RWTH AACHEN

SE SOFTWARE ENGINEERING

Ibrahim Armaç

Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit

Aachener Informatik-Berichte,
Software Engineering

Band 5

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe
Prof. Dr.-Ing. Manfred Nagl

[Arm10] I. Armaç
Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit.
Shaker Verlag, ISBN 978-3-8322-9377-2.
Aachener Informatik-Berichte,
Software Engineering Band 5. 2010.
www.se-rwth.de/publications

Personalisierte eHomes:
Mobilität, Privatsphäre und Sicherheit

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines Doktors der
Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Ibrahim Armaç

aus Pertek/Türkei

Berichter: Universitätsprofessor Dr.-Ing. Manfred Nagl
Universitätsprofessor Dr. Dr. h.c. Otto Spaniol

Tag der mündlichen Prüfung: 06.07.2010

Die Druckfassung dieser Dissertation ist unter der ISBN 978-3-8322-9377-2 erschienen.

Zusammenfassung

eHomes sind intelligente Umgebungen, die ihren Benutzern durch die Kopplung von Geräte- und Softwarefunktionalitäten einen Mehrwert in Form von integrierten Diensten bieten. Der Mehrwert besteht darin, dass den Benutzern neuartige Funktionalitäten angeboten werden, die sie durch die getrennte Verwendung der einzelnen Geräte nicht erlangen könnten. Es existiert eine breite Palette von Anwendungsfeldern für eHome-Dienste wie etwa in den Bereichen Komfort, Sicherheit, Multimedia, Verbrauchserfassung oder betreutes Wohnen.

Es gibt eine Reihe von Herausforderungen, die für die Entwicklung und den Betrieb von eHomes überwunden werden müssen. Um die Interoperabilität der Dienste zu gewährleisten, muss zunächst die aus der Vielzahl von unterschiedlichen Geräten und Standards resultierende *Heterogenität* bewältigt werden. Außerdem muss die hochgradige *Dynamik* berücksichtigt werden, um zur Betriebszeit auf Änderungen eingehen zu können, die sich etwa aus der Mobilität von Geräten und Benutzern ergeben. Weiterhin sind geeignete Konzepte zur *Personalisierung* erforderlich, um Funktionalitäten auf die individuellen Bedürfnisse der Benutzer anzupassen. Darüber hinaus muss die *wechselseitige Sicherheit* von Benutzern und eHomes gewährleistet werden.

Zur Bewältigung der Heterogenitätsherausforderung wurden im Rahmen eines Forschungsprojekts am Lehrstuhl für Software Engineering (i3) an der RWTH Aachen bereits Konzepte und Werkzeuge für die Konfigurierung und den Betrieb von eHomes entwickelt. Hierfür wurde eine *komponentenbasierte* Softwarearchitektur entworfen, auf deren Basis Dienste flexibel und unverändert in mehreren eHomes wiederverwendet werden können. Die Wiederverwendung wird dabei durch den sogenannten SCD-Prozess unterstützt, der sich aus den drei Phasen *Spezifizierung*, *Konfigurierung* und *Deployment* zusammensetzt. In diesem Prozess kann auf Basis der erfassten Kundenwünsche komfortabel eine eHome-spezifische Dienstkomposition erstellt und zur Ausführung gebracht werden. Aufgrund des *kontinuierlichen* Charakters des SCD-Prozesses kann die Dienstkomposition im laufenden Betrieb eines eHomes an Änderungen im Rahmen der erwähnten Dynamik angepasst werden.

Die vorliegende Arbeit ist im Rahmen dieses Projekts entstanden und betrachtet zwei signifikante Aspekte von eHomes, die in den Vorgängerarbeiten nicht behandelt wurden.

Die erste Fragestellung betrifft die Unterstützung der Personalisierung im Rahmen der *Inter-eHome-Mobilität*, einer speziellen Art der Dynamik. Hierunter wird der Sachverhalt verstanden, dass Benutzer häufig zwischen verschiedenen eHomes wechseln. Ziel ist es, ihre Präferenzen in allen eHomes ohne mehrfachen Konfigurierungsaufwand soweit wie möglich zu berücksichtigen. Die im Rahmen dieser Arbeit neu entwickelten Konzepte ermöglichen die Personalisierung von eHomes auf zwei Arten: Erstens wird ein mobiles Benutzermodell entwickelt, mit dessen Hilfe die Benutzer ihre persönlichen Daten eHome-übergreifend auf einem mobilen Gerät verwalten und den jeweiligen eHomes zur Verfügung stellen können. Zweitens wird ermöglicht, dass die Benutzer eigene Dienste mitnehmen und auf ihren mobilen Geräten ausführen. Dadurch können sie gewohnte Funktionalitäten auch dann nutzen, wenn das besuchte eHome die Funktionalitäten nicht direkt über einen eigenen Dienst anbietet, obwohl es über die dafür nötigen Geräte verfügt.

Die zweite Fragestellung bezieht sich auf die Gewährleistung der wechselseitigen Sicherheit im Rahmen dieser Mobilität. Es werden sowohl die Schutzziele der Benutzer als auch die der eHomes erfüllt.

Um die *Privatsphäre* der Benutzer zu schützen, werden Konzepte entwickelt, die die Einhaltung der Prinzipien *Datensparsamkeit* und *Unverkettbarkeit* persönlicher Daten ermöglichen.

Für die Realisierung der Datensparsamkeit gegenüber eHomes wird ein *aushandlungsbasiertes Identitätsmanagementsystem* eingeführt. Mithilfe dieses Systems können mobile Benutzer sowohl die Menge der einem eHome preisgegebenen Daten minimieren als auch jedem eHome gegenüber mit einer anderen Identität entgegentreten. Zur Realisierung der Datensparsamkeit gegenüber Diensten wird eine *selektive Zugriffskontrolle* für die Daten umgesetzt, die einem eHome bereits offengelegt wurden. Dadurch wird die Menge der einem Dienst zur Verfügung gestellten Daten auf ein notwendiges Minimum reduziert.

Um eine Verkettbarkeit unterschiedlicher Identitäten durch mehrere eHomes auszuschließen, wird ein *Authentifizierungsmechanismus auf Basis anonymer Credentials* entwickelt. Eine mögliche Verkettbarkeit der Daten durch mehrere Dienste in einem eHome hingegen wird durch den Einsatz von *Pseudonymen* verhindert.

Im Hinblick auf den *Schutz von eHome-Diensten vor unbefugten Zugriffen* werden zwei Arten von Zugriffen betrachtet. Für den Schutz vor dem Zugriff unbefugter Benutzer werden alternative Zugriffskontrollmechanismen entwickelt, die grob in *Credential- und rollenbasierte* Mechanismen unterteilt werden können. Sie erfüllen das Ziel der Zurechenbarkeit, das die Zuordnung von Aktivitäten zu den Benutzern ermöglicht, wenn dies erforderlich ist. Für den Schutz vor unbefugten Zugriffen durch Dienste wird eine *rollenbasierte* Zugriffskontrolle eingesetzt. Die Generierung und die dynamische Anpassung der Zugriffsrechte an eine sich zur Laufzeit ändernde Dienstkomposition findet dabei vollständig automatisch statt.

Die beschriebenen Konzepte werden in Form neuer *Werkzeuge* umgesetzt. Die Anwendbarkeit der Ergebnisse dieser Arbeit wird durch ihre Evaluierung anhand von Testszenarien gezeigt. Hierfür werden verschiedene *Demonstratoren* herangezogen, die im Rahmen dieser Arbeit entwickelt wurden.

Durch die Ergebnisse dieser Arbeit werden die einander widerstrebenden Anforderungen *Schutz der Privatsphäre* und *Personalisierung* miteinander in Einklang gebracht. Der entwickelte Lösungsansatz bietet hierfür einen guten Kompromiss zwischen Anonymität und Zurechenbarkeit. Insgesamt leistet diese Arbeit einen Beitrag zur Akzeptanz von eHomes, da mobile Benutzer durch die entwickelten Konzepte von den Vorteilen personalisierbarer Dienste in unterschiedlichen eHomes profitieren können, während ihre Privatsphäre geschützt bleibt.

Danksagung

An dieser Stelle möchte ich mich herzlich bei den Menschen bedanken, die mich während meiner Promotion begleitet und unterstützt haben.

Mein erster Dank gilt meinem Doktorvater Herrn Prof. Manfred Nagl für die Möglichkeit der Promotion an seinem Lehrstuhl. Er hat durch seine wissenschaftliche Betreuung und kontroversen Diskussionen zum Gelingen dieser Arbeit beigetragen. Darüber hinaus habe ich unsere Gespräche über verschiedene gesellschaftspolitische Themen genossen und konnte von ihm so auch viel über das Fachliche hinaus lernen.

Herrn Prof. Otto Spaniol danke ich für meine Aufnahme als Stipendiat in das DFG-Graduiertenkolleg „Software für mobile Kommunikationssysteme“ und für die Übernahme des Zweitgutachtens. Des Weiteren danke ich ihm für die Unterstützung bei der Organisation des Fußballturniers der Fachgruppe Informatik, das ohne ihn bei Weitem nicht auf so ein breites Interesse gestoßen wäre.

Herrn Prof. Bernhard Rumpe danke ich für die neuen Ideen und die Unterstützung im Hinblick auf die Entwicklung des Second-Life-Demonstrators. Ferner freut es mich, dass ich nach meiner Promotion noch etwas länger am Lehrstuhl bleiben kann.

Als Nächstes möchte ich den Mitgliedern der eHome-Gruppe danken. Die fruchtbare Zusammenarbeit bei der Implementierung und beim Verfassen von Papieren mit Michael Kirchhof, Ulrich Norbisrath und Daniel Retkowitz habe ich stets genossen. Viele Momente auf der Ruderetage werden mir immer in Erinnerung bleiben. Meinen Studenten danke ich für die tolle Zusammenarbeit und ihre sehr guten Resultate. Dies sind im Einzelnen meine Diplomanden Daniel Evers [Eve07], Olaf Gliewe [Gli06], Guido Oelmann [Oel07], Daniel Rose [Ros08], Roland Gruessner [Grü09] und Marcel Pettau [Pet09], mein Bachelorstudent Julian Meichsner [Mei10] sowie meine „HiWis“ André Egners und Markus Look.

Andriy Panchenko vom Lehrstuhl für Informatik 4 danke ich für die intensiven Diskussionen und die wichtigen Impulse aus der Perspektive eines Sicherheitsexperten. Herrn Prof. Dogan Kesdogan danke ich für seine wegweisenden Ratschläge zu Beginn meiner Promotion. Die kritische Betrachtung der Arbeit aus verschiedenen Perspektiven hat sehr zur Verbesserung des Ansatzes beigetragen.

Des Weiteren möchte ich allen Personen danken, die meine Arbeit korrekturgelesen haben: Ismet Aktas, Ebru Armac, Roland Gruessner, Thomas Heer, Julian Meichsner, Cem und Azime Mengi, Markus Look, Guido Oelmann, Marcel Pettau, Daniel Retkowitz, Florian Schmidt und René Woerzberger.

Ferner möchte ich den Kollegen des Lehrstuhls für Software Engineering für die tolle Zeit und die schönen Momente danken. Ein besonderer Dank geht an Cem Mengi und seine Ehefrau Azime, die stets sehr gute Freunde waren und mir nicht nur als Trauzeugen immer hilfreich und beruhigend zur Seite standen. Außerdem ist Cem ein super Fußballspieler, ohne ihn hätten wir den InfoCup 2008 nicht gewinnen können. Für ein Lächeln zwischendurch haben immer wieder Thomas Heer mit seinen lustigen E-Mails sowie Daniel Retkowitz mit seinem trockenen Humor gesorgt. Michael Kirchhof und Ulrich Norbisrath haben mir das eHome-Thema schmackhaft gemacht und mich zur Promotion ermuntert. René Wörzberger konnte den Lehrstuhl bei jeder Weihnachtsfeier mit lustigen Anekdoten beim Jahresrückblick

amüsieren. Dafür gebührt ihm ein besonderer Dank, auch wenn dies meist auf meine Kosten geschah ;) Bewundert habe ich Theresa Koertgen für ihren Ehrgeiz und Christoph Mosler für seinen Pragmatismus, nicht nur bei unseren gemeinsamen Laufaktivitäten. Simon Becker, Boris Böhlen, Marita Breuer, Felix Gatzemeyer, Thomas Haase, Markus Heller, Rim Jnidi, Bodo Kraft, Oliver Meyer, Ulrike Ranger, Galina Volkova, Erhard Weinell und Ingo Weisemöller danke ich für ihre vielseitigen Ratschläge und die amüsanten Diskussionen. Für die tolle Unterstützung bei allen möglichen organisatorischen Angelegenheiten danke ich unseren Sekretärinnen Angelika Fleck, Silke Cormann und Sylvia Gunder. Den neuen Kollegen aus Braunschweig Christian Berger, Arne Haber, Christoph Hermann, Shahar Maoz, Claas Pinkernell, Holger Rendel, Jan Oliver Ringert und Martin Schindler danke ich für ihre fachliche und organisatorische Unterstützung sowie für ihre Rücksicht für mein langsames Essen und für meine bescheidenen Kickerfähigkeiten. Besonders beeindruckt hat mich Claas mit seinem außerordentlichen Organisationstalent. Thomas Kurpick und Antonio Navarro Peréz danke ich für die Zusammenarbeit bei der Organisation des Tags der Informatik 2010.

Auch den Kollegen des Lehrstuhls für Informatik 4 möchte ich meinen Dank dafür aussprechen, dass ich mich wie einer von ihnen fühlen konnte. Insbesondere die vielen gemeinsamen Grillstunden auf der Terrasse und die vielseitigen Vorteile ihres Convenience Systems werden mir in Erinnerung bleiben. Ismet Aktas, Jó Bitsch, Uta Christoph, Jan Kritzner, Rainer Krogull, Tim Seipold und Dirk Thissen sind nur einige, die ich hier namentlich nennen kann.

Ich danke auch meinen Freunden außerhalb der RWTH, die während meiner Freizeit für angenehme Ablenkung gesorgt haben. Insbesondere möchte ich hier Fatma und Bahtiyar sowie Ali und Eda aus Schweinfurt hervorheben, die vor meiner Prüfung teilweise mehr gefiebert haben, als ich selbst ;)

Mein größter Dank gebührt meiner Familie. Ohne die Unterstützung meiner Eltern und Geschwister, auch in schwierigen Zeiten, hätte ich sowohl mein Studium als auch meine Promotion nicht so erfolgreich meistern können. Ich bin Stolz, Teil dieser Familie zu sein. Meinen Schwiegereltern danke ich für Ihre aufmunternden Worte und für das Daumendrücken aus der Ferne. Schließlich möchte ich mich ganz besonders bei meiner Frau Ebru bedanken. Sie hat mir in der letzten Zeit immer den notwendigen Freiraum für diese Arbeit verschafft, mich stets aufgemuntert und mir unglaublich viel Geduld entgegen gebracht. Schön, dass es Dich gibt, mein Engel.

Aachen, Juli 2010
Ibrahim Armaç

Ibrahim Armaç

Personalisierte eHomes:
Mobilität, Privatsphäre und Sicherheit

Die in dieser Arbeit erwähnten Verfahren sowie Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen.

