführten Evaluation verwirklicht, wo eine sehr nützliche Sammlung von Daten betreffend Verbesserungsvorschläge, Ideen zur Weiterentwicklung und Aussagen über die Arbeitsbelastung erfasst werden konnte.

Als letzter Schritt der Arbeit wird im kommenden Kapitel eine Zusammenfassung der Arbeit sowie einen Ausblick über möglichen weitere Entwicklungen gegeben.

[1] Artikel “Why You Only Need to Test with 5 Users”, Jakob Nielsen, Erscheinungsdatum: 19. März 2000, `http://www.useit.com/alertbox/20000319.html`

[2] Artikel “How Many Test Users in a Usability Study?”, Jakob Nielsen, Erscheinungsdatum: 04. Juni 2012, `http://www.useit.com/alertbox/number-of-test-users.html`

9 Zusammenfassung

Zusammenfassend wurde in dieser Arbeit zuerst ein Konzept vorgestellt, welches als Erweiterung des “Home Sharing” Ergebnisses vorgesehen ist. Diese vereinfachte Perspektive der Arbeit wurde zudem ergänzt durch die Definition einer Forschungsfrage, die die weitere Entwicklung der Arbeit ausprägte.

Anhand der abgeleiteten Anforderungen und mit Rücksicht auf vorher geleistete Arbeiten in den umgebenden Forschungsfeldern wurde eine technische Architektur definiert, welche als Zielsetzung die Unterstützung der genannten Konzepte hatte. Ein großer Teil der technischen (und zugleich “konzeptuellen”) Architektur wurde von der Definition eines Kontext-Modells anhand spezifischer Vorarbeiten und domänenspezifischer Anforderungen repräsentiert.

Das allgemein geltende Modell (ein “Meta-Modell” entsprechend der MDA- bzw. MOF-Hierarchie) wurde spezialisiert in ein Kontext-Modell, das zielgerichtet für das “Home Sharing” Szenario eingesetzt werden sollte. Weiterhin diente dieses Modell (ein Datenmodell) als Hauptkomponente zu einer prototypischen Umsetzung des gesamten Konzeptes. Nachdem ein zufriedenstellender Reifegrad des Prototyps erhalten ist, wurde dieser Prototyp anhand der Forschungsfrage und bestimmten Metriken in einer kleinen Gruppe evaluiert.

Die Evaluation zeigte das Potential des Produktes, auf Grund positiver Eindrücke der Probanden, was ihr Ansehen der Nützlichkeit und Bedienbarkeit der prototypischen Anwendungen anging. In diesem Zusammenhang werden im Folgenden weitere offene Fragen und Perspektiven bezüglich der aktuellsten Forschungsergebnisse aus den mit dieser Arbeit verwandten Forschungsfeldern diskutiert.

9.1 Ausblick

Eine im Jahr 1996 aufgestellte Vision von Mark Weiser, der “Vater” des markanten “Ubiquitous Computing” Forschungsfeldes beschrieb, wie die vielbelobten “Thin Clients’ (leichtgewichtige Geräte, die einen Zugang zum Internet besitzen) einen erheblich geringeren Effekt auf die Mensch-Maschinen-Interaktionen haben werden als die so genannten “Thin Servers” [WB96]. Die “Thin Servers” sollen dieselbe Preispolitik wie “Thin Clients” haben (zur damaligen Zeit, einige hunderte Dollar) aber sollen eine Vielfalt von erweiterten Funktionalitäten anbieten. Dadurch würde ein Web Server in jedem Haushaltsgerät oder Büroartikel eingebettet sein. Eine starke Aussage wird am Ende des Abschnittes über “Thin Clients” und “Thin Servers” getroffen: das Netzwerkprotokoll der nächsten Generation (nämlich IPv6) ist in der Lage, mehr als ein tausend Geräte für jedes Atom auf der Erdoberfläche einer IP-Adresse zuzuordnen. Wir werden alle diese Adressen brauchen.

Wir befinden uns gerade an der Konvergenz zwischen drahtloser Kommunikation und eingebetteten Systemen – dies hat zur Folge, dass elektronische Geräte zunehmend miteinander verbunden werden [DRAPZ04]. Die Verbindungen werden jede Facette des Alltags umfassen – Büros, Wohnungen und öffentlicher Raum werden in der Lage sein, umfangreiche Informationen und Dienste wie eine instante Internet-Verbindung bis auf Steuerung-Möglichkeiten ihrer Bewohner anzubieten. Das Angebot wird zudem in einer kontextbewussten, personalisierten Art und Wise erfolgen.

Die genannte Konvergenz zeigt sich auch in verwandten Forschungsfeldern: eine kürzlich erschienene Arbeit verbindet die Ideen von “Smart Objects” zu der Vision von “Internet of Things” (“Using Smart Objects to build the Internet of Things” [LRHM12]).