Inhaltsverzeichnis

Abbildungsverzeichnis **XV**

1 Einführung **1**
1.1 Das eHome-Projekt 2
1.2 Motivation 4
1.2.1 Szenario 5
1.2.2 Analyse des Szenarios 6
1.3 Anforderungen 9
1.3.1 Dienstauswahl 9
1.3.2 Interaktion von Benutzern mit Diensten 9
1.3.3 Personalisierung 9
1.3.4 Schutz der Privatsphäre 10
1.3.5 Schutz von eHomes 11
1.4 Lösungsskizze 12
1.4.1 Dienstauswahl 12
1.4.2 Interaktion von Benutzern mit Diensten 13
1.4.3 Personalisierung 13
1.4.4 Schutz der Privatsphäre 14
1.4.5 Schutz von eHomes 16
1.5 Gliederung der Arbeit 18

2 Grundlagen **21**
2.1 Allgemeine Begriffe und Konzepte 21
2.1.1 Benutzermodelle 21
2.1.2 Kontextbezogenes Computing 26
2.1.3 Komponentenbasierte Softwareentwicklung 28
2.1.4 Die OSGi-Service-Plattform 29
2.2 Struktur von eHome-Systemen 34
2.2.1 eHomes 34
2.2.2 eHome-Systeme 36
2.2.3 eHome-Dienste 37
2.2.4 Personalisierbare eHomes und eHome-Dienste 39
2.3 Anwendungsbeispiele für eHome-Dienste 40
2.3.1 Lichtdienst 40

2.3.2 Musikdienst . . . 40
2.3.3 Lifesytle-Dienst . . . 41
2.3.4 Weckdienst . . . 41
2.3.5 Klingeldienst . . . 41
2.3.6 Medizindienst . . . 42
2.3.7 Sicherheitsdienst . . . 42
2.4 Digitale Identität und Identitätsmanagement . . . 43
2.4.1 Was ist eine Identität? . . . 43
2.4.2 Digitale Identitäten und Teilidentitäten . . . 45
2.4.3 Anonymität . . . 46
2.4.4 Pseudonymität . . . 48
2.4.5 Identitätsmanagement . . . 50
2.5 Datenschutz und Privatsphäre . . . 52
2.5.1 Schutzziele mehrseitiger Sicherheit und Privatsphäre . . . 53
2.6 Zugriffskontrolle . . . 56
2.6.1 Zugriffsmatrixmodell und Zugriffskontrollliste . . . 56
2.6.2 Weitere Zugriffskontrollmodelle . . . 57
2.6.3 Authentifizierung als Voraussetzung der Zugriffskontrolle . . . 59
2.7 Kryptografie . . . 59
2.8 Anonyme Credentials . . . 63
2.8.1 Eigenschaften von Credentials . . . 63
2.8.2 Bekannte Credential-Systeme . . . 65
2.8.3 idemix . . . 66
2.9 Zusammenfassung . . . 71

3 Einordnung der Arbeit in den Projektkontext 73
3.1 Kirchhof: Prozesse und Infrastrukturen . . . 73
3.2 Norbisrath: SCD-Prozess . . . 74
3.2.1 Vom klassischen Entwicklungsprozess zum SCD-Prozess . . . 74
3.2.2 Das eHome-Modell . . . 81
3.2.3 Werkzeugunterstützung: der eHomeConfigurator . . . 87
3.3 Retkowitz: kontinuierlicher SCD-Prozess . . . 88
3.3.1 Strukturelle Adaption . . . 89
3.3.2 Semantische Adaption . . . 91
3.3.3 Werkzeugunterstützung . . . 92
3.4 Bewertung . . . 92
3.4.1 Dienstauswahl . . . 92
3.4.2 Dienstinteraktion . . . 93
3.4.3 Personalisierung . . . 93
3.4.4 Schutz der Privatsphäre . . . 96
3.4.5 Schutz von eHomes . . . 96
3.5 Zusammenfassung . . . 96

4 Benutzermodell **99**
4.1 Entkopplung von Benutzerdaten . . . 100
4.2 Produzenten und Konsumenten . . . 103
4.3 Unterstützung der Intra-eHome-Mobilität . . . 104
4.3.1 Ausführungsort vs. Auswirkungsort . . . 105
4.3.2 Typen personalisierbarer Dienste . . . 105
4.3.3 Rolle des Benutzermodells . . . 109
4.4 Verwendete Ontologie . . . 111
4.4.1 Der Ontologie-Begriff . . . 111
4.4.2 General User Model Ontology (GUMO) . . . 112
4.4.3 Die Datenstruktur SituationalStatement . . . 113
4.5 Architektur des Benutzermodells . . . 114
4.5.1 DataStorage . . . 115
4.5.2 ProfileManager . . . 115
4.5.3 SessionManager . . . 116
4.5.4 Person . . . 117
4.5.5 SituationalStatement . . . 117
4.5.6 Beispiel . . . 118
4.6 Zusammenfassung . . . 120

5 Inter-eHome-Mobilität **121**
5.1 Dienstauswahl und -interaktion . . . 122
5.1.1 Erkennen und Auswählen eines eHomes . . . 123
5.1.2 Auswahl eines Dienstes . . . 124
5.1.3 Interaktion mit einem Dienst per Handheld . . . 124
5.2 Personalisierung . . . 126
5.2.1 Personalisierung durch Benutzerdaten . . . 126
5.2.2 Personalisierung durch persönliche Dienste . . . 132
5.3 Realisierung . . . 141
5.3.1 Grundlegende Eigenschaften von JXTA . . . 141
5.3.2 Neu entwickelte Komponenten . . . 143
5.3.3 SimpleRMI: RMI auf Basis von JXTA . . . 145
5.3.4 Architektur von SimpleRMI . . . 148
5.3.5 MobileCommunicator und eHomeCommunicator . . . 150
5.3.6 Verteilte Dienstkomposition und -kommunikation . . . 155
5.3.7 Realisierung von Benutzeroberflächen . . . 165
5.4 Zusammenfassung . . . 167

6 Privatsphäre **169**
6.1 Schutz auf eHome-Ebene . . . 170
6.1.1 Datensparsamkeit . . . 170
6.1.2 Unverkettbarkeit durch Anonymität . . . 174
6.2 Schutz auf Dienstebene . . . 183

6.2.1 Selektive Zugriffskontrolle für Benutzerdaten 184
6.2.2 Unverkettbarkeit durch Pseudonyme 186
6.3 Verschlüsselung persistenter Benutzerdaten 188
6.4 Sicherung der drahtlosen Kommunikation 189
6.5 Realisierung 189
6.5.1 Selektive Zugriffskontrolle für Benutzerdaten und deren Unverkettbarkeit 190
6.5.2 Integration von idemix in den eHome-Prototyp 194
6.5.3 Beteiligte Bausteine am Aushandlungsprozess 204
6.6 Zusammenfassung 208

7 Zugriffskontrolle 211
7.1 Schutz vor Zugriff unbefugter Benutzer 211
7.1.1 Authentifizierung über Beziehungspseudonyme 212
7.1.2 Authentifizierung über anonyme Credentials 213
7.1.3 Sonderfall: Basisdienste 214
7.2 Schutz vor Zugriff unbefugter Dienste 215
7.2.1 Sonderfall: mobil ausgeführte Dienste 218
7.3 Durchführung der Zugriffskontrolle 218
7.3.1 Durchführung durch Dienste 218
7.3.2 Durchführung durch einen Interceptor 219
7.3.3 Diskussion 220
7.4 Realisierung 221
7.4.1 Aspektorientierte Programmierung und AspectJ 221
7.4.2 Aspektorientierte Realisierung des Interceptors 224
7.5 Zusammenfassung 228

8 Werkzeugunterstützung und Demonstratoren 231
8.1 Werkzeugunterstützung im eHome 231
8.1.1 Benutzer 232
8.1.2 Identitäten 233
8.1.3 Dienste 235
8.1.4 Rollen 236
8.2 Werkzeugunterstützung auf dem Handheld 238
8.3 Demonstratoren 243
8.3.1 Der X10-Demonstrator 243
8.3.2 Der Lego-Demonstrator 243
8.3.3 Der eHomeSimulator 245
8.3.4 Der OpenSim/Second-Life-Demonstrator 246
8.3.5 Implementierte Anwendungsbeispiele 249
8.4 Zusammenfassung 250

9 Verwandte Ansätze **251**
9.1 Ansätze zur Mobilität . . . 251
9.1.1 Gaia . . . 251
9.1.2 P2PComp . . . 254
9.1.3 JXTA für Virtual Home Environments . . . 256
9.1.4 R-OSGi . . . 257
9.1.5 DIANE . . . 258
9.1.6 DynAMITE . . . 259
9.1.7 Mobile Geräte als Benutzerschnittstelle . . . 260
9.2 Ansätze zur Benutzermodellierung und Personalisierung . . . 260
9.2.1 CUMULATE . . . 260
9.2.2 Personis, Personis-lite und PersonisAD . . . 261
9.2.3 Der UserModelService . . . 263
9.2.4 Weitere Ansätze . . . 265
9.3 Ansätze zur Privatsphäre und Sicherheit . . . 266
9.3.1 Datensparsamkeit . . . 266
9.3.2 Anonymität durch MIXe . . . 270
9.3.3 Sicherheit und Privatsphäre in intelligenten Umgebungen . . . 273
9.3.4 Kooperative Schutzmechanismen . . . 277
9.3.5 Weitere Ansätze . . . 279

10 Schlussbemerkungen **283**
10.1 Zusammenfassung . . . 283
10.2 Ausblick . . . 287
10.3 Fazit . . . 288

Ausgewählte Veröffentlichungen **289**

Literaturverzeichnis **291**

Lebenslauf **317**

Abbildungsverzeichnis

1.1 Mögliche Anwendungsbereiche für eHome-Dienste. 3
1.2 Grobe Übersicht über die im Rahmen dieser Arbeit entwickelten Bausteine. 13

2.1 Lebenszyklus eines OGSi-Bundles (Quelle: [CG01]). 31
2.2 Schichtenarchitektur der OSGi-Dienstplattform (angelehnt an [OSG09a]). . 32
2.3 Die Eclipse-Architektur (Quelle: [SDF+04]). 33
2.4 Beispielhafte Skizze eines eHomes. 35
2.5 Hierarchische Dienstarchitektur am Beispiel des `Temperaturdienstes`. . . 37
2.6 Beispiel für die Identität einer Person. 44
2.7 Abbildung realer Identität auf digitale Identität. 45
2.8 Beispiel für Teilidentitäten. 46
2.9 Unterschiedliche Arten von Pseudonymen und ihr Zusammenhang zur Anonymität (angelehnt an [KP03]). 49
2.10 Beispiel für die Verkettbarkeit von Teilidentitäten. 50
2.11 Beispiel für Identitätsmanagement. 51
2.12 Kategorisierung von Schutzzielen. 55
2.13 Rollenbasierte Zugriffskontrolle (RBAC). 58
2.14 Symmetrische Verschlüsselung: Konzept. 60
2.15 Asymmetrische Verschlüsselung: Konzept. 61
2.16 Asymmetrische Verschlüsselung: Beispiel für Authentifizierung. 62
2.17 Grundlegende Protokolle von idemix (Quelle: [CH02]). 69

3.1 Klassischer Entwicklungsprozess (angelehnt an [Nor07]). 75
3.2 SCD-Prozess nach Norbisrath (angelehnt an [Nor07]). 76
3.3 Beispiel einer Dienstspezifikation (angelehnt an [Nor07]). 77
3.4 Beispiel einer Dienstkomponierung (angelehnt an [Nor07]). 79
3.5 Funktionalitäten von eHome-Diensten (angelehnt an [Nor07]). 81
3.6 Modellierung von Umgebungsinformationen (Quelle: [NARS06]). 82
3.7 Modellierung von Diensttypen (Quelle: [NARS06]). 83
3.8 Modellierung von Konfigurationen (Quelle: [NARS06]). 84
3.9 Modellierung des Personenkontexts (Quelle: [NARS06]). 85
3.10 Spezifikation des Musikdienstes mit dem eHomeConfigurator (Quelle: [Nor07]). 86
3.11 Spezifikation einer Umgebung mit dem eHomeConfigurator (Quelle: [Nor07]). 87
3.12 Auswahl von Diensten mit dem eHomeConfigurator (Quelle: [Nor07]). . . . 88

3.13 Parametrisierung von Diensten mit dem eHomeConfigurator (Quelle: [Nor07]). 89
3.14 Kontinuierlicher SCD-Prozess (angelehnt an [Ret10]). 90

4.1 Entkopplung von Benutzerdaten aus dem eHome-Modell. 101
4.2 Produzenten und Konsumenten von Benutzerdaten im eHome. 102
4.3 Unterschiedliche Typen personalisierbarer Dienste zur Handhabung der Intra-eHome-Mobilität am Beispiel des `Musikdienstes`. 106
4.4 Umkonfigurierung des personengebundenen `Musikdienstes`. 110
4.5 Hierarchische Struktur eines SituationalStatement (Quelle: [Hec06]). 113
4.6 Ausschnitt aus der Architektur des Benutzermodells. 115
4.7 SituationalStatement und seine Schichten. 118
4.8 Ausschnitt einer möglichen Ausprägung von Benutzerdaten im Benutzermodell. 119
4.9 Möglicher Ablauf einer Abfrage von Benutzerdaten am Beispiel des `Musikdienstes`. 120

5.1 Inter-eHome-Mobilität. 122
5.2 Erkennen und Auswählen eines eHomes. 123
5.3 Auswahl eines Dienstes über das Handheld. 124
5.4 Interaktion mit einem Dienst über das Handheld. 125
5.5 Mögliche Ansätze zur Lösung des Redundanzproblems. 127
5.6 Zusammenspiel mobiler Benutzermodelle mit dem Benutzermodell auf dem eHome-Gateway. 129
5.7 Ereignisse, die eine Synchronisierung von Benutzerdaten nach sich ziehen. . 130
5.8 Anzeige der im eHome verfügbaren Dienste. 134
5.9 Konfigurierung des `Weckdienstes` für die Ausführung im eHome. 135
5.10 `Weckdienst` nach dem Deployment im eHome. 136
5.11 Konfigurierung des `Weckdienstes` für die Ausführung auf dem Handheld. . 137
5.12 `Weckdienst` nach dem Deployment auf dem Handheld. 138
5.13 Umkonfigurierung des auf dem Handheld ausgeführten `Weckdienstes` im Rahmen der Intra-eHome-Mobilität. 140
5.14 Die JXTA-Architektur (Quelle: [Gon01c]). 142
5.15 JXTA verbindet verschiedene Arten von Peers über eine Vielfalt von Netzwerken zu Peer Groups. 143
5.16 Neue entwickelte Komponenten für Handheld und eHome-Gateway. 144
5.17 Funktionsweise von SimpleRMI. 146
5.18 Advertisements werden verwendet, um RMI-Server im Netzwerk zu veröffentlichen. 147
5.19 SimpleRMI-Klassendiagramm. 148
5.20 Der eHomeCommunicator stellt eine eigene JXTA-Pipe für Kommunikationsanfragen von Clients auf Handhelds bereit. 151
5.21 Klassendiagramm zum eHomeCommunicator. 152
5.22 Der MobileCommunicator stellt eine eigene JXTA-Pipe für Kommunikationsanfragen von Clients auf dem eHome-Gateway bereit. 153

5.23 Klassendiagramm zum MobileCommunicator. 153
5.24 Klassendiagramm zum personengebundenen Lichtdienst. 156
5.25 Klassendiagramm zum ServiceManager. 158
5.26 Beteiligte Komponenten an der Konfigurierung und dem Deployment des `Weckdienstes`. 159
5.27 Entfernte Dependency Injection am Beispiel des `Weckdienstes`. 162
5.28 Übertragen von dienstspezifischen Benutzeroberflächen auf das Handheld und ihre Ausführung am Beispiel des `Temperaturdienstes`. 166