Zusammenfassend lässt sich sagen, dass die Ereignisse von Forschungsfeldern, die die eigene Vision mit einer anderen verwandten Vision verknüpfen, an dieser Stelle nicht aufhören werden. Der Zusammenhang zwischen “Home Sharing” und “The Internet of Things” ist schon sichtbar; vielleicht wird “Home Sharing” der Katalysator für die kommerzielle Akzeptanz für Felder wie “Internet of Things” oder sogar “Smart Home”. Wir befinden uns aber nur am Anfang dieser Integration zwischen Domänen, in einer ursprünglichen “Phase der Evolution”. Die Architektur und zugleich die Vision namens “Internet of Things” kann mithilfe des folgenden Bildes 9.1 veranschaulicht werden:

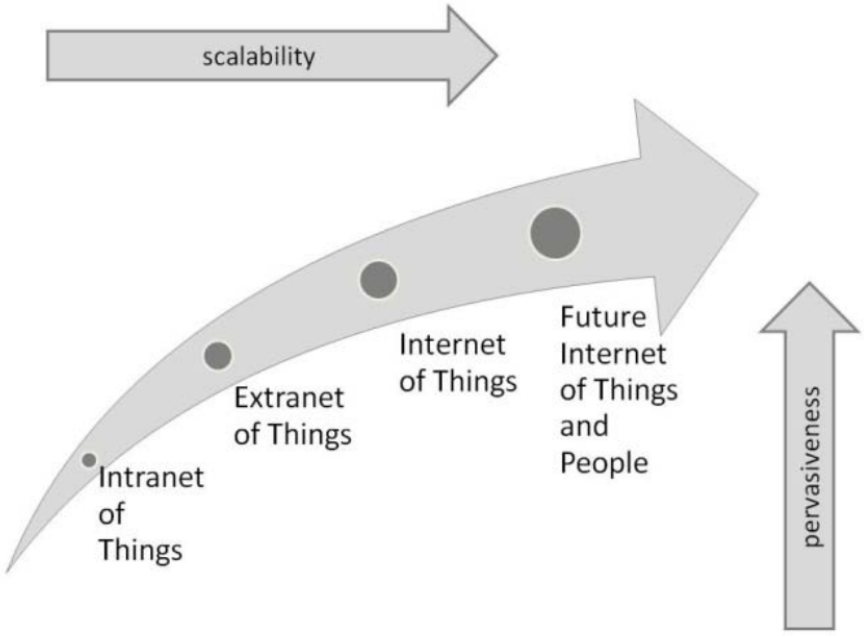


Abbildung 9.1: Überblick der Architektur von “The Internet of Things” (IoT). Quelle: [Dou12]

Mit einem fast exponentiellen Wachstum[1] in den letzten Jahren und mit wachsender Akzeptanz des Publikums, bleibt Marktführer Airbnb an der Spitze der “Home Sharing” Domäne. Die Frage, die sich noch stellt ist folgende: wohin sollte es weitergehen? Was für Evolutionspläne stellen sich die Gründer vor? Es ist alles nur eine Frage der Zeit, bis eine neue Konvergenz entsteht.

9.2 Future Work

Hinsichtlich eventueller zusätzlicher Forschungsvorhaben bezüglich dieser Arbeit lassen sich folgende Punkte in einer zusammengefassten Art und Weise nennen:

- Die Verknüpfung der *Situation* in den einzelnen Kontext-Informationen stellt sich als die größte Herausforderung in die vorstellbare Evolution des definierten Kontext-Modells dar. Aktuelle Herangehensweisen [EKLW11] benutzen physische Sensoren wie ein Ultrabreitbrand-unterstütztes

[1] Artikel ‘Airbnb Announces 10 Million Guest Nights Booked”, Erscheinungsdatum: 19, Juni 2012, `http://mashable.com/2012/06/19/airbnb-users/`

System für die Lokalisierung, verknüpft mit zeitlichen Angaben (Tag der Woche, aktuelle Zeit) und aus dem Web abgeleiteten Informationen, um eine Einschätzung der aktuellen Situation des Benutzers zu formulieren. Diese Herangehensweise der "Sensor Fusion" und Konzepte wie "Linked Open Data" eröffnet ein breites Spektrum an Möglichkeiten.

- Der Kommunikationskanal könnte von Peer-To-Peer Techniken wie AllJoyn[2] profitieren, die eine direkte Verbindung zwischen Clients, ohne die Handhabung einer zentralen Kontroll-Instanz ermöglichen. Dies kann als eine gesamte Generation von neuen Technologien angesehen werden, die diese Form der ad-hoc Kommunikation verwendet.
- Darüber hinaus werden Neuigkeiten erwartet über die von Google entwickelte Plattform für "Android in der Wohnung", namens "Android@Home"[3], die auch als der erste Schritt von Google in der Domäne "Internet of Things" angesehen wird. Eventuelle weitere Versionen dieser Arbeit können von dieser Entwicklung profitieren.