6.1 Benutzer nutzt nicht alle personalisierbaren Dienste. 171
6.2 Daten werden entsprechend der genutzten Dienste offengelegt. 172
6.3 Grober Ablauf der Aushandlung von Diensten und Identitäten. 173
6.4 Verkettbarkeit im Rahmen der Inter-eHome-Mobilität. 175
6.5 Auswahl eines TTP-Credentials am Beispiel eines Lehrstuhls. 177
6.6 Auswahl eines TTP-Credentials am Beispiel eines Hotels. 178
6.7 Rollenbasiertes Session-Credential. 179
6.8 Dienstbasiertes Session-Credential. 180
6.9 Um anonyme Credentials erweiterter Aushandlungsprozess. 181
6.10 Vertraulichkeit durch selektiven Zugriff auf Benutzerdaten. 184
6.11 Unverkettbarkeit durch Pseudonyme. 187
6.12 Realisierung von Zugriffskontrolllisten mit der PrivacyBox. 190
6.13 Möglicher Ablauf einer Abfrage von Benutzerdaten mit einem Token am Beispiel des `Musikdienstes`. 192
6.14 Zusammenhang von Parametern, Schlüsseln und Credentials. 195
6.15 Eingaben von Empfänger und Aussteller eines Credentials. 196
6.16 Eingaben von Beweiser und Verifizierer eines Credentials. 197
6.17 IdemixHandhler und IdemixAuthenticator realisieren die Anbindung von idemix an den eHome-Prototyp. 200
6.18 Beteiligte Bausteine an der Aushandlung von Diensten und Identitäten. . . 206

7.1 Beispiel für befugte und unbefugte Dienstzugriffe. 215
7.2 Der Authenticator ist für die Verwaltung von Zugriffsrollen zuständig. . . . 217
7.3 Zugriffskontrolle durch die Dienste am Beispiel des `Faxdienstes`. 219
7.4 Zugriffskontrolle durch den Interceptor am Beispiel des `Faxdienstes`. . . . 220
7.5 Ein Joinpoint mit Kontextinformationen. 222
7.6 Ablauf der Zugriffskontrolle für einen Benutzerzugriff auf Basis eines Session-Credentials. 226
7.7 Ablauf der Zugriffskontrolle für einen Zugriff zwischen zwei Diensten. . . . 228

8.1 Zusammenhang zwischen Diensten, Rollen, Benutzern, Identitäten und Attributen. 232
8.2 eHomeAdministrator: Benutzerverwaltung. 233
8.3 eHomeAdministrator: Identitäten. 234
8.4 eHomeAdministrator: Bearbeiten eines Identitätsattributs. 235

8.5 eHomeAdministrator: Dienste. 236
8.6 eHomeAdministrator: Rollen. 237
8.7 MobileGUI: Startbild und Identitätsattribute. 238
8.8 MobileGUI: Auswählen von Diensten. 239
8.9 MobileGUI: Bearbeiten und Auswählen einer Identität. 240
8.10 MobileGUI: Auswählen der Authentifizierungsart und Interaktion mit einem Dienst. 241
8.11 Die MobileGUIauf einen Dell-Axim-PDA. 242
8.12 Der X10-eHome-Demonstrator (Quelle: [Nor07]). 244
8.13 Der Lego-eHome-Demonstrator (Quelle: [Nor07]). 244
8.14 Der eHomeSimulator. 245
8.15 Anbindung des OpenSim/Second-Life-Demonstrators an ein eHome-Gateway über XML-RPC und HTTP. 247
8.16 Der OpenSim/Second-Life-Demonstrator. 248

9.1 Struktur einer Anwendung in Gaia (nach [RHC02]). 252
9.2 Ports-Konzept von P2PComp (nach [FHMO04]). 255
9.3 Übersicht über die R-OSGI-Architektur (aus [RAR07]). 257
9.4 DIANE-Beispiel: Arbeitsprozess auf mobilem Gerät (angelehnt an [Vay05]). 258
9.5 Grobarchitektur von CUMULATE (Quelle: [BSS05]). 261
9.6 Architekturübersicht über Personis (Quelle: [KKL02]). 262
9.7 Prozedurale Übersicht über den UserModelService (Quelle: [Hec06]). 264
9.8 Ablauf eines Webseitenaufrufs mit P3P (Quelle: [Wor10]). 268
9.9 Überblick über den Aufbau von PawS (Quelle: [Lan02]). 269
9.10 Konzept eines MIX-Netzes (Quelle: [FJMP97]). 271
9.11 Interaktion zwischen MUSDAC-Komponenten (Quelle: [CRI07a]). 272
9.12 Architektur des Ansatzes von Lou et al. (Quelle: [RL07]). 273
9.13 Middleware for Secure Home Access and Control am C-LAB in Paderborn (Quelle: [MMS+07]). 275
9.14 Platform for Enterprise Privacy Practices (Quelle: [KSW03]). 278

Kapitel 1

Einführung

Die letzten Jahrzehnte sind von einem rasanten Fortschritt in der Computertechnik geprägt. Während sich in den 70er Jahren nur Großunternehmen und Universitäten eigene Rechner leisten konnten, waren in den 80er Jahren immer mehr Menschen in der Lage, persönliche Computer (PC) zu erwerben. Diese waren viel kleiner als ihre überdimensionierten Vorgänger und daher für private Haushalte erschwinglich geworden. Obwohl sich ihre Größe, ihre Kosten und ihr Energieverbrauch reduzierten [Mat08], nahm ihre Speicherkapazität und die Leistungsfähigkeit ihrer Prozessoren stetig zu. Heutzutage besitzen wir üblicherweise nicht nur einen Computer, sondern mehrere gleichzeitig, sei es in Form von PCs oder weiteren Geräten, die über Mikroprozessoren verfügen.

Die stetige Verkleinerung von Prozessoren wurde schon in den 60er Jahren von Gordon Moore beobachtet. Nach dem sogenannten *Mooreschen Gesetz* verdoppelt sich die Anzahl der Transistoren pro Flächeneinheit alle 18 Monate [Moo65]. Diese Entwicklung macht sich dadurch bemerkbar, dass gegenwärtig schon viele Haushaltsgeräte, Mobiltelefone, Uhren usw. Mikroprozessoren besitzen. Beispielsweise existieren inzwischen Automobile, in denen bis zu 100 Steuergeräte eingebettet sind.

Parallel dazu eröffnen uns weitere Entwicklungen in der Informationstechnik ein breites und neues Spektrum an Anwendungsmöglichkeiten. Eine dieser Entwicklungen macht sich durch die kontinuierliche Zunahme der *Speicherkapazität* bemerkbar. Während die Computer der 80er Jahre noch mit einigen Kilobyte Arbeitsspeicher auskommen mussten, verfügen heutige PCs über mehrere Gigabyte. Darüber hinaus sind die meisten Geräte, die über Mikroprozessoren und Speicher verfügen, heutzutage auch in der Lage, mit anderen Geräten kabelgebunden oder drahtlos zu *kommunizieren*. Als einige Beispiele seien Computer mit Ethernet oder WLAN, Handys mit UMTS, WLAN, GPRS oder viele andere Geräte mit USB, Bluetooth, x10 usw. genannt.

Mark Weiser war einer der ersten Forscher, der seine Visionen bzgl. der Weiterentwicklung von Computern veröffentlichte. In [Wei91] verkündete er die Allgegenwärtigkeit von Computern. Er prägte in diesem Zusammenhang den Begriff des *Ubiquitous Computing*, der dafür steht, dass die Technik in Zukunft immer mehr in den Hintergrund treten wird. Sie wird dann zwar benutzt, aber nicht mehr bewusst wahrgenommen. Seine Behauptung begründet Weiser dadurch, dass der PC nicht als Universalwerkzeug für jegliche Aufgaben

verwendet werden kann. Dafür sei er zu komplex und erfordere zu viel Aufmerksamkeit. Also geht er davon aus, dass dieser durch eine Technik verdrängt wird, die sich so in unsere Umwelt integriert, dass sie nicht mehr davon unterscheidbar ist.

Seit Weisers Aufsatz haben sich weltweit viele Forschergruppen aus Wissenschaft und Industrie mit Ubiquitous Computing und verwandten Themen befasst. In diesem Zusammenhang sind auch Begriffe wie *Pervasive Computing* oder *Ambient Intelligence* entstanden, die synonym zu Ubiquitous Computing verwendet werden. Unter ihnen befinden sich viele Projekte zur Entwicklung intelligenter Wohnungen bzw. Häuser [CD04, KOA+99, LAB+99, YHC05, Res03, Pad10, DTA10], die ihren Bewohnern durch das Zusammenspiel von Software und Geräten innovative Dienste mit neuartigen Funktionalitäten bieten können. Doch nicht ausschließlich private Haushalte sind ein Anwendungsfeld für Ubiquitous Computing. Beispielsweise wird im *inHaus2* [GS08], einer von der Fraunhofer-Gesellschaft (IMS) in Duisburg errichteten Nutzimmobilie zur Entwicklung von Techniken für intelligente Gebäude im Allgemeinen (engl. *Smart Buildings*), an integrierten Systemen für Krankenhäuser, Hotels, Büros und Veranstaltungen geforscht.

1.1 Das eHome-Projekt

Die vorliegende Arbeit ist im Rahmen eines Projekts entstanden, das durch Ubiquitous Computing motiviert ist. Das sogenannte *eHome-Projekt* ist am Lehrstuhl für Software Engineering der Rheinisch-Westfälischen Technischen Hochschule Aachen (RWTH Aachen) angesiedelt und wurde zum Teil von der Deutschen Forschungsgesellschaft (DFG) im Rahmen des Graduiertenkollegs „Software für mobile Kommunikationssysteme“ unterstützt. Die Forschungsgruppe (*eHome-Gruppe*) beschäftigt sich hauptsächlich mit softwaretechnischen Aspekten von vernetzten, automatisierten *Wohnungen und Häusern*, die im Kontext des eHome-Projekts als *eHomes* bezeichnet werden [Kir05, Nor07, Ret10, AK06, NARS06, RS08]. In der Literatur finden sich auch Begriffe wie *Intelligent Home*, *Smart Home*, *Smart Environment* oder auch *Active Space*, wobei sich einige dieser Begriffe nicht ausschließlich auf private Haushalte beziehen.

Im Gegensatz zu heute üblich ausgestatteten Häusern und Wohnungen haben eHomes zwei grundlegende Merkmale, die sie auszeichnen. Erstens sind in einem eHome alle Geräte *vernetzt*, d. h., sie können miteinander direkt oder indirekt kommunizieren. Zweitens sind die Funktionalitäten der Geräte nicht abgeschottet, sondern können zusätzlich durch *Software* gesteuert werden. Diese Merkmale ermöglichen die Entwicklung von Softwareanwendungen (*eHome-Dienste*), die den Bewohnern durch die Kopplung von Geräte- und Softwarefunktionalitäten einen gewissen *Mehrwert* an Nutzen und Komfort bieten können. Der Mehrwert entsteht dadurch, dass der Nutzen und Komfort durch neuartige Funktionalitäten über das hinaus gehen, was die Bewohner durch die getrennte Verwendung der einzelnen Geräte erlangen könnten.

Dabei existiert eine breite Palette von Anwendungsfeldern für eHome-Dienste. Abbildung 1.1 zeigt eine mögliche Kategorisierung von Anwendungsbereichen für Mehrwertdienste, die in eHomes zur Anwendung kommen können. Dem Bereich *Komfort* könnte beispielsweise

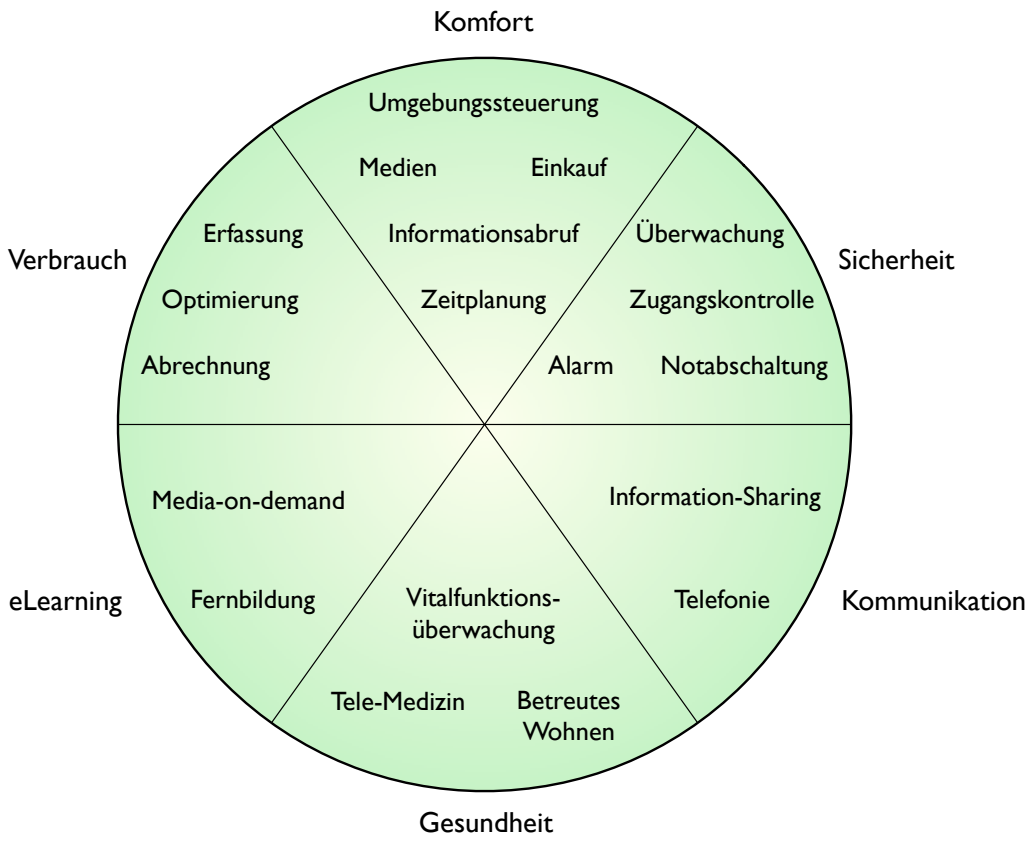


Abbildung 1.1: Mögliche Anwendungsbereiche für eHome-Dienste.

ein Dienst zur Umgebungssteuerung zugeordnet werden, der die Beleuchtung und die Temperatur automatisch an die Wünsche der Benutzer anpasst. Dazu müssten beispielsweise Geräte zur Personenerkennung, Schalter, Lampen und Heizkörper verknüpft werden. Wird der Dienst so erweitert, dass die Beleuchtung sich der Aktivität des Benutzers anpasst, etwa beim Fernsehen, kommen dann weitere Geräte dazu. Ein anderer Dienst aus dem Bereich *Sicherheit* kann Gefahrensituationen wie Einbruch oder Feuer automatisch erkennen und Alarm geben. Dazu könnten Glasbruchsensoren, Rauchmelder und Sirenen verknüpft werden. Zusätzlich zur Alarmierung per Sirene könnte der abwesende Benutzer per E-Mail oder MMS mit Fotos oder Videos des Ereignisses benachrichtigt werden. Es kämen also Kameras dazu. Aus dem Bereich *Gesundheit* könnte ein Dienst über unterschiedliche Sensoren die Vitalwerte von Benutzern überwachen und sie an die Einnahme ihrer Medikamente erinnern. Bodensensoren könnten zusätzlich dazu genutzt werden, Stürze älterer Personen zu erkennen, um dann medizinisches Personal zu benachrichtigen.

Obwohl auf dem Markt schon eine Vielzahl von netzwerkfähigen und günstigen Geräten existiert, die in eHomes eingesetzt werden können, muss festgestellt werden, dass sich eHomes trotz intensiver Forschung und visionärer Szenarien [IST01] noch nicht etablieren konnten. Dies liegt vor allem daran, dass die Entwicklung von eHome-Diensten für den Massenmarkt sowie die Einrichtung und der Betrieb von eHomes aufwendig und kostenintensiv sind. Die Herausforderungen hierbei sind vielfältig. In [BB02] werden beispielsweise drei

Hauptherausforderungen identifiziert, die bei der Entwicklung von intelligenten Umgebungen wie eHomes berücksichtigt werden müssen. Dies sind die *Dynamik* in einer Umgebung, die *Heterogenität* von Geräten und die Nutzung von Informationstechnik in einer *sozialen Umgebung*. Letzteres hat insbesondere Konsequenzen auf die *Privatsphäre* und ist ein zentrales Thema dieser Arbeit.

Die ersten beiden Herausforderungen dagegen wurden bereits im eHome-Projekt im Rahmen von zwei Vorarbeiten und einer parallel zu dieser laufenden Arbeit aus der Perspektive der Softwaretechnik behandelt.

Das erste im eHome-Projekt behandelte zentrale Thema ist die *Heterogenität* von Geräten und Diensten in eHomes. Aufgrund unterschiedlicher Ausprägungen einzelner eHomes muss die zum Betrieb benötigte Software speziell auf einzelne eHomes zugeschnitten sein. Die Entwicklung spezifischer Individualsoftware für jedes eHome ist jedoch mit hohen Kosten verbunden, sodass eHomes für den Normalverbraucher nicht bezahlbar sind. Die im eHome-Projekt entwickelten Konzepte ermöglichen hingegen die *Wiederverwendung komponentenbasierter* eHome-Dienste, die unabhängig von den Ausprägungen einzelner eHomes entwickelt werden können. Die Einrichtung eines eHomes reduziert sich dadurch auf einen Prozess der *Spezifizierung* des eHomes und der Kundenwünsche sowie der *Konfigurierung* von eHome-Diensten. Aufgrund der Wiederverwendbarkeit werden eHome-Dienste erschwinglich, wodurch der Weg für einen Massenmarkt geebnet wird.

Das zweite im eHome-Projekt behandelte Thema ist die *Dynamik*. Diese entsteht einerseits daraus, dass Benutzer und Geräte in eHomes mobil sind. Andererseits entsteht eine gewisse Dynamik dadurch, dass im Laufe der Zeit neue Geräte oder neue Dienste erworben werden. Damit in solchen Fällen auf die neue Situation ohne Unterbrechung des eHome-Betriebs reagiert werden kann, wurden hierfür spezielle Konzepte entwickelt. Beispielsweise ermöglicht eine dynamische Umkonfigurierung eines Musikdienstes zur Laufzeit, dass die Musik dem Benutzer von einem Raum zu einem anderen folgen kann.

Für die Umsetzung der entwickelten Konzepte wurden im Rahmen der Vorarbeiten spezielle Werkzeuge erstellt, die eHome-Betreiber sowohl bei der Einrichtung als auch beim Betrieb ihrer eHomes unterstützen.

1.2 Motivation

Bisherige Ergebnisse des eHome-Projekts dienen der Bewältigung von zwei wichtigen Herausforderungen, nämlich der Heterogenität der Geräte- und Dienstlandschaft sowie der Dynamik *in* eHomes (auch *Intra-eHome-Mobilität* genannt). Dennoch gibt es Fragestellungen, die bisher nicht behandelt wurden. Insbesondere wurden eHomes für sich allein als abgeschottete Umgebungen betrachtet. Folglich konzentrieren sich die entwickelten Konzepte auf Dienste und Benutzer, die sich ausschließlich in einem eHome befinden. Es gibt hingegen kaum Überlegungen, welche Aspekte zu beachten sind, wenn sich mobile Benutzer *zwischen* mehreren eHomes bewegen. Diese Art der Mobilität wird als *Inter-eHome-Mobilität* bezeichnet und stellt eine wesentliche Herausforderung für die Erforschung und Entwicklung intelligenter Umgebungen dar [IST01, FD09, BB96].

Einige der bisher nicht betrachteten Aspekte werden in dieser Arbeit behandelt. Hierzu gehören etwa die Unterstützung *mobiler* Benutzer bei der *Personalisierung* besuchter eHomes, die Wahrung der *Privatsphäre* der Benutzer sowie der Schutz von eHomes vor *unbefugtem Zugriff*. Um eine erste Idee von der Problematik zu bekommen, ist es hilfreich, zunächst ein Beispiel zu betrachten.

1.2.1 Szenario

Das Beispiel beschreibt den Tagesablauf des Geschäftsmanns Bob, der an einer Konferenz in einer entfernten Stadt teilnehmen möchte. Sein Haushalt ist mit unterschiedlichen eHome-Diensten ausgestattet, die ihm Funktionalitäten im Bereich Komfort, Sicherheit und Multimedia anbieten.

Am Morgen der Abreise wird Bob durch das Abspielen seiner Lieblingsmusik und durch das langsame Hochfahren der Rollläden geweckt. Er wurde früher geweckt als erwartet. Auf dem Display seines Weckers steht eine Nachricht, dass es über Nacht geschneit hat und die Reise wahrscheinlich länger dauern wird. Auch das Badezimmer und das Badewasser wurden schon früher als sonst auf die gewünschte Temperatur vorgeheizt.

Nach dem Duschen begibt sich Bob in die Küche, wo sein Kaffee schon fertiggebrüht wurde. Währenddessen spielt die Musik weiter und „folgt“ ihm ohne Unterbrechung von Raum zu Raum. Ähnlich verhält es sich mit der Beleuchtung, die automatisch in dem Raum angeht, in den sich Bob begibt.

Kurz darauf steigt Bob in sein Auto und fährt los. Im Auto wird die Musik an der gleichen Stelle fortgesetzt, an der sie im Haus gestoppt wurde. Diese Funktionalität ist insbesondere bei Hörbüchern oder Podcasts praktisch. Gleichzeitig ermittelt das Navigationssystem die Zieladresse aus Bobs Terminkalender, der auf seinem Handy verfügbar ist, und berechnet die Route.

Im Hotel angekommen, legt Bob seinen Personalausweis vor und erhält eine Schlüsselkarte mit elektronischen Informationen zum Öffnen seines Hotelzimmers. In seinem Zimmer angekommen, merkt er, dass seine Lieblingsmusik spielt und die Beleuchtung seinen Präferenzen entspricht. Dies liegt daran, dass Bob der Hotelkette bekannt ist und diese Kette personalisierbare Dienste anbietet. Während die Musik im Hintergrund läuft, nimmt er sich noch vor Beginn der Konferenz eine Flasche Bier aus der Minibar.

Kurze Zeit später verlässt er sein Zimmer und beschließt, sich ein wenig in der kleinen Einkaufspassage umzusehen, die zum Hotel gehört, aber nicht von diesem betrieben wird. Als er die Auslagen der Geschäfte betrachtet, fällt sein Blick auf eine kleine Apotheke und ihm fällt ein, dass er vor seiner Abreise nicht mehr geschafft hat, ein vom Arzt verschriebenes Medikament zu besorgen. Da die regelmäßige Einnahme wichtig ist, legt er das Rezept in der Apotheke vor und bezahlt per Kreditkarte.

Anschließend sucht er den Tagungsraum auf. Die Konferenz erweist sich für Bob als eine gute Gelegenheit, wichtige geschäftliche Kontakte zu knüpfen. Er ist mit zwei anderen Firmenvertretern ins Gespräch gekommen. Nachdem alle den Eindruck gewonnen haben, dass sich eine lukrative Zusammenarbeit ergeben könnte, wird beschlossen, das weitere Gespräch in einem vertraulicheren Rahmen zu führen. Da das Hotel eine hauseigene Sauna

hat, soll diese als weiterer Gesprächsraum genutzt werden. Weil es sich hier um eine kostenpflichtige Leistung handelt, müssen sich alle drei Geschäftsleute beim Betreten der Sauna mit ihren Schlüsselkarten identifizieren.

Am späten Abend sucht Bob schließlich sein Hotelzimmer auf. Da er jedoch noch nicht sofort schlafen möchte, genehmigt er sich noch zwei Flaschen Bier aus der Minibar und schaut sich einen kostenpflichtigen, nicht-jugendfreien Film an.

Am nächsten Tag wird Bob wie gewohnt mit seiner Lieblingsmusik und dem langsamen Beleuchten seines Zimmers geweckt. Auch die Wassertemperatur der Dusche ist automatisch auf die Wunschtemperatur eingestellt. Nach dem Frühstück begleicht er an der Rezeption die angefallenen Kosten. Dabei achtet er darauf, dass Geschäftskosten von den privat entstandenen Kosten getrennt berechnet werden. Anschließend reist er ab.

Gemeinsam mit Bob leben im gleichen Haushalt seine berufstätige Lebensgefährtin, seine zwei Kinder und seine pflegebedürftige Mutter. Die Mutter trägt mehrere Sensoren am Körper, die ihre Vitalwerte messen. Diese Vitalwerte werden von einem Medizindienst rund um die Uhr analysiert. Falls Notfälle auftreten, kann dieser Dienst automatisch das Pflegepersonal oder einen Notarzt benachrichtigen. Bei Ankunft des Rettungspersonals kann der Medizindienst die Außentür automatisch öffnen, falls sonst niemand im Haus anwesend ist. Zusätzlich wird die Mutter zeitweise von einer Pflegeschwester betreut, die auch Zutritt ins Haus hat. Diese Schwester hat Zugriff auf notwendige Dienste und Räumlichkeiten im Haus. Beispielsweise kann sie die Küche benutzen, um Gerichte aufzuwärmen. Sie kann ferner die von den Sensoren aufgezeichneten Werte über ihren PDA begutachten.

1.2.2 Analyse des Szenarios

Die Analyse des obigen Szenarios lässt zunächst erkennen, dass in privaten Haushalten *benutzeradaptive* Dienste ausgeführt werden, die *personalisierbar* sind, also ihre Funktionalitäten an ihre Benutzer anpassen können. Beispielsweise berechnet ein Weckdienst die optimale Weckzeit, heizt rechtzeitig das Badezimmer vor und weckt seine Benutzer nach Wunsch mit Musik, Licht oder in Kombination beider. Der Musikdienst spielt die Musik immer in dem Raum, in dem sich der Benutzer befindet. Ferner überwacht der Medizindienst die Vitalwerte seiner Benutzer und führt angemessene Aktionen aus, falls Probleme auftreten.

Das Beispiel zeigt darüber hinaus, dass personalisierbare Dienste nicht nur in privaten eHomes angeboten werden, sondern auch in *weiteren „eHomes“* wie in Autos und Hotels. Beispielsweise weiß der Musikdienst im Auto, welche Musik der Benutzer zuletzt gehört hat und setzt an der gleichen Stelle fort. Das gleiche gilt auch für den Musikdienst im Hotelzimmer. Auch einen Weckdienst mit den gewohnten Funktionalitäten findet der Gast in seinem Hotelzimmer vor, der seine Präferenzen und Termine berücksichtigt. Der Medizindienst überwacht die Vitalwerte seiner Benutzer und reagiert auf Notsituationen. Dabei benötigen alle personalisierbaren Dienste *Benutzerdaten*, um ihre Funktionalitäten an ihre Benutzer anpassen zu können. Tabelle 1.1 listet einige Benutzerdaten auf, die der Weckdienst oder der Medizindienst benötigen.

Während die Personalisierung von Diensten viele Vorteile mit sich bringt, ergeben sich durch das Hinterlassen personenbezogener Daten bzw. *digitaler Datenspuren* auch eine Reihe

Dienst	Benötigte Benutzerdaten
Weckdienst	Terminzeit
	Terminort
	Duschzeit
	Wassertemperaturpräferenz
	Raumtemperaturpräferenz
	Beleuchtungspräferenz
	Weckmusik
	Weckart
	Aufenthaltsort
Medizindienst	Puls
	Blutdruck
	Körpertemperatur

Tabelle 1.1: Beispiele für personalisierbare Dienste und benötigte Benutzerdaten.

bedenklicher Einflüsse auf die Privatsphäre. Nachfolgend werden diese Einflüsse diskutiert. Die Schlussfolgerung daraus wird dann sein, dass eHomes nicht ohne einen angemessenen Schutz der Privatsphäre betrieben werden sollten.

Das erste Problem ergibt sich bei der Anmeldung im Hotel. Hier legt Bob mit seinem Personalausweis ein amtlich beglaubigtes Dokument vor, womit er eindeutig identifiziert wird. Hierbei werden jedoch mehr Informationen offengelegt als nötig. Beispielsweise muss hier die Preisgabe der Hausanschrift oder des Geburtsdatums eigentlich nicht erfolgen. In einem pessimistischen Szenario könnte beispielsweise ein Mitarbeiter des Hotels mit kriminellen Absichten diese Informationen für einen Einbruch in Bobs Haus verwenden.

Das nächste Problem entsteht aufgrund der Schlüsselkarte im Hotel. Diese ermöglicht dem Gast einen komfortablen Zugang in sein Zimmer. Solche Schlüsselkarten werden immer mehr auch in Bürogebäuden eingesetzt. Im Gegensatz zum Einsatz nicht-elektronischer Schlüssel haben sie den Vorteil, dass sie leicht umkonfigurierbar sind und das komfortable Öffnen und Schließen der Türen erlauben. Sie haben jedoch den Nachteil, dass – entweder im Schloss oder auf einem zentralen Server – protokolliert werden kann, wer welchen Raum wann betreten oder verlassen hat. Allein die Erstellung eines solchen *Bewegungsprofils* kann sich negativ auf die Privatsphäre auswirken.

Solche Schlüsselkarten können über die übliche Zutrittskontrolle hinaus aber auch für die Abrechnung verschiedener Zusatzleistungen dienen, wie etwa der Sauna oder der Bezahlung der Getränke an der Bar bzw. der kostenpflichtigen Filme. Im Hotelbeispiel kann also zusätzlich ermittelt werden, wann ein Gast welche Dienstleistungen in Anspruch genommen hat. Die Kombination solcher Informationen über mehrere Gäste lässt sogar weiter reichende Schlussfolgerungen zu. Weil im obigen Fall drei Konferenzteilnehmer zur gleichen Zeit in der Sauna waren, kann daraus impliziert werden, dass diese im Privaten über mögliche Geschäfte gesprochen haben. Diese Information könnte unter Umständen weit schwerer wiegen als die reine Information, dass ein Geschäftsmann wohl gerne in die Sauna geht.

Darüber hinaus können die Informationen über das Rezept und die Kreditkarte dazu verwendet werden, das Bewegungsprofil auch auf die Einkaufspassage auszuweiten. Falls

das Hotel mit den übrigen Geschäften kolludiert, oder ein externer Beobachter deren Daten ausspäht, können die im Hotel gesammelten Daten über gewisse Vorlieben, wie Saunabesuch, Minibarkonsum oder Filmauswahl um weitere Informationen wie etwa über seine Krankheit angereichert werden.

Hieraus wird bereits deutlich, dass die Gefahr vor allem in der *Verkettung* von Informationen und den daraus ableitbaren Erkenntnissen liegt. Wenn die gesammelten Daten im Hotel und in den Geschäften nicht angemessen *geschützt* würden und eine dritte Person davon Kenntnis bekäme, z. B. Bobs Arbeitgeber, so könnte dieser den Konsum in der Minibar bemängeln oder aber die Information über die Krankheit als internen Grund heranziehen, sich von ihm zu trennen.