Anhand der aufgelisteten Merkmale, die für eventuelle künftige Arbeiten von Bedeutung sein können, wird ein möglicher Weg dieser Arbeit durch die technologischen Neuheiten beschrieben. Zum Abschluss lässt sich resümieren, dass die Weiterarbeit an den Konzepten die vielversprechende Thematik der "Home Sharing" Domäne weiterentwickeln könnte.

[2] https://developer.qualcomm.com/mobile-development/mobile-technologies/peer-peer-alljoyn

[3] Artikel "Google's Platform Extends Its Reach With Android@Home", Erscheinungsdatum: 05. November 2011, http://www.wired.com/gadgetlab/2011/05/android-at-home-google-io/

Abkürzungen

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AAL	Ambient Assisted Living
AF	Anforderung
AGB	Allgemeine Geschäftsbedingungen
AMQP	Advanced Message Queuing Protocol
API	Application programming interface
BDD	Behaviour-Driven Development
BI	Business Intelligence
CC/PP	Composite Capabilities / Preference Profile
CERP	Cluster of European Research Projects
CML	Context Modeling Language
CRUD	Create, Read, Update and Delete
DB	Database
DDL	Data Definition Language
DML	Data Manipulation Language
DSL	Domain Specific Language
FM	Frequency modulation
GPS	Global Positioning System
GUI	Graphical User Interface
GUM	Guide to the Expression of Uncertainty in Measurement
HAL	Hypertext Application Language
HMAC	Hash-based message authentication code
HTTP	Hyper Text Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifikation
IDL	Interface Description Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IKT	Informations- und Kommunikations-Technologien
IM	Instant Messaging
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information technology
JSON	JavaScript Object Notation
LD	Linked Data
MAC	Message Authentication Code
MDA	Model-Driven Architecture
MIME	Multipurpose Internet Mail Extensions
MOF	Meta Object Facility
MQ	Message Queuing
MVC	Model View Controller
NASA	National Aeronautics and Space Administration
NFC	Near-Field Communication
O/RM	Object-Relational Mapping

OGC	Open Geospatial Consortium
OMG	Object Management Group
ORM	Object-Relational Mapping, seltener Object-Role Mapping
OTP	The Open Telecom Platform
PC	Personal Computer
PIM	Platform-Independent Base Model
PSM	Platform-Specific Model
QR	Quick Response Code
RDF	Resource Description Framework
REST	Representational State Transfer
pREST	Pico-REST
RFID	Radio-Frequency Identification
RSS	Rich Site Summary
SDK	Software Development Kit
SGML	Standard Generalized Markup Language
SMS	Short Message Service
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
SWT	Standard Widget Toolkit
TCP	Transmission Control Protocol
TDD	Test-Driven Development
TLS	Transport Layer Security
TLX	Task Load Index
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modeling Language
URI	Uniform resource identifier
UTC	Coordinated Universal Time
UX	User Experience
VDT	Visual Display Terminals
VO	Value Object
WADL	Web Application Description Language
WLAN	Wireless LAN
WSDL	Web Service Description Language
WSN	Wireless Sensor Network
XMI	XML Metadata Interchange
XML	Extensible Markup Language
YAML	YAML Ain’t Markup Language
YARD	Yay! A Ruby Documentation Tool

Abbildungen

A Leitfaden der Evaluation

Auf den folgenden Seiten erfolgt eine Auflistung der Dokumente, die im Laufe der Evaluation verwendet worden sind.

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Leitfaden

Willkommen zur Evaluation der im Rahmen meiner Masterarbeit entwickelten mobilen Anwendung! Im Folgenden werden einführende Merkmale erklärt.

Die grundlegenden Ideen des Projektes

Das in dieser Arbeit betrachtete Szenario ist "Home Sharing": das Teilen bzw. Vermieten der eigenen Wohnung – ein Wohnungseigentümer möchte seinen zusätzlichen Raum, seine Couch usw. vermieten. Nachdem die Wohnung gebucht ist und die Mieter angereist sind, müssen sich die Mieter erstmal an die Wohnung und ihre Besonderheiten (Umgebung und andere Aspekte) anpassen.

Dabei können Situationen eintreten, in denen spezifische Informationen fehlen, wie z.B. eine bestimmte Hausregel oder die Anwendungshinweise für ein Haushaltsgerät. Es wäre nützlich, wenn solche Informationen vorab gespeichert vorliegen würden oder noch nachträglich erfasst werden könnten. Dies soll durch mobile Anwendungen gelöst werden: eine Anwendung für den *Wohnungseigentümer* sowie eine für die *Mieter*.

Dem Wohnungseigentümer wird es ermöglicht, bestimmte Hinweise oder Richtlinien an gewissen Stellen in der Wohnung (z.B. in einem Zimmer oder in der Nähe eines bestimmten Haushaltsgerätes) zu hinterlassen. Die Erfassung dieser Informationen läuft über die mobile Anwendung für den Wohnungsinhaber.

Die dadurch erstellten Daten werden den Mietern im Form einer "personalisierten" bzw. an die vermietete Wohnung und deren Bestandteile angepassten mobilen Anwendung zur Verfügung gestellt.

Zur Evaluation

Für die Evaluation wird zuerst die *Wohnungseigentümer* Anwendung benutzt. Das Szenario entspricht dem vorher erwähnten: wir stellen uns vor, dass Ihre Wohnung vermietet wird. Die Rolle des Mieters werden Sie auch "spielen" – aber bis dahin müssen die von Ihnen als relevant betrachteten Gegenstände, Regeln und andere Aspekte beschrieben werden. Begleitend dazu werden einige Aufgaben vorgestellt, die in Kürze folgen.

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Ablaufplan

Ablauf der Evaluation

I. Einführung und allgemeine Erklärungen – fast geschafft !

II. Ausführung der Aufgaben

   a) Einweisung – Aufgaben lesen, Fragen stellen bei Unklarheiten

   b) Ausführung – Zeitgrenze liegt bei **5 Minuten pro Aufgabe**

III. Ausfüllen der Fragebögen und Diskussion

   a) Beantwortung von weiteren Fragen Ihrerseits

   b) Diskussion über Ihre Erfahrung mit der Evaluation, der Anwendung an sich, Verbesserungsvorschläge u. Ä.

Anmerkungen

- Sie können zu jeder Zeit aufhören!
- Alle eventuell auftretenden Fehler kommen **vom System** und *nicht* von Ihnen!
- Ich werde während der Aufgabe nicht eingreifen, deshalb stellen Sie mir bitte im Vorfeld so viele Fragen wie möglich, wenn Ihnen etwas unklar ist

Unklarheiten ?

Haben Sie vielleicht Fragen, die vor der Ausführung gestellt werden sollen?

Aufgaben !

Die einzelnen Aufgaben lassen sich der nächsten Seite entnehmen.

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Aufgaben

Aufgaben

Sie werden beide Rollen spielen. Zuerst versuchen Sie, Ihr Haus mit Regeln, Gegenständen und allgemeinen Informationen zu beschreiben. Danach verwenden Sie die Informationen :

Als Wohnungseigentümer

1. Beschreiben bzw. modellieren Sie einige Räume des Hauses und die räumlichen Beziehungen dazu – z.B. die Küche befindet sich neben dem Wohnzimmer, welches eine Entfernung von etwa 5m vom Schlafzimmer hat.

2. Beschreiben Sie einige Gegenstände Ihrer Wohnung, z.B. einen Tisch, welcher nicht zum Essen benutzt werden sollte, einen Rechner an dem man sich nur über ein bestimmtes Konto und das dazugehörige Passwort anmelden kann.

3. Fügen Sie zu den eventuell vorhandenen Eigenschaften eines Gegenstandes erweiterte Attributte Ihrer Wahl hinzu, wie z.B. das Attribut "Maximale Anzahl von einzugebenen Kaffeepads" mit dem Wert "2" für eine Kaffeemaschine.

4. Erweitern Sie die vorhandene Liste der Gegenstände und Kategorien mit einer zustätzlichen Kategorie oder einem zusätzlichen Gerät / Gegenstand.

5. Fügen Sie eine allgemein geltende Hausregel hinzu, z.B. "Nichtraucher Wohnung" oder "Kein Lärm nach 23 Uhr, empfindliche Nachbarschaft"

Als Wohnungsvermieter

1. Rufen Sie die Liste von verfügbaren Gegenständen in der Wohnung ab.
2. Versuchen Sie zu einem vorher modellierten Gerät/Gegenstand zu navigieren

Fazit

Vielen Dank für Ihre Mitarbeit. Als Nächstes würde ich mich freuen, wenn Sie sich noch kurz Zeit nehmen würden, zwei kleine Fragebögen zu beantworten.

Fragebogen – TLX

Sub Scale	End Points	Description
Mental Demand	Low/High	How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.) Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical Demand	Low/High	How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal Demand	Low/High	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
Performance	Good/Poor	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	Low/High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration Level	Low/High	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Fragebogen

Fragebogen – TLX

Participant Code: Date:

Mental Demand How mentally demanding was the task?

Very low Very high

Physical Demand How physically demanding was the task?

Very low Very high

Temporal Demand How hurried or rushed was the pace of the task?