Das Problem, dass sich Transaktionen mit Kreditkarten oder allgemein die elektronische Zahlung negativ auf die Privatsphäre auswirken können, ist nicht neu und auch nicht eHome-spezifisch. Das für diese Arbeit wichtige Problem betrifft die Art der *zusätzlichen Informationen*, die bedingt durch die Verbreitung von eHomes nun einfacher gesammelt werden können, als es früher möglich war. Angefangen von den Vitalwerten bis zum Verhalten beim Fernsehkonsum sind darunter sensible Informationen enthalten, die einen sehr detaillierten Einblick in den Alltag, in die Lebensgewohnheiten und in die Vorlieben einer Person geben können. Damit können eHomes ihren Benutzern zwar eine reihe innovativer Dienste anbieten. Gleichzeitig eröffnen sie dadurch aber auch eine neue Ebene der Datensammlung, die sich negativ auf die Privatsphäre auswirken kann. Je mehr Daten nämlich anfallen, um so mehr Möglichkeiten ergeben sich für potenzielle Datendiebe. Beispielsweise ist es vorgesehen, dass die Pflegeschwester die Vitalwerte der Mutter per PDA abruft. Sollten die Daten ungesichert übertragen werden und in falsche Hände geraten, ist ein Missbrauch nicht mehr zu verhindern. Fallen solche Daten darüber hinaus im Rahmen der Inter-eHome-Mobilität auch unterwegs in fremden eHomes an, so verstärkt sich das Problem noch einmal erheblich.

Ein weiterer Aspekt, der dem Beispiel entnommen werden kann, betrifft den *Schutz von eHomes*. Sowohl die Gäste im Hotel als auch die Pflegeschwester der Mutter befinden sich nur zeitweise in einem bestimmten eHome und haben dort nur eingeschränkten Zutritt zu Räumen bzw. *Zugriff auf Dienste*. Beispielsweise dürfen Hotelgäste, die kein Entgelt für die Sauna zahlen möchten, diese auch nicht betreten. Die Pflegeschwester darf nicht das Schlafzimmer des Ehepaars betreten. Sie darf beispielsweise auch nicht den Sicherheitsdienst umkonfigurieren, der in Gefahrensituationen wie etwa bei einem Einbruch Alarm gibt. Die Kinder im Kindergartenalter sollen wiederum nicht ohne Aufsicht den Herd oder die Waschmaschine anschalten dürfen, um möglichen Verletzungen vorzubeugen.

Doch nicht nur Zugriffe von Benutzern sollten einem Kontrollmechanismus unterliegen, sondern auch die Zugriffe von Diensten untereinander. Beispielsweise kann der Medizindienst in Notfällen die Außentür automatisch öffnen. Diese Funktionalität ist in Notfällen sinnvoll und sollte erlaubt werden. Umgekehrt sollte jedoch nicht auch der Musikdienst die Möglichkeit haben, die Außentür zu öffnen, beispielsweise wenn sich niemand im Haus befindet, um Unbefugten den Zugang zum Haus zu ermöglichen. Daher sollte in jedem eHome ein Mechanismus vorhanden sein, der unbefugte Zugriffe auf eHome-Dienste unterbindet.

1.3 Anforderungen

Im Rahmen dieser Arbeit soll der bestehende eHome-Prototyp so erweitert werden, dass Szenarien wie das oben beschriebene realisiert und dabei die Wahrung der Privatsphäre der Benutzer sowie der Schutz von eHomes vor unbefugten Zugriffen gewährleistet werden können. Ein besonderes Merkmal solcher Szenarien ist die *Inter-eHome-Mobilität* von Benutzern, die sich in ihrem Alltag nicht nur in einem eHome aufhalten, sondern unterschiedliche eHomes besuchen und dort eHome-Dienste in Anspruch nehmen.

Im Folgenden werden wichtige Anforderungen beschrieben, die für die Unterstützung von Benutzern und eHome-Betreibern im Rahmen der Inter-eHome-Mobilität erfüllt werden sollten, aber im eHome-Projekt bisher überhaupt nicht oder nur rudimentär berücksichtigt wurden.

1.3.1 Dienstauswahl

Zunächst sollte ein mobiler Benutzer *auswählen* können, welche Dienste er in einem besuchten (fremden) eHome verwenden möchte. Bisher wurden Dienste im eHome-Projekt mithilfe eines Werkzeugs namens *eHomeConfigurator* installiert und standen somit den Benutzern zur Verfügung. Die Benutzung dieses Werkzeugs ist jedoch eher für den Administrator eines eHomes vorgesehen, nicht aber für normale Benutzer. Eine individuelle Einflussnahme normaler Benutzer auf die Dienstauswahl ist mit dem eHomeConfigurator daher nur mit erheblichem Aufwand möglich. Insbesondere für mobile Benutzer, die Dienste in fremden eHomes verwenden möchten, ist dieser Ansatz daher nicht angemessen. Ihnen muss also eine andere Möglichkeit geboten werden, eHome-Dienste auf eine einfache Art und Weise auswählen und nutzen zu können.

1.3.2 Interaktion von Benutzern mit Diensten

Der eHomeConfigurator wurde nicht nur für administrative Aufgaben wie das Konfigurieren und das Deployment von eHome-Diensten verwendet, sondern auch für die *Interaktion* der Benutzer mit diesen. Dazu gehört sowohl die Parametrisierung als auch das Starten und Stoppen der Dienste. Bisher konnten diese Aufgaben lediglich über vom eHomeConfigurator bereitgestellte Attributlisten durchgeführt werden. Dieser Ansatz ist in Bezug auf die *Bedienbarkeit* nicht zufriedenstellend, weil in einem eHome sehr unterschiedliche Dienste mit vielfältigen Interaktionsmöglichkeiten zum Einsatz kommen können. Diese Vielfältigkeit sollte möglichst auf dienstspezifische Benutzeroberflächen abgebildet werden, und zwar unabhängig von dem jeweiligen eHome. Zudem sollten ähnliche Funktionalitäten auch durch ähnliche Benutzeroberflächen bedienbar sein, damit sich die Benutzer in neuen eHomes mit minimalem Aufwand zurechtfinden können.

1.3.3 Personalisierung

Wie die Analyse des Szenarios hervorgebracht hat, können mobile Benutzer sowohl daheim als auch unterwegs personalisierbare eHome-Dienste verwenden. Um ihre Funktionalitäten

an ihre Benutzer anpassen zu können, benötigen solche Dienste Informationen über die Präferenzen, Gewohnheiten oder Wünsche ihrer Benutzer. Diese Informationen werden als *Benutzerdaten* bezeichnet. Eine in diesem Zusammenhang zu klärende Frage betrifft die Art und Weise der Bereitstellung dieser Daten für personalisierbare Dienste.

Bislang wurde auch für diesen Zweck der eHomeConfigurator eingesetzt. Er kann verwendet werden, um Benutzerdaten anzulegen, diese zu bearbeiten und Diensten über eine Datenstruktur zur Verfügung zu stellen. Abgesehen davon, dass die Bedienbarkeit nicht zufriedenstellend ist, existiert noch kein Mechanismus, Benutzerdaten komfortabel mehreren eHomes bereitzustellen. Bisher musste jeder Benutzer seine Daten in jedem eHome separat anlegen und pflegen. Dies führte zu einem *Redundanzproblem*, das den Personalisierungsaufwand erhöhte. Es besteht daher Bedarf für einen neuen Ansatz, der die eHome-übergreifende Bereitstellung von Benutzerdaten mit minimalem Aufwand ermöglicht.

Ferner besteht für einen Benutzer derzeit keine Möglichkeit, *eigene Dienste* in einem fremden eHome zu nutzen, falls das eHome selbst keinen Dienst anbietet, der die gewünschten Funktionalitäten realisiert. Falls der Benutzer aber einen ähnlichen Dienst besitzt und das eHome die für diesen Dienst benötigte Infrastruktur bereitstellen kann, sollte auch eine Möglichkeit geboten werden, diesen Dienst nutzen zu können.

1.3.4 Schutz der Privatsphäre

Wie in der Analyse des Szenarios erläutert, sammeln eHomes im hohen Maße Benutzerdaten, mit dem Ziel, ihren Benutzern mit personalisierbaren Diensten einen größtmöglichen Komfort zu bieten. Die gesammelten Daten haben einen sensiblen Charakter, weil sie einen sehr detaillierten Einblick in den Alltag, die Lebensgewohnheiten und die Vorlieben einer Person bieten können. Im Rahmen der Inter-eHome-Mobilität fallen solche sensiblen Daten auch unterwegs in fremden eHomes an, wodurch das Problem noch weiter verschärft wird.

Überall dort, wo personenbezogene Daten anfallen, besteht auch die Gefahr ihres Missbrauchs (s. etwa [Wel09, Hei09]). Dieses Problem wird durch die Verbreitung allgegenwärtiger Datenverarbeitungstechnik verstärkt und zunehmend in das private Lebensumfeld übertragen. Durch die Digitalisierung der Welt werden neue Möglichkeiten des Missbrauchs von Benutzerdaten geschaffen. Die Möglichkeit des sogenannten gläsernen Menschen rückt damit in greifbare Nähe [Mat08].

Diese Diskussion führt zu der Erkenntnis, dass ein wesentlicher *Konflikt* zwischen der *Personalisierung* von eHomes auf der einen Seite und dem *Schutz der Privatsphäre* auf der anderen Seite besteht. Während die Personalisierung voraussetzt, dass Benutzer ihre Daten freigeben, um maximalen Komfort zu genießen, verlangt der Schutz der Privatsphäre das Geheimhalten von Benutzerdaten und -identitäten.

Eines der Hauptziele dieser Arbeit ist die Entwicklung von Konzepten zur Aufweichung des genannten Konflikts. Dadurch sollen personalisierbare eHome-Dienste genutzt werden können, ohne dabei auf das Grundrecht der *informationellen Selbstbestimmung* verzichten zu müssen. Dieses Grundrecht „gewährleistet die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen“, wie das Bundesverfassungsgericht am 15.12.1983 in seiner Entscheidung zur Volkszählung dargelegt

hat (BVerfGE 65, 1). Dabei sollte die informationelle Selbstbestimmung nicht als „*eigentumsähnliches Herrschaftsrecht über personenbezogene Daten*“ verstanden werden, sondern als „*Schutz des selbstbestimmt in der Gesellschaft Agierenden und Kommunizierenden*“ [Roß07]. Mit anderen Worten bedeutet dies, dass die komplette Unterbindung der Preisgabe von Benutzerdaten und eine damit einhergehende völlige Abschottung von Personen von der Gesellschaft zugunsten der Privatsphäre nicht im Interesse einer funktionierenden Gesellschaft sein kann [Mat08]. Umgekehrt muss eine Person jedoch selbst bestimmen können, was mit ihren Daten passiert.

Um dieses Ziel zu erreichen, müssen wesentliche Bestandteile des „*normativen Schutzprogramms*“ des derzeitigen deutschen Datenschutzrechts berücksichtigt werden. Zu diesen gehören die *Einwilligung* des Betroffenen zur Verwendung personenbezogener Daten, die *Transparenz* der Datenverarbeitung ihm gegenüber („wer weiß was wann über ihn“), die *Zweckbindung* und *-begrenzung* der Datenverwendung, die *Erforderlichkeit* der Datenerhebung und verbunden damit die *Minimierung* und das frühzeitige Löschen der Daten sowie die Ermöglichung der *Mitwirkung* des Betroffenen bei der Datenverarbeitung zusammen mit *Auskunfts-* und *Korrekturrechten* [Roß07].

Im besten Falle wird die Umsetzung dieses Schutzprogramms die Datenschutzsituation auf einen ähnlichen Stand bringen wie vor der Einführung von intelligenten, Daten sammelnden Umgebungen. Zumindest wird sie jedoch dem Benutzer die Kontrolle über seine Daten geben und eine Verletzung der Privatsphäre durch Datenmissbrauch erschweren. Es ist ein zentrales Ziel dieser Arbeit, diese Anforderung für eHomes umzusetzen.

1.3.5 Schutz von eHomes

Schließlich hat die Analyse des obigen Szenarios gezeigt, dass eHomes, oder genauer gesagt eHome-Dienste, vor unbefugtem Zugriff geschützt werden müssen. Dadurch soll in Kombination mit dem Schutz der Privatsphäre *mehrseitige Sicherheit* erreicht werden, die sowohl die Interessen von Benutzern als auch die von eHome-Betreibern berücksichtigt.

Dabei sollen Konzepte ausgearbeitet werden, die zwei Arten von Zugriffen auf eHome-Dienste kontrollieren können. Erstens müssen Zugriffe von *Diensten* berücksichtigt werden, die andere Dienste benutzen. Die Zugriffe sollen so kontrolliert werden, dass ein Dienst nur jene Dienste verwenden kann, die er zur Realisierung seiner eigentlichen Funktionalitäten benötigt. Auf diese Weise sollen bösartige Dienste keine Möglichkeit haben, im eHome unerlaubte Aktionen auszuführen.

Zweitens müssen die Zugriffe von *Benutzern* berücksichtigt werden, die in einem eHome mit unterschiedlichen Funktionen und Rollen agieren können. Sie sollen nur die Dienste nutzen können, für deren Verwendung sie eine Berechtigung besitzen.

Es sollte hier erwähnt werden, dass es einen Zusammenhang zwischen einer Zugriffskontrolle und dem Schutz der Privatsphäre gibt. In nicht-vernetzten Haushalten reicht es aus, einen mechanischen Schlüssel zu haben, um sich Zutritt in Gebäude bzw. Räume zu verschaffen. Das Aufschließen der Türe entspricht dabei einer anonymen Authentifizierung, bei der der Schlüssel die geforderte Berechtigung repräsentiert. Nachdem die Tür geöffnet ist, kann prinzipiell jeder den Raum betreten oder verlassen, ohne dass das von einem

System erkannt wird. Ferner können Lampen oder Fernsehgeräte ohne eine Zugriffskontrolle und anonym benutzt werden.

In eHomes ist das nun nicht mehr so. Um ihre Berechtigung zur Nutzung von Diensten nachzuweisen bzw. bestimmte Räume zu betreten, wird verlangt, dass sich die Benutzer elektronisch authentifizieren. Abhängig davon, welches Verfahren für die Authentifizierung verwendet wird, können Benutzeraktionen erkannt und einer konkreten Person zugeordnet werden. Dadurch wird die Sicherheit in eHomes gewährleistet, gleichzeitig kann sich dies aber nachteilig auf den Schutz der Privatsphäre auswirken. Offensichtlich ergibt sich auch hier ein *Konflikt*, den es zu lösen gilt. Während ein Authentifizierungsmechanismus für den sicheren Betrieb von eHomes entwickelt wird, sollte darauf geachtet werden, dass dabei eine möglichst starke Anonymitätsstufe eingehalten wird. Denn nur dann, wenn die Benutzer das Gefühl haben, nicht einer ständigen Überwachung ausgeliefert zu sein, kann eine breite Akzeptanz von eHomes erreicht werden.

1.4 Lösungsskizze

Aufbauend auf den bisherigen Erläuterungen werden in diesem Abschnitt die Beiträge dieser Arbeit zur Erfüllung der zuvor geschilderten Anforderungen beschrieben. Ein grober Überblick über den in dieser Arbeit verfolgten Lösungsansatz ist in Abbildung 1.2 dargestellt. Es erweitert den bereits existierenden eHome-Prototyp um neue Bausteine und Funktionalitäten. Das Grundkonzept basiert dabei auf der Nutzung eines Handhelds, einem Smartphone oder einem PDA, das drahtlos mit eHomes kommunizieren kann. Das Handheld hat die Aufgabe, seinen Benutzer im Rahmen der Inter-eHome-Mobilität bei der Ausführung wiederkehrender Aufgaben zu unterstützen. In Abbildung 1.2 ist das Handheld zusammen mit den zugehörigen Bausteinen auf der linken Seite abgebildet, wohingegen das eHome mit seinen Bausteinen auf der rechten Seite dargestellt ist. Im Folgenden wird erläutert, welche Rolle diese Bausteine bei der Umsetzung der in Abschnitt 1.3 erörterten Anforderungen spielen.