Very low Very high

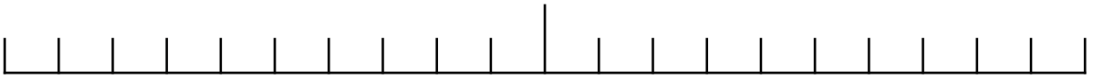
Performance How successful were you in accomplishing what you were asked to do?

Perfect Failure

Effort How hard did you have to work to accomplish your level of performance?

Very low Very high

Frustration How insecure, discouraged, irritated, stressed and annoyed were you?



Very low Very high

PLEASE TURN PAGE OVER

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Fragebogen

Fragebogen – TLX

Weights

In each pair of factors below, which of the two do you think is more important for the task that you just performed?

Place an "X" mark next to the one you think is more important than the other.

Mental Demand	☐	☐	Physical Demand
Physical Demand	☐	☐	Temporal Demand
Effort	☐	☐	Performance
Frustration Level	☐	☐	Performance
Mental Demand	☐	☐	Effort
Effort	☐	☐	Physical Demand
Frustration Level	☐	☐	Mental Demand
Mental Demand	☐	☐	Temporal Demand
Effort	☐	☐	Frustration Level
Physical Demand	☐	☐	Performance
Temporal Demand	☐	☐	Effort
Mental Demand	☐	☐	Performance
Physical Demand	☐	☐	Frustration Level
Performance	☐	☐	Temporal Demand
Temporal Demand	☐	☐	Frustration Level

Mobile **HomeSharing** Anwendung

Im Rahmen der Masterarbeit von *Cosmin Pitu*

CARL VON OSSIETZKY universität OLDENBURG

Evaluation: Fragebogen

Fragebogen – Freitext

1. Haben Sie eine Kategorie von Gegenständen oder ein bestimmtes Gerät / Möbelstück / Raumtyp etc. in dem vorgeschlagenen Verzeichnis vermisst? Bitte begründen Sie kurz Ihre Wahl.

2. Welche Informationen außer den Hausregeln, Nutzungsrichtlinien und Attributen zu den einzelnen Gegenständen fehlen, Ihrer M. nach? Würden Sie sich wünschen, etwas Anderes erfassen zu können?

3. Glauben Sie, dass Sie die erfassten Daten über Ihre Wohnung auch zu anderen Zwecken bzw. Szenarien rund um die Wohnung verwenden können?

4. Haben Sie Verbesserungsvorschläge, Ideen, allgemeine Anmerkungen zu den Konzepten, die Sie kurz kennengelernt haben? Halten Sie die angebotene Funktion für ausreichend, hilfreich ?

Literatur

[ACS09] ABLESON, Frank ; COLLINS, Charlie ; SEN, Robi: *Unlocking Android: A Developer's Guide*. Manning Publications, 2009. – ISBN 1933988673

[ALGMVM10] AVILÉS-LÓPEZ, Edgardo ; GARCÍA-MACÍAS, J.Antonio ; VILLANUEVA-MIRANDA, Ismael: Developing Ambient Intelligence Applications for the Assisted Living of the Elderly. In: *Ambient Systems (ANT)*, 2010, S. 53–60

[ATH06] ANAGNOSTOPOULOS, Christos B. ; TSOUNIS, Athanasios ; HADJIEFTHYMIADES, Stathes: Context Awareness in Mobile Computing Environments : A Survey. In: *Wireless Personal Communications* 42 (2006), November, Nr. 3, S. 445–464

[BA09] BLUMENDORF, Marco ; ALBAYRAK, Sahin: Towards a Framework for the Development of Adaptive Multimodal User Interfaces for Ambient Assisted Living Environments. In: *Universal Access in HCI* (2009), S. 150–159

[BBH+08] BETTINI, Claudio ; BRDICZKA, Oliver ; HENRICKSEN, Karen ; INDULSKA, Jadwiga ; NICKLAS, Daniela ; RANGANATHAN, Anand ; RIBONI, Daniele: *A Survey of Context Modelling and Reasoning Techniques*. 2008

[BD05] BECKER, Christian ; DÜRR, Frank: On Location Models for Ubiquitous Computing. In: *Personal and Ubiquitous Computing* 9 (2005), Nr. 1, S. 20–31

[BN04] BECKER, C. ; NICKLAS, D.: Where do spatial context-models end and where do ontologies start? A proposal of a combined approach. In: *Proceedings of the First International Workshop on Advanced Context Modeling, Reasoning and Management*, 2004 (UbiComp 2004)

[BOQ+11] BOLCHINI, Cristiana ; ORSI, Giorgio ; QUINTARELLI, Elisa ; SCHREIBER, Fabio A. ; TANCA, Letizia: Context Modelling and Context Awareness: Steps forward in the Context-ADDICT project. 2011. – Forschungsbericht. – 1–8 S.

[BPP07] BRATSKAS, Pyrros ; PASPALLIS, Nearchos ; PAPADOPOULOS, George A.: An Evaluation of the State of the Art in Context-aware Architectures. (2007)

[BR10] BOTSMAN, Rachel ; ROGERS, Roo: *What's Mine Is Yours: The Rise of Collaborative Consumption*. HarperBusiness, 2010. – ISBN 0061963542

[CFL+09] CURRAN, Kevin ; FUREY, Eoghan ; LUNNEY, Tom ; SANTOS, Jose ; WOODS, Derek: An Evaluation of Indoor Location Determination Technologies. 2009. – Forschungsbericht. – 101–162 S.

[Cur04] CURRY, Edward: Message-oriented Middleware. In: MAHMOUD, Qusay H. (Hrsg.): *Middleware for Communications*. Wiley & Sons, Ltd., 2004. – ISBN 0470862068, S. 1–28

[Cur11] CURRAN, Kevin: Ambient Intelligence - Context Aware, Pervasive and Making a Difference in a Modern World. In: CURRAN, Kevin (Hrsg.): *Ubiquitous Innovative Applications of Ambient Intelligence: Advances in Smart Systems*. IGI Global, Januar 2011 (2011). – ISBN 9781466600386, S. i–xv

[CWGN11] CIPRIANI, N. ; WIELAND, M. ; GROSSMANN, M. ; NICKLAS, D.: Tool support for the design and management of context models. In: *Information Systems* 36 (2011), März, Nr. 1, S. 99–114

[DA00] DEY, Anind K. ; ABOWD, Gregory D.: The Context Toolkit: Aiding the Development of Context-Aware Applications. (2000)

[Dav10] DAVIDOVSKI, Vlatko: *A Web-oriented Infrastructure for Interacting with Digitally Augmented Environments*, Diss., 2010

[DBS+01] DUCATEL, K. ; BOGDANOWICZ, M. ; SCAPOLO, F. ; LEIJTEN, J. ; BURGELMAN, J-C.: ISTAG - Scenarios for Ambient Intelligence in 2010 / IPTS-Seville. Seville, 2001. – Forschungsbericht

[DLY+06] DAVIDOFF, Scott ; LEE, Min K. ; YIU, Charles ; ZIMMERMAN, John ; DEY, Anind K.: Principles of Smart Home Control. In: *Work* (2006)

[Dou12] DOUKAS, Charalampos: Introduction to the Internet of Things. Version: 2012. `http://www.buildinginternetofthings.com/`. In: *Building The Internet of Things with Arduino*. 2012. – ISBN 1470023431, Kapitel 1, 1–12

[DRAPZ04] DRYTKIEWICZ, W. ; RADUSCH, I. ; ARBANOWSKI, S. ; POPESCU-ZELETIN, R.: pREST: a REST-based protocol for pervasive systems. In: *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No.04EX975)* (2004), S. 340–348

[EKLW11] ELLENBERG, Jens ; KARSTAEDT, Bastian ; LUCK, Kai V. ; WENDHOLT, Birgit: An Environment for Context-Aware Applications in Smart Homes. In: *International Conference on Indoor Positioning and Indoor Navigation*, 2011, S. 21–23

[EPR06] EUZENAT, Jérôme ; PIERSON, Jérôme ; RAMPARANY, Fano: A Context Information Manager for Pervasive Computing Environments. (2006)

[Fie00] FIELDING, Roy: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, PhD, 2000

[Gan10] GANSKY, Lisa: *The Mesh: Why the Future of Business Is Sharing*. Portfolio Hardcover, 2010. – ISBN 1591843715

[Gei11] GEISTERT, Jonas: *WLAN Positioning*. 2011

[GL08] GITSHAM, Matthew ; LENSSEN, Gilbert: Developing the Global Leader of Tomorrow. 2008. – Forschungsbericht. – ISBN 9780903542753

[HCH+11] HAUSEN, Doris ; CONRADI, Bettina ; HANG, Alina ; HENNECKE, Fabiant ; KRATZ, Sven ; LÖHMANN, Sebastian ; RICHTER, Hendrik ; BUTZ, Andreas ; HUSSMANN, Heinrich: Ubiquitous Computing - An overview of current trends, developments and research. 2011. – Forschungsbericht

[HIM05] HENRICKSEN, Karen ; INDULSKA, Jadwiga ; MCFADDEN, Ted: Modelling Context Information with ORM. 2005. – Forschungsbericht

[HP12] HARTSON, Rex ; PYLA, Pardha: *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Morgan Kaufmann, 2012. – ISBN 0123852412

[HS88] HART, Sandra ; STAVELAND, Lowell: Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: *Human Mental Workload* (1988), S. 239–250

[Ind12] INDOOR ATLAS LTD.: Ambient magnetic field-based indoor location technology. (2012), Nr. July