1.4.1 Dienstauswahl

Zu den in Abschnitt 1.3 beschriebenen Anforderungen gehört zunächst die Unterstützung des Benutzers bei der *Dienstauswahl* nach dem Betreten eines eHomes. Das Handheld verbindet sich drahtlos mit dem eHome und ruft über den InformationService allgemeine Informationen über das eHome ab. Falls der Benutzer sich dafür entscheidet, sich beim eHome anzumelden, zeigt ihm das Handheld die Liste der im eHome verfügbaren eHome-Dienste an. Die Liste der eHome-Dienste wird von dem ServiceManager im eHome bereitgestellt. Aus diesen Diensten kann der Benutzer diejenigen auswählen, die er verwenden möchte. Für die Darstellung der Benutzeroberfläche auf dem Handheld ist der Baustein MobileGUI zuständig.

Um den Interaktionsaufwand des Benutzers zu reduzieren, kann die Auswahl des Benutzers auf dem Handheld gespeichert und bei erneutem Betreten desselben eHomes automatisch ausgewählt werden.

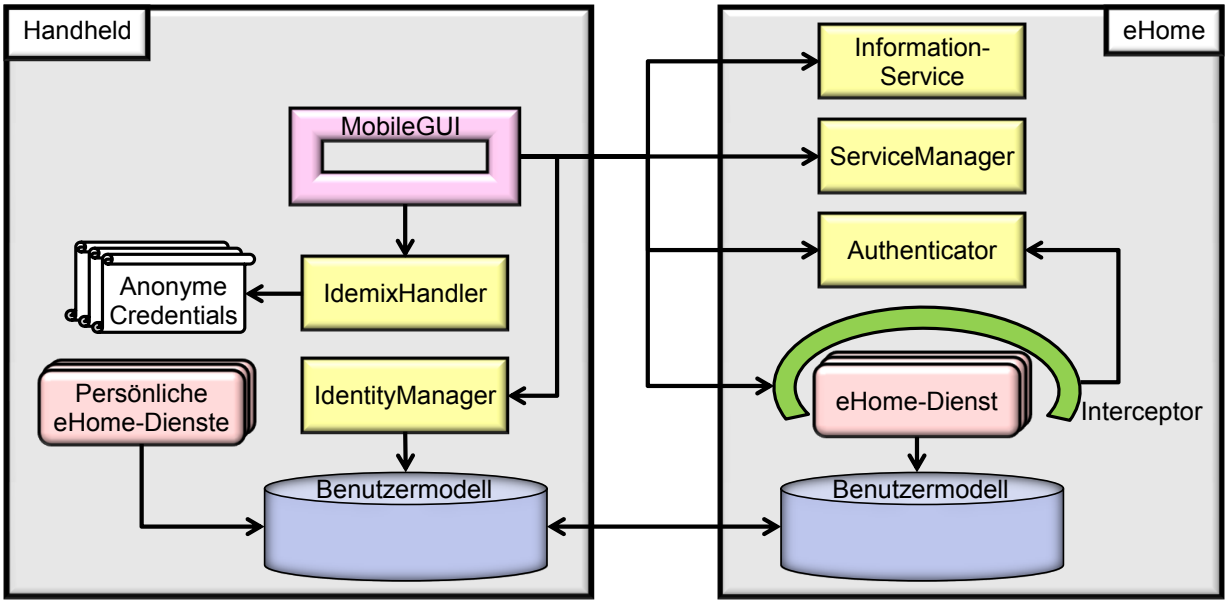


Abbildung 1.2: Grobe Übersicht über die im Rahmen dieser Arbeit entwickelten Bausteine.

1.4.2 Interaktion von Benutzern mit Diensten

Durch die Nutzung solch eines Handhelds werden dem Benutzer auch neue Möglichkeiten zur Interaktion mit Diensten geboten. Um mit einem eHome-Dienst zu interagieren, kann sich der Benutzer die Benutzeroberfläche des Dienstes auf seinem Handheld anzeigen lassen und die gewünschten Funktionalitäten, wie etwa das Starten der Musik oder das Anpassen der Beleuchtung, bedienen. Der Lösungsansatz sieht hierfür vor, dass jeder Dienst mit seiner spezifischen Benutzeroberfläche ausgeliefert wird, die zur Laufzeit auf ein Handheld übertragen und angezeigt werden kann. Dadurch können, im Gegensatz zum aktuellen Stand des eHome-Prototyps, jeweils auf die Funktionalitäten der Dienste angepasste Benutzeroberflächen entwickelt werden. Somit können die Benutzer mit den Diensten auf eine praktikablere Art und Weise *interagieren*, als es mit dem eHomeConfigurator möglich war.

1.4.3 Personalisierung

Für die Personalisierung von eHomes wird zunächst ein *Benutzermodell* entwickelt, das für die Verwaltung von Benutzerdaten und ihre Bereitstellung für personalisierbare Dienste zuständig ist. Diese Benutzerdaten wurden im eHome-Prototyp zuvor nicht separat verwaltet, sondern waren mit anderen Daten im eHome vermischt. Durch die *Entkopplung* dieser Daten wird die Verwaltung von Benutzerdaten vereinfacht und die Wartbarkeit von eHomes verbessert.

Das Benutzermodell spielt darüber hinaus eine wichtige Rolle für den gesamten Lösungsansatz dieser Arbeit. Ein auf Basis des Benutzermodells entwickeltes Konzept dient der Behebung des zuvor beschriebenen *Redundanzproblems*, das durch die Inter-eHome-

Mobilität bedingt ist. Hierfür wird das Benutzermodell so erweitert, dass es auch auf Handhelds ausgeführt werden kann (*mobiles Benutzermodell*, s. linke Seite der Abbildung 1.2). Mobile Benutzer können somit ihre persönlichen Daten auf ihren Handhelds mitnehmen und eHomes unterwegs zur Verfügung stellen. Ein *Synchronisierungsmechanismus* sorgt dafür, dass Benutzerdaten zwischen Handheld und eHome *automatisch* ausgetauscht werden. Somit müssen Benutzerdaten nicht mehr in jedem eHome separat gespeichert oder manuell eingegeben werden, weil die aktuellen Daten stets auf dem Handheld vor Ort verfügbar sind.

Die Verfügbarkeit von Handhelds bietet ferner eine neue, bislang nicht mögliche Art der Personalisierung. Bisher konnte ein Benutzer nur die Dienste in Anspruch nehmen, die das eHome angeboten hat. Falls jedoch kein Dienst angeboten wurde, der bestimmte Funktionalitäten realisiert, musste der Benutzer ohne diese Funktionalitäten auskommen. Im Rahmen dieser Arbeit wird der eHome-Prototyp so erweitert, dass mobile Benutzer *persönliche Dienste* mitnehmen und auf ihren Handhelds *mobil ausführen* können (s. Abbildung 1.2). Die einzige Voraussetzung hierfür ist, dass das eHome die Geräte und Unterdienste bereitstellt, die für die Realisierung der Funktionalitäten des mobilen Dienstes benötigt werden. Die Erweiterung sieht dabei vor, dass der mobile Dienst mit seinen Unterdiensten im eHome kommunizieren kann. Aus Übersichtsgründen ist diese Kommunikation in Abbildung 1.2 nicht dargestellt. Auf diese Weise können mobile Benutzer gewohnte Funktionalitäten auch in solchen eHomes nutzen, die diese Funktionalitäten nicht direkt über vorhandene Dienste anbieten. Für den Benutzer macht es dabei keinen Unterschied, ob der Dienst auf dem Handheld oder im eHome ausgeführt wird. Eine weitere Besonderheit dieses Ansatzes besteht darin, dass sich die Verteilung auch nicht auf die Dienstentwicklung auswirkt.

1.4.4 Schutz der Privatsphäre

Für den Schutz der Privatsphäre wird ein Konzept entwickelt, das mobilen Benutzern im Rahmen der Inter-eHome-Mobilität die Möglichkeit zur informationellen Selbstbestimmung bietet.

Hierfür werden auf einer ersten Ebene persönliche Daten betrachtet, die ein mobiler Benutzer einem eHome mindestens zur Verfügung stellen muss, um in diesem personalisiere Diensten zu verwenden. Um den oben beschriebenen Konflikt zwischen Personalisierung und dem Schutz der Privatsphäre zu lösen, wird ein Ansatz zum *selbstgesteuerten Identitätsmanagement* verfolgt. Mit anderen Worten werden mobile Benutzer in die Lage versetzt, entscheiden zu können, welchen eHomes sie wann mit welcher Identität gegenüber treten möchten. Dabei wird in dieser Arbeit als *Identität* die *Menge und Ausprägung* der persönlichen Daten verstanden, die ein Benutzer dem eHome preisgibt. Gleichzeitig basiert der Ansatz auf dem Grundsatz der *Datensparsamkeit*, um die Menge der einem eHome preisgegebenen Daten auf ein notwendiges Minimum zu reduzieren. Das notwendige Minimum hängt dabei von den personalisierbaren Diensten ab, die der Benutzer in dem eHome nutzen möchte.

Für diesen Zweck wird der eHome-Prototyp so erweitert, dass der Benutzer mit einem eHome aushandeln kann, welche personalisierbaren Dienste er verwenden und welche Daten

er hierfür preisgeben möchte. An diesem *Aushandlungsprozess* sind die MobileGUI und der ServiceManager aus Abbildung 1.2 beteiligt. Ein weiterer Baustein, der in diesem Prozess eine wichtige Rolle spielt, ist der IdentityManager. Dieser sorgt auf dem Handheld dafür, dass das mobile Benutzermodell nur die Daten freigibt, die in der ausgehandelten Identität enthalten sind. Ferner kann der Benutzer mithilfe dieses Bausteins Identitäten anlegen, verwalten und löschen. Auch die Ausführung personalisierbarer Dienste auf Handhelds begünstigt die Datensparsamkeit, weil die von diesen Diensten benötigten Daten auf dem Handheld des Benutzers verbleiben können.

Mit dem IdentityManager kann zwar kontrolliert werden, welches eHome welche persönlichen Daten erhält. Doch wie schon in Abschnitt 1.2.2 diskutiert wurde, liegt eine große Gefahr auch in der *Verkettbarkeit* von Identitäten. Wenn mehrere eHomes miteinander kolludieren, könnten sie ihre Daten kombinieren und mehr Informationen über die Benutzer erlangen, als ihnen tatsächlich zusteht. Um solch eine Verkettbarkeit zu verhindern, oder zumindest zu erschweren, wird der eHome-Prototyp so erweitert, dass *anonyme Credentials* [CH02] für die Authentifizierung und für die Zugriffskontrolle verwendet werden können. Durch die Verwendung solcher Credentials können Benutzer einem eHome gegenüber bestimmte Eigenschaften nachweisen, ohne ihre echte Identität preiszugeben. Diese Eigenschaften müssen dem Benutzer lediglich zuvor von einer vertrauenswürdigen dritten Stelle (engl. *Trusted Third Party* (TTP)) in Form von Credentials bescheinigt worden sein. Da der Nachweis der Eigenschaften anonym funktioniert, kann kein Bezug zwischen den preisgegebenen Daten und der reellen Identität des Benutzers hergestellt werden. Infolgedessen wird Verkettbarkeit von Identitäten zwischen mehreren eHomes, und auch zwischen mehreren Sitzungen im selben eHome, unterbunden (*Unverkettbarkeit auf eHome-Ebene*).

In Abbildung 1.2 sind beispielhaft einige Credentials auf dem Handheld des Benutzers dargestellt. Außerdem ist der IdemixHandler abgebildet, der für die Verwaltung der anonymen Credentials auf dem Handheld zuständig ist. Der IdemixHandler berechnet auch die Nachweise für die vorzuzeigenden Eigenschaften auf Basis vorhandener Credentials und kommuniziert diese mit dem Authenticator im eHome.

Auf einer zweiten Ebene werden die Benutzerdaten betrachtet, die einem eHome schon preisgegeben wurden und durch das Benutzermodell im eHome verwaltet werden. Für die folgenden Überlegungen wird dabei angenommen, dass dem Benutzermodell im eHome mehr vertraut werden kann, als den eHome-Diensten im gleichen eHome. Während Ersteres von einer vertrauenswürdigen Stelle stammen kann, wird es viele verschiedene Anbieter von eHome-Diensten geben, die neben ihrer eigentlichen Funktionalität auch extensiv Benutzerdaten z. B. für Werbezwecke sammeln könnten. Es sollten daher nicht allen Diensten eines eHomes die gleichen, ausgehandelten Daten vollumfänglich zur Verfügung gestellt werden.

Hierfür wird das Benutzermodell um eine *selektive Zugriffskontrolle* erweitert, die unbefugten Zugriff auf Benutzerdaten verhindert. Dadurch wird die Menge der einen Dienst zur Verfügung gestellten Daten auf das notwendige Minimum reduziert. Dabei kann der Benutzer *interaktiv* entschieden, welcher Dienst auf welche Daten zugreifen darf. Einmal getroffene Entscheidungen werden im mobilen Benutzermodell gespeichert,

sodass in ähnlichen Situationen keine neue Interaktion notwendig ist. Auch hier besteht die Möglichkeit der Verkettbarkeit der Daten durch das Zusammenspiel mehrerer Dienste. Diese Art der Verkettbarkeit wird durch den Einsatz von *Pseudonymen* verhindert, die das Benutzermodell für jede Dienst-Benutzer-Beziehung erstellt. Somit können die Dienste nicht herausfinden, welche anderen Dienste von demselben Benutzer verwendet werden. Weil so das *Dienstnutzungsprofil* der Benutzer verborgen wird, ist eine Verkettbarkeit von personenbezogenen Daten außerhalb des Benutzermodells nicht möglich (*Unverkettbarkeit auf Dienstebene*). Diese Art der Pseudonymisierung ist in Abbildung 1.2 nicht explizit gezeigt, sondern implizit durch den Pfeil zwischen den eHome-Diensten und dem Benutzermodell repräsentiert.

1.4.5 Schutz von eHomes

Während der Schutz der Privatsphäre Benutzerinteressen schützt, sollte auch der *Schutz von eHomes* gewährleistet werden, um mehrseitige Sicherheit zu erreichen. Der Schutz von eHomes wird hier mit dem Schutz von eHome-Diensten vor unbefugten Zugriffen gleichgesetzt. Dabei wird zwischen zwei Arten von Zugriffen unterschieden, nämlich zwischen Zugriffen von *Diensten* und Zugriffen von *Benutzern*.

Für die Kontrolle von Zugriffen zwischen Diensten wird ein Konzept entwickelt, der auf dem rollenbasierten Zugriffskontrollmodell (engl. *Role-Based Access Control* (*RBAC*)) basiert. Demnach wird für jeden Dienst automatisch eine *Dienstrolle* erzeugt, die genau vorgibt, auf welche Unterdienste zugegriffen werden darf. Als Grundlage für die Generierung von Rollen dient die aktuelle *Dienstkomposition* im eHome, die alle expliziten Benutzungsbeziehungen von eHome-Diensten enthält. Dadurch wird sichergestellt, dass ein Dienst nur auf die Dienste zugreifen darf, mit denen er in der Dienstkomposition verbunden ist. Weil diese Komposition unter der Kontrolle des eHome-Betreibers steht, wird davon ausgegangen, dass Dienstzugriffe im Rahmen der Komposition für einen reibungslosen Betrieb nötig sind und erlaubt sein sollten. Der Ansatz kann auch auf dynamische Änderungen der Dienstkomposition zur Laufzeit reagieren, also Rollen löschen oder neu erstellen, falls Dienste dazukommen, umkonfiguriert oder entfernt werden.

Die Kontrolle von Zugriffen durch Benutzer hängt von der oben erwähnten Aushandlung der Identität und der damit verbundenen Art der Authentifizierung ab. Im Rahmen dieser Arbeit wird der eHome-Prototyp so erweitert, dass mobile Benutzer zwischen mehreren Möglichkeiten zur Authentifizierung wählen können.

Die erste Möglichkeit erlaubt eine einfache Authentifizierung per *Benutzername* und Passwort. Hierfür wird vorausgesetzt, dass der Benutzer möchte, dass sein Benutzername im eHome gespeichert wird. Der Benutzername wird einer Rolle zugeordnet, die für die Zugriffskontrolle herangezogen wird. Auf diese Weise wird auch hier eine Art rollenbasierte Zugriffskontrolle realisiert. Diese Möglichkeit sollte jedoch nur in privaten eHomes eingesetzt werden, weil das eHome unterschiedliche Sitzungen des gleichen Benutzers über den Benutzernamen verketten kann.

Die zweite Möglichkeit erlaubt die Authentifizierung der Benutzer mit den vorhin erwähnten *anonymen Credentials*. Solche Credentials können für die Authentifizierung in

eHomes eingesetzt werden, weil sie den zuverlässigen Nachweis von Eigenschaften erlauben, die dem Benutzer von einer TTP bescheinigt wurden. Beispielsweise kann ein eHome abhängig von der Liste der ausgehandelten Dienste verlangen, dass der Benutzer durch ein geeignetes Credential seine Volljährigkeit oder die Buchung eines All-Inclusive-Urlaubs nachweisen muss. Bei Erfolg wird für den Benutzer ein einmaliges Pseudonym erstellt, das bis zum Ende seiner Sitzung gültig ist. Ferner wird dieses Pseudonym mit einer Rolle verknüpft, die dem Benutzer Zugriff auf die ausgehandelten Dienste erlaubt. Weil in diesem Fall sowohl das eHome als auch der Benutzer der TTP vertrauen, bietet diese Möglichkeit einen guten Kompromiss zwischen Zurechenbarkeit und Anonymität. Dies ist insbesondere in eHomes mit viel Benutzerfluktuation nützlich, weil hier auch nicht vorausgesetzt wird, dass die Benutzer dem eHome zuvor bekannt sind. Somit kann die Benutzerverwaltung in eHomes flexibel gestaltet werden.

Obwohl die zweite Alternative die Verkettbarkeit mehrerer Sitzungen des gleichen Benutzers unterbindet, kann innerhalb einer Sitzung beobachtet werden, wann der Benutzer welche Dienste verwendet. Beispielsweise könnte im obigen Szenario festgestellt werden, welche Benutzer gemeinsam in der Sauna sind. Um dies zu vermeiden, kann sich der Benutzer für eine erweiterte Version der zweiten Möglichkeit entscheiden. Er erhält in diesem Fall nach der Authentifizierung ein automatisch generiertes anonymes *Session-Credential*, dass ihn berechtigt, genau auf die Dienste zugreifen zu können, die er mit dem eHome ausgehandelt hat. Jedes Mal, wenn der Benutzer mit einem Dienst interagiert, muss er fortan mit seinem Session-Credential nachweisen, dass er berechtigt ist, den entsprechenden Dienst zu verwenden. Weil diese Nachweise aber anonym vollzogen werden, kann das eHome diese Interaktionen nicht auf den konkreten Benutzer zurückführen. Auf diese Weise wird Anonymität innerhalb einer Sitzung erzielt.

Jede dieser Möglichkeiten kann für die Kontrolle von Benutzerzugriffen auf Dienste verwendet werden. Es sollte jedoch erwähnt werden, dass mit steigender Stärke der Anonymität der Grad der Benutzbarkeit abnimmt. Der in dieser Arbeit verfolgte Ansatz gibt nicht vor, welche Alternative zu wählen ist. Stattdessen wird dem Benutzer die Flexibilität geboten, situationsabhängig den richtigen Kompromiss zu treffen. Es wird auch dem eHome-Betreiber freigestellt, welche Authentifizierung er zulassen möchte.

Für die Authentifizierung von Benutzern und die Kontrolle von Zugriffen auf eHome-Dienste in eHomes ist der Authenticator zuständig, der auf der rechten Seite von Abbildung 1.2 dargestellt ist. Der Authenticator teilt der MobileGUI mit, welche Möglichkeiten dem Benutzer für die Authentifizierung geboten werden. Ferner überprüft er die Nachweise des Benutzers auf Gültigkeit, also entweder den Benutzernamen und das Passwort oder die von dem IdemixHandler berechneten Nachweise auf Basis der Credentials.

Um überprüfen zu können, ob Zugriffe auf Dienste gewährt oder abgelehnt werden sollen, wird der *Interceptor* eingesetzt (s. Abbildung 1.2). Er ist in der Lage, alle Zugriffe, sowohl von Benutzern als auch von Diensten, auf eHome-Dienste abzufangen und in Zusammenarbeit mit dem Authenticator auf deren Befugnis zu überprüfen. Abhängig von dem Ergebnis der Überprüfung entscheidet er, ob der Zugriff gewährt oder abgelehnt wird.

1.5 Gliederung der Arbeit

In diesem Kapitel wurde das zu behandelnde Thema eingeführt und der Lösungsansatz skizziert. Die weiteren Kapitel dieser Arbeit sind wie folgt strukturiert.

Kapitel 2 beschreibt die Grundlagen des hier vorgestellten Ansatzes. Zunächst werden einige allgemeine Begriffe und Techniken sowie die Struktur von eHome-Systemen vorgestellt. Anschließend werden Anwendungsbeispiele für eHomes beschrieben, von denen einige in den folgenden Kapiteln zur Illustration der vorgestellten Konzepte herangezogen werden. Danach werden grundlegende Konzepte zur Sicherheit und Privatsphäre beschrieben, darunter fallen u. a. Identitätsmanagement, Datenschutz, Zugriffskontrolle, Kryptografie und anonyme Credentials.

Kapitel 3 führt in das eHome-Projekt ein, der den Kontext der vorliegenden Arbeit bildet. Dabei werden die übrigen Arbeiten der eHome-Gruppe, also die von Kirchhof [Kir05], Norbisrath [Nor07] und Retkowitz [Ret10], beschrieben. Weil der hier vorgestellte Ansatz auf den Ergebnissen der letzteren beiden Arbeiten basiert, wird detaillierter auf diese beiden eingegangen. Von besonderer Wichtigkeit ist der von Norbisrath eingeführte und von Retkowitz weiterentwickelte SCD-Prozess und das dazugehörige eHome-Modell. Anschließend werden die erzielten Ergebnisse in Bezug auf die Ziele der vorliegenden Arbeit bewertet.

Kapitel 4 führt das Benutzermodell ein, das für die Verwaltung von Benutzerdaten zuständig ist, die zuvor zusammen mit den übrigen Daten im eHome-Modell abgelegt wurden. Dabei wird auch erläutert, warum diese Daten aus dem eHome-Modell entkoppelt werden. In diesem Zusammenhang wird das Zusammenspiel des Benutzermodells mit den Produzenten und Konsumenten von Benutzerdaten sowie ihre Rolle bei der Unterstützung der Mobilität von Benutzern innerhalb eines eHomes erläutert. Ferner wird in diesem Kapitel die der Modellierung von Benutzerdaten zugrunde liegende Ontologie und deren Abbildung auf die Architektur des Benutzermodells beschrieben.

Kapitel 5 stellt die Konzepte zur Unterstützung mobiler Benutzer im Rahmen der Inter-eHome-Mobilität vor. Am Anfang dieses Kapitels werden die Auswahl von und die Interaktion mit Diensten über das Handheld beschrieben. Anschließend werden unterschiedliche Möglichkeiten der Personalisierung von eHomes diskutiert. Hier wird insbesondere die Rolle des Benutzermodells bei der Lösung des Redundanzproblems hervorgehoben. Gleichzeitig werden die entwickelten Konzepte und ihre Auswirkungen auf den eHome-Prototyp dargelegt. Einzelheiten zur technischen Realisierung etwa der drahtlosen Kommunikation zwischen Handhelds und eHomes oder von Verteilungsaspekten bilden den Schluss dieses Kapitels.

Kapitel 6 beschreibt den Ansatz zum Schutz der Privatsphäre im Rahmen der Inter-eHome-Mobilität. Hier wird schrittweise erläutert, wie der Benutzer die Kontrolle über die Freigabe seiner Daten erhält und wie die Verkettbarkeit der Daten sowohl auf eHome- als auch auf Dienstebene verhindert werden kann. Anschließend werden Einzelheiten

zur Realisierung der zuvor eingeführten Konzepte beschreiben. Insbesondere wird dargelegt, wie das Benutzermodell den Schutz von Benutzerdaten sicherstellt und wie das anonyme Credential-System idemix in den eHome-Prototyp integriert wird.

Kapitel 7 widmet sich der Zugriffskontrolle für eHome-Dienste. Zunächst werden alternative Mechanismen zur Kontrolle von Zugriffen durch Benutzer vorgestellt, durch die sichergestellt werden kann, dass nur befugte Benutzer auf eHome-Dienste zugreifen können. Diese lassen sich grob in Credential- und rollenbasierte Mechanismen unterteilen. Anschließend wird ein flexibler, rollenbasierter Ansatz für die Kontrolle von Zugriffen von Diensten untereinander beschrieben. Dieses Kapitel wird nach einer Diskussion alternativer Realisierungsmöglichkeiten zur Durchführung der Zugriffskontrolle mit den Einzelheiten zur Realisierung der Konzepte abgeschlossen.

Kapitel 8 erläutert die entwickelten Werkzeuge zur Umsetzung des Lösungsansatzes. Eingangs wird ein Werkzeug beschrieben, der den eHome-Betreiber bei der Verwaltung von Benutzern sowie deren Identitäten und Zugriffsrechten unterstützt. Danach wird das für Handhelds entwickelte Werkzeug erörtert, den mobile Benutzer für die Ausführung unterschiedlicher Aufgaben im Rahmen der Inter-eHome-Mobilität verwenden können, beispielsweise für die Interaktion mit Diensten oder die Verwaltung ihrer Identitäten. Abschließend werden die Demonstratoren dargestellt, die zur Evaluation der entwickelten Konzepte und Werkzeuge herangezogen wurden. Zwei dieser Demonstratoren sind in Software realisiert und wurden im Rahmen dieser Arbeit entwickelt.

Kapitel 9 stellt einen Ausschnitt verwandter Arbeiten dar, die sich mit ähnlichen Konzepten wie in dieser Arbeit beschäftigt haben.

Kapitel 10 schließt die Arbeit mit einer Zusammenfassung der erzielten Ergebnisse und einem Ausblick für mögliche Forschungsarbeiten ab, die auf diesen Ergebnissen aufbauen können.

Kapitel 2

Grundlagen

In diesem Kapitel werden wichtige Grundlagen beschrieben, die für das Verständnis der Arbeit erforderlich sind. Zunächst werden allgemeine Begriffe und Konzepte wie Benutzermodellierung, Kontextbezogenheit und komponentenbasierte Softwareentwicklung geklärt. Anschließend wird die in dieser Arbeit verwendete Komponentenplattform OSGi vorgestellt. Als Nächstes folgt die Erläuterung der Struktur von eHome-Systemen. Danach werden Anwendungsbeispiele für eHome-Dienste vorgestellt, von denen einige später zur Erläuterung der erarbeiteten Konzepte herangezogen werden. Hiernach werden Begriffe wie Datenschutz, Identitätsmanagement und Anonymität erörtert, die mit Privatsphäre in Verbindung stehen. Ferner werden elementare Eigenschaften von Zugriffskontrollmodellen und der Kryptografie skizziert. Abgeschlossen wird dieses Kapitel mit der Beschreibung von anonymen Credentials, die eine interessante Möglichkeit der anonymen Authentifizierung bieten.

2.1 Allgemeine Begriffe und Konzepte

In dieser Arbeit wird der eHome-Prototyp um ein Benutzermodell erweitert, das für die Verwaltung von Benutzerdaten zuständig ist. Die im eHome-Projekt entwickelten eHome-Dienste haben einen kontextsensitiven Charakter und basieren auf einer komponentenbasierten Architektur. Die grundlegende Kenntnis dieser Konzepte ist daher wichtig für das Verständnis der später vorzustellenden Konzepte. Aus diesem Grund werden sie in den nächsten Unterabschnitten beschrieben.

2.1.1 Benutzermodelle

Die ersten Ansätze zur Benutzermodellierung reichen in die späten 70er Jahre zurück. Damals haben Allen et al. [All79] und Rich [Ric79] mehrere *benutzeradaptive* [Sch01] Anwendungen aus unterschiedlichen Domänen entwickelt. Diese Anwendungen zeichneten sich dadurch aus, dass sie Informationen über ihre Benutzer gesammelt haben und ihre Funktionalität auf Basis dieser Informationen an ihre Benutzer anpassen konnten. Damals gehörte die Benutzermodellierung zu den Aufgaben der Anwendungen, sodass nicht konkret zwischen den Funktionalitäten zur Benutzermodellierung und anderer Aufgaben unter-

schieden werden konnte. Kobsa definiert in [KW89] ein Benutzermodell (engl. *User Model*) jedoch folgendermaßen:

> "A user model is a knowledge source in a system which contains explicit assumptions on all aspects of the user that may be relevant to the behavior of the system. These assumptions must be separable by the system from the rest of the system's knowledge."

Diese Forderung nach der Trennung des Benutzermodells vom restlichen Anwendungssystem, um die Erweiterbarkeit und Wartbarkeit des Systems zu gewährleisten, erfüllten die ersten Ansätze Mitte der 80er Jahre mit der Auslagerung des Benutzermodells in eigene *anwendungsspezifische* Bausteine. Den Vorreiter dieser Ansätze bildet das 1986 entwickelte *General User Modeling System* (GUMS) [FD86]. Es folgten weitere prominente Beispiele wie um [Kay94], UMT [BT94] und TAGUS [PS95]. Sie alle gehören zur Klasse der Benutzermodell-Shells (engl. *User Model Shells*) und unterscheiden sich meist in ihrer Art, Informationen zu repräsentieren und durch Schlussfolgerung neue zu generieren.

Sie alle sind trotz der genannten Trennung zur Laufzeit jedoch immer noch „Teil" *eines* Anwendungssystems. Erst die sogenannten Benutzermodell-Server (engl. *User Modeling Servers*) konnten komplett von einer speziellen Anwendung getrennt ausgeführt werden und Informationen mit Anwendungen per Inter-Prozess-Kommunikation austauschen. Diese Server können somit auch verteilt auf entfernten Rechnern ausgeführt werden. Diese Art von Client/Server-Architektur bringt unter anderem den Vorteil mit sich, dass mehrere Anwendungssysteme zur Laufzeit mit demselben Benutzermodell interagieren können, um Daten desselben Benutzers auszutauschen. Bekannte Beispiele von Benutzermodell-Servern sind *BGP-MS* [KP95], *DOPPELGÄNGER* [Orw94], *CUMULATE* [BSS05], *Personis* [KKL02] und *UMS* [KF06].