[KCA+11] KRANENBURG, Rob van ; CAPRIO, Dan ; ANZELMO, Erin ; DODSON, Sean ; BASSI, Alessandro ; RATTO, Matt: *The Internet of Things*. 2011

[KK06] KOVÁCS, George L. ; KOPÁCSI, Sándor: Some Aspects of Ambient Intelligence. In: *Acta Polytehnica Hungarica* 3 (2006), Nr. 1, S. 35–60

[KLN08] KAWSAR, Fahim ; LYARDET, Fernando ; NAKAJIMA, Tatsuo: Three Challenges for Future Smart Object Systems. (2008)

[KSY12] KOSBA, Ahmed E. ; SAEED, Ahmed ; YOUSSEF, Moustafa: RASID : A Robust WLAN Device-free Passive Motion Detection System. In: *2012 IEEE International Conference on Pervasive Computing and Communications*, 2012. – ISBN 9781467302586, S. 180–189

[KTP11] KAMILARIS, Andreas ; TRIFA, Vlad ; PITSILLIDES, Andreas: HomeWeb: An Application Framework for Web-based Smart Homes. (2011), S. 1–6. ISBN 9781457700248

[Lee07] LEE, Matthew: *Intro to Context-Aware Computing (Slides)*. 2007

[LG12] LANTHALER, Markus ; GÜTL, Christian: On Using JSON-LD to Create Evolvable RESTful Services. In: *WS-REST*, 2012. – ISBN 9781450311908

[LLCS10] LEE, Karen ; LUNNEY, Tom ; CURRAN, Kevin ; SANTOS, Jose: *Proactive Context-Awareness in Ambient Assisted Living*. 2010

[LRBA10] LEHMANN, Grzegorz ; RIEGER, Andreas ; BLUMENDORF, Marco ; ALBAYRAK, Sahin: A 3-Layer Architecture for Smart Environment Models. A model-based approach. (2010), S. 636–641. ISBN 9781424453283

[LRHM12] LÓPEZ, Tomás S. ; RANASINGHE, Damith C. ; HARRISON, Mark ; MCFARLANE, Duncan: *Using Smart Objects to build the Internet of Things*. Cambridge Service Alliance, 2012

[Meh12] MEHRA, Pankaj: Context-Aware Computing - Beyond Search and Location-based Services. In: *IEEE Internet Computing* (2012)

[MO11] MACEDO, Tiago ; OLIVEIRA, Fred: *Redis Cookbook*. O'Reilly Media, 2011. – ISBN 1449305040

[MPOMI10] MATIC, Aleksandar ; PAPLIATSEYEU, Andrei ; OSMANI, Venet ; MAYORA-IBARRA, Oscar: Tuning to your position: FM radio based indoor localization with spontaneous recalibration. In: *2010 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (2010), März, S. 153–161

[MWT+02] MEDLOCK, M. C. ; WIXON, D. ; TERRANO, M. ; ROMERO, R. ; FULTON, B.: Using the RITE method to improve products: A definition and a case study. In: *Proceedings of the Usability Professionals Association*, 2002

[Nie93] NIELSEN, Jakob: *Usability Engineering*. Morgan Kaufmann, 1993. – ISBN 0125184069

[Nor12] NORDAN, Robert: *A Framework for Testing Indoor Positioning Systems*. 2012

[Per10] PERROTTA, Paolo: *Metaprogramming Ruby: Program Like the Ruby Pros*. Pragmatic Bookshelf, 2010. – ISBN 1934356476

[PGS+11] PREHOFER, C. ; GURP, Jiles van ; STIRBU, Vlad ; SATHISH, Sailesh ; LIIMATAINEN, Pasi P. ; FLORA, Cristiano di ; TARKOMA, Sasu: Practical Web-based Smart Spaces. 2011. – Forschungsbericht. – 1–20 S.

[POM12] POPLETEEV, Andrei ; OSMANI, Venet ; MAYORA, Oscar: Investigation of indoor localization with ambient FM radio stations. In: *2012 IEEE International Conference on Pervasive Computing and Communications*, 2012. – ISBN 9781467302586, S. 171–179

[Pre11] PRECHELT, Lutz: *Online verfügbare Folien der Lehrveranstaltung 'Softwaretechnik', Freie Universität Berlin, SoSe 2011*. URL: `http://www.inf.fu-berlin.de/inst/ag-se/teaching/V-SWT-2011` [Letzter Zugriff: 19. Oktober 2012] , 2011. – Vorlesung

[Rai11] RAINER, Mautz: Overview of Indoor Positioning Technologies (Slides). In: *IPIN*, 2011

[RLB10] ROGERS, Rick ; LOMBARDO, John ; BLAKE, Meike: *Android Application Development*. Shroff Publishers & Distributors Pvt Ltd, 2010. – ISBN 8184047339

[RQBA09] RUNGE, Mathias ; QUADE, Michael ; BLUMENDORF, Marco ; ALBAYRAK, Sahin: Towards a Common Smart Home Technology. (2009)

[Rup09] RUPP, Chris: *Requirements-Engineering und -Management*. Hanser Fachbuchverlag, 2009. – ISBN 3446418415

[Sat02] SATYANARAYANAN, M: Challenges in implementing a Context-Aware System. In: *Pervasive Computing* (2002), S. 2

[SAW94] SCHILIT, B. ; ADAMS, N. ; WANT, R.: Context-Aware Computing Applications. In: *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA : IEEE Computer Society, 1994 (WMCSA '94). – ISBN 978–0–7695–3451–0, 85–90

[SB07] SAVIO, Nadav ; BRAITERMAN, Jared: Design Sketch: The Context of Mobile Interaction. In: *Mobile HCI* (2007), Nr. September, S. 5–7

[SBG99] SCHMIDT, Albrecht ; BEIGL, Michael ; GELLERSEN, Hans-W: There is more to context than location. In: *Computers & Graphics* 23 (1999), Dezember, Nr. 6, S. 893–901

[Sch06] SCHMIDT, Douglas C.: Model-Driven Engineering. (2006), Nr. February, S. 25–31

[Sha03] SHADBOLT, Nigel: Ambient Intelligence. In: *IEEE Intelligent Systems* (2003)

[Sib] SIBYLLE FRÖSCHLE: *Folien der Lehrveranstaltung 'Sichere Kommunikation', Universität Oldenburg, SoSe*

[SLP04] STRANG, Thomas ; LINNHOFF-POPIEN, Claudia: A Context Modeling Survey. In: *Graphical Models* (2004)

[SSbO10] SCHULZ, Lasse ; SÜSS BAUER, Elisabeth ; OTTO, Siegmar: Nutzen statt Besitzen – Perspektiven für ressourcen-effizienten Konsum durch innovative Dienstleistungen. (2010), Nr. August

[SW98] SHANNON, Claude E. ; WEAVER, Warren: *The Mathematical Theory of Communication*. University of Illinois Press, 1998. – ISBN 0252725468

[TGD+10] TRIFA, Vlad ; GUINARD, Dominique ; DAVIDOVSKI, Vlatko ; KAMILARIS, Andreas ; DELCHEV, Ivan: Web Messaging for Open and Scalable Distributed Sensing Applications. 2010. – Forschungsbericht. – 1–15 S.

[Tog92] TOGNAZZINI, Bruce: *Tog on Interface*. Addison-Wesley Professional, 1992. – ISBN 0201608421

[TS07] TANENBAUM, Andrew S. ; STEEN, Maarten van ; EDUCATION, Pearson (Hrsg.): *Distributed Systems - Principles and Paradigms*. 2nd Editio. 2007. – ISBN 0132392275

[Tun09] TUNGARE, Manas: Mental Workload in Personal Information Management : Understanding PIM Practices Across Multiple Devices. (2009)

[UHM11] UCKELMANN, Dieter ; HARRISON, Mark ; MICHAHELLES, Florian: *Architecting the Internet of Things*. Springer, 2011. – ISBN 9783642191565

[VW12] VIDELA, Alvaro ; WILLIAMS, Jason J. W.: *RabbitMQ in Action: Distributed Messaging for Everyone*. Manning Publications, 2012. – ISBN 1935182978

[Wag04] WAGELAAR, Dennis: *Towards a Context-Driven Development Framework for Ambient Intelligence*. 2004

[WB96] WEISER, Mark ; BROWN, John S.: The coming age of Calm Technology. 01 (1996), Nr. July, S. 1–17

[Wei91] WEISER, Mark: The computer for the 21st century. In: *Scientific American* 265 (1991), Nr. 3, S. 94–104

[Wei03] WEISFELD, Matt: *Object-Oriented Thought Process, The (2nd Edition)*. Sams, 2003. – ISBN 0672326116

[WHZM12] WAGNER, Stephan ; HANDTE, Marcus ; ZUNIGA, Marco ; MARRON, Pedro J.: On Optimal Tag Placement for Indoor Localization. In: *2012 IEEE International Conference on Pervasive Computing and Communications*, 2012. – ISBN 9781467302586, S. 162–170

[Win11] WINTER, Andreas: *Folien der Lehrveranstaltung 'Software Technik', Universität Oldenburg, SoSe*. 2011. – Vorlesung

[WX08] WOJCIECHOWSKI, Manfred ; XIONG, Jinhua: A User Interface Level Context Model for Ambient Assisted Living. In: HELAL, S (Hrsg.): *Smart homes and health telematics, ICOST2008*, Springer, 2008, S. 105–112

Versicherung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 19. Oktober 2012

Cosmin Pitu