2.1.1.1 Aufgaben und Eigenschaften von Benutzermodellen

Es gibt im Allgemeinen einige Kernaufgaben, für die ein Benutzermodell zuständig ist. In ihrem Aufsatz [PSK99] identifizieren Pohl et al. folgende Aufgaben für Benutzermodelle:

- **Erfassung**: Das System sammelt (engl. *Acquisition*) Informationen über Benutzereigenschaften wie seine Präferenzen, Gewohnheiten oder Wünsche. Dies kann z. B. durch die Analyse der Benutzerinteraktion mit dem Anwendungssystem oder aber auch durch die direkte Interaktion mit dem Benutzer über spezielle Menüoptionen zum Bearbeiten von Präferenzen geschehen.

- **Repräsentation**: Das System speichert (engl. *Representation*) gesammelte Informationen einheitlich in einem Repository. Dadurch können sowohl Anwendungen als auch das Benutzermodell selbst effektiv auf die Daten zugreifen.

- **Schlussfolgerung** Das System kann aus den schon vorhandenen Informationen neue generieren (engl. *Reasoning*).

- **Entscheidung**: Benutzeradaptive Anwendungen entscheiden (engl. *Decision*) basierend auf den im Benutzermodell vorhandenen Informationen, wie sie ihre Funktionalität an einen Benutzer anpassen.

Eine etwas längere Liste von häufig vorkommenden Eigenschaften von allgemeinen Benutzermodellen führt Kobsa in seinem Aufsatz [Kob07] auf. Das sind im Einzelnen die Allgemeinheit und Domänenunabhängigkeit, Ausdruckskraft und Schlussfolgerung, schnelle Anpassung an neue Benutzer, Erweiterbarkeit, Import benutzerbezogener Informationen von externer Stelle, Verwaltung verteilter Informationen, Unterstützung von offenen Standards, Load Balancing, Ausfallsicherheit, Transaktionskonsistenz und Schutz der Privatsphäre. Während die ersten beiden Eigenschaften schon früh von Interesse waren, als die Benutzermodellierung hauptsächlich dem Gebiet der künstlichen Intelligenz zugeordnet wurde, haben letztere Eigenschaften in den letzten Jahren mehr an Bedeutung gewonnen. Zu begründen ist dies dadurch, dass immer mehr kommerzielle Benutzermodelle für die Personalisierung von Webseiten eingesetzt werden.

2.1.1.2 Benutzerdaten

Die in einem Benutzermodell abgelegten Informationen können in unterschiedliche Kategorien unterteilt werden. Zunächst kann zwischen *Roh-* und *verarbeiteten Daten* unterschieden werden. Verarbeitete Daten sind solche, die nach ihrer Erfassung durch das Benutzermodell weiter verarbeitet wurden. In [AT99] unterteilen Adomavicius et al. Rohdaten in *demografische* und *transaktionale* Daten. Demografische Daten beschreiben, um *wen* es sich handelt. Hierzu gehören etwa Name, Alter, Geschlecht, Adresse, Einkommen usw. Dies sind besonders sensible Daten, die daher geschützt werden sollten. Transaktionale Daten sind solche, die beschreiben, *was* ein Benutzer tut. Hierzu gehören etwa die besuchten Orte (*Bewegungsprofil*) oder die *Interaktionshistorie* mit einer bestimmten Anwendung. Aus diesen Daten können dann Benutzerprofile erstellt werden, die wiederum in *faktische* Profile und *Verhaltensprofile* unterteilt werden können. Während faktische Profile demografische Daten und weitere, teilweise generierte, Fakten wie „Benutzer hält sich hauptsächlich im Wohnzimmer auf“ enthalten, umfassen Verhaltensprofile Informationen wie „Benutzer trinkt immer Bier auf Geschäftsreisen“. In [AT99] wird die automatische Gewinnung solcher Informationen mithilfe von Data-Mining-Verfahren erforscht.

Auch Hämmerle et al. [HWRB05] unterteilen personenbezogene Daten in zwei Kategorien. Daten über einen Benutzer, also demografische Daten, werden für die *Personalisierung* der Interaktion mit dem System verwendet. Dagegen dienen Benutzerpräferenzen der *Individualisierung*. In der vorliegenden Arbeit wird diese Unterscheidung nicht explizit durchgeführt. Stattdessen wird der Begriff der *Personalisierung* für die Anpassung personalisierbarer Dienste an Benutzerdaten verwendet (s. auch Abschnitt 2.2.4).

Ferner unterscheiden Kuflik et al. [KS00] *inhaltsbasierte* und *kollaborative* Profile. Dabei ist ein Profil inhaltsbasiert, wenn es aus einer Liste von Name-Wert-Paaren besteht. Kollaborative Profile hingegen werden für die Gruppierung von Benutzern auf Basis ähnlicher Präferenzen verwendet, wobei ein Benutzer unterschiedlichen Gruppen zugeordnet werden

kann. Benutzerdaten werden in der vorliegenden Arbeit im weiteren Sinne in Form von inhaltsbasierten Profilen modelliert.

Eine etwas umfangreichere Unterteilung von Informationen in Benutzermodellen nimmt Kobsa in [Kob04] vor. Er unterscheidet zwischen den Hauptgruppen *Benutzerdaten*, *Benutzungsdaten* und *Umgebungsdaten*. Benutzerdaten umfassen demografische Daten sowie Informationen über die Präferenzen, Ziele, Fähigkeiten, Interessen und Pläne von Benutzern. Benutzungsdaten beziehen sich auf Interaktionen von Benutzern mit Anwendungen, wie etwa Selektionen, Benutzungshistorie, Bewertungen, Aktionssequenzen usw. Umgebungsdaten dagegen enthalten beispielsweise Informationen über die Hardware- und Softwareumgebung oder den Aufenthaltsort des Benutzers.

In dieser Arbeit werden Informationen, die für die Personalisierung von eHome-Diensten relevant sind, in einem Benutzermodell verwaltet. Für diese Informationen wird der Begriff *Benutzerdaten* verwendet, eine detaillierte Unterteilung wie etwa bei Kobsa ist für diese Arbeit nicht erforderlich. Gelegentlich werden auch die Begriffe *persönliche Daten* und *personenbezogene Daten* verwendet. Persönliche Daten umfassen Informationen, die ein mobiler Benutzer auf einem Gerät mit sich „trägt“, die er unterwegs für die Personalisierung von besuchten eHomes verwenden kann. Sie stammen also explizit vom Benutzer. Benutzerdaten können hingegen zusätzlich zu den vom Benutzer stammenden persönlichen Daten weitere, von eHome-Diensten oder anderen Komponenten generierte Daten enthalten. Der Begriff personenbezogener Daten wird dagegen im Kontext der Privatsphäre verwendet und bezeichnet die Daten, die in Bezug zu einer konkreten Person gesetzt werden können.

2.1.1.3 Erfassung und Verarbeitung von Benutzerdaten

Um Benutzerdaten verwalten zu können, müssen diese zunächst *erfasst* werden. Es existieren dabei unterschiedliche Möglichkeiten zur Erfassung von Benutzerdaten. In [Chi93] werden hierfür fünf orthogonale Dimensionen aufgeführt. Demnach kann ein Benutzermodell *aktiv* mit dem Benutzer interagieren oder *passiv* seine Aktionen beobachten. Ferner können dem Benutzermodell die Daten *benutzerinitiiert* oder *automatisch* zugespielt werden. Weiterhin kann ein Benutzermodell nur *logisch korrekte* Daten speichern oder aber die Daten mit verschiedenen *Plausibilitätsstufen* versehen. Außerdem können die gespeicherten Daten entweder *direkt* mit Benutzeraktionen assoziiert werden oder *indirekt*, wenn sie mit komplexen Inferenzregeln generiert wurden. Schließlich können Benutzerdaten *online* erfasst und verarbeitet werden, währen der Benutzer die Anwendung verwendet, oder *offline*, wenn die Interaktion mit der Anwendung beendet wurde.

Viele Benutzermodelle sind in der Lage, erfasste Rohdaten weiter zu verarbeiten. Traditionell wurden für die *Verarbeitung* der erfassten Daten Wissensrepräsentationsmechanismen eingesetzt. In den letzten Jahren sind auch maschinelle Lernverfahren dazugekommen [PSK99]. Einige dieser Mechanismen werden im Folgenden beschrieben:

- **Tell/Ask**: Die einfachste Möglichkeit des Informationsaustauschs findet über eine *Tell/Ask*-Schnittstelle statt. Benutzeradaptive Anwendungen teilen dem Benutzermodell über die Tell-Funktion neue Benutzerdaten mit. Umgekehrt können sie über die

Ask-Funktion Daten erfragen. Zusätzlich kann das Benutzermodell anfangs oder bei Bedarf den Benutzer nach Benutzerdaten fragen.

- **Accretion/Resolution**: Dieser Ansatz verfolgt das Ziel, erfasste Daten (*Accretion*) nicht direkt an Anwendungen weiter zu geben, sondern erst bei Bedarf, also wenn eine Anwendung die Daten anfordert (ähnlich zu Ask). Tritt dieser Fall ein, werden die erfassten Rohdaten analysiert, um die angefragten Informationen zu generieren (*Resolution*). Dieser Ansatz wurde in [Kay94] eingeführt. Das Ziel dieses Ansatzes ist die Erhöhung der Flexibilität durch Einsatz unterschiedlicher Resolver für verschiedene Zwecke und der Effizienz durch bedarfsorientierte Ausführung der Resolution.

- **Stereotypen**: Die zuerst in [Ric79] als „a collection of frequently occurring characteristics of a user“ eingeführten *Stereotypen* (*Stereotypes*) dienen der Gruppierung von Benutzern. Beispielsweise können Benutzer basierend auf ihrer Erfahrung mit einer Anwendung in Anfänger und Experten unterteilt werden. Ist ein Benutzer Mitglied einer Gruppe, werden automatisch bestimme Attribute mit ihm assoziiert. In [FD86] werden Stereotypen noch mal unterteilt in änderbare und nicht-änderbare Teile. Während *nicht-änderbare* Teile für jeden Benutzer gleich sind, können *änderbare* Teile für jeden Benutzer individuell angepasst werden. Weitere Erweiterungen von Stereotypen finden sich z. B. in [Chi89].

- **Data Mining**: Eine weitere Möglichkeit, Benutzerprofile zu erstellen, ist das *Data Mining*. Dazu werden durch Anwendung von meist statistisch-mathematischen Methoden große Datenmengen, z. B. an Benutzerinteraktionen, analysiert, um Muster zu erkennen, aus denen dann z. B. Benutzerprofile erstellt werden können.

- **Maschinelles Lernen**: Ähnlich wie bei Data Mining ist das Ziel des maschinellen Lernens die Ermittlung zusätzlicher Informationen durch die Analyse einer initialen Datenmenge. Im Kontext der Benutzermodelle ist das Ziel die Ermittlung von Benutzerpräferenzen oder -gewohnheiten. Beispielsweise wurden in [MCF+94] Entscheidungsbäume eingesetzt, um die Präferenzen von Benutzern bei der Festlegung von Terminen zu ermitteln. Auf Basis der gewonnen Erkenntnisse wurde ein intelligentes Kalendersystem entwickelt. Maschinelles Lernen erfordert jedoch eine zu große initiale Datenmenge, sodass dieser Ansatz nicht für Systeme geeignet ist, bei denen sich die Anwendungen möglichst schnell an die Benutzer anpassen sollen.

2.1.1.4 Benutzermodelle in dieser Arbeit

Im Rahmen dieser Arbeit wird ein Benutzermodell entwickelt, das Benutzerdaten in eHomes verwaltet, die über Sensoren oder direkt durch Benutzerinteraktionen erfasst und personalisierbaren eHome-Diensten zur Verfügung gestellt werden (s. Kapitel 4). Insofern existiert eine Ähnlichkeit zum Tell/Ask-Verfahren. Dieses Benutzermodell erfüllt nicht alle von Kobsa erwähnten Eigenschaften, sondern beschränkt sich vielmehr auf die für die Forschungsziele dieser Arbeit relevanten Merkmale wie auf das Erfassen und Bereitstellen von Benutzerdaten,

auf den Schutz der Privatsphäre und auf die Verwaltung verteilter Informationen. Es ist jedoch so entworfen, dass es leicht um benötigte Eigenschaften erweitert werden kann.

In seinem Aufsatz [Kob07] stellt Kobsa fest, dass sich Ubiquitous Computing und die fortschreitende Mobilisierung von Computern auch auf Benutzermodelle auswirken werden. Daher schlägt er vor, Benutzermodelle so zu entwerfen, dass sie auch *mobil* verfügbar sind. Nur so könnten den Benutzern weiterhin überall personalisierbare Dienste angeboten werden. In der vorliegenden Arbeit wird besonders auf diesen Aspekt eingegangen. Das Ergebnis ist eine mobile Version des Benutzermodells, die auf einem Handheld ausgeführt werden kann. Dadurch haben mobile Benutzer persönliche Daten überall verfügbar und können sie bei Bedarf eHomes zur Verfügung stellen (s. Kapitel 5).

2.1.2 Kontextbezogenes Computing

Eine wichtige Eigenschaft von eHomes bzw. eHome-Diensten, wie sie in dieser Arbeit vorkommen, ist ihre *Kontextbezogenheit*. Die Forschung bezüglich der Entwicklung und des Betriebes kontextbezogener Anwendungen hat Ihren Ursprung in dem in [WHFG92] beschriebenen Personen-Lokalisierungssystem. Eine kontextbezogene Anwendung ist dabei ein Programm, dessen Funktionalität und Interaktionsverhalten mit seinen Benutzern an Kontextinformationen angepasst bzw. adaptiert werden kann. Die allgemein akzeptierte Definition des Kontextbegriffes hat sich seit der gezielten Erforschung des kontextbezogenen Computing stetig weiterentwickelt. Eine weitverbreitete Definition stammt von Dey et al. aus [ADB+99]. Demnach wird jegliche Information als Kontext bezeichnet, die die Situation einer Entität charakterisiert. Dabei kann eine Entität eine Person, ein Ort oder ein Objekt sein, das für die Interaktion zwischen Benutzer und Anwendung von Relevanz ist. Durch die Adaption von Anwendungsfunktionalität an den Kontext kann die Software optimal an die aktuell zu bewältigende Aufgabe des Anwenders angepasst und somit der durch die Anwendung erbrachte Nutzen gesteigert werden.

Die Anzahl der Beispiele für kontextbezogene Anwendungen ist aufgrund der weit gefassten Definition im Prinzip unbegrenzt. Die automatische Anpassung der Spiegel- und Sitzeinstellungen an den aktuellen Fahrer in einem Auto ist genauso kontextbezogen wie die Empfehlung von Büchern in einer Online-Buchhandlung in Abhängigkeit der Einkaufshistorie eines Benutzers. Ein noch bekannteres Beispiel ist die von Suchmaschinen wie Google geschaltete Werbung in Abhängigkeit des aktuellen Suchbegriffs.

Um die Entwicklung von kontextbezogenen Anwendungen zu erleichtern, sollte eine Plattform verwendet werden, die wiederkehrende Aufgaben aus den Anwendungen auslagert und so den Entwicklern den Fokus auf die anwendungsspezifischen Aufgaben ermöglicht. Dey et al. haben in [DAS01] folgende Bedingungen identifiziert, die solch eine Plattform erfüllen sollte.

- **Trennung von Zuständigkeiten**: Idealerweise sollte sich der Anwendungsentwickler nicht mit den Details der Kontexterfassung, wie z. B. der Funktionsweise von Sensoren, auseinandersetzen. Durch eine Trennung der Zuständigkeiten sollte die Anwendung über definierte Schnittstellen mit dem kontexterfassenden Teil kommunizieren.

- **Kontextinterpretation**: Eine Anwendung sollte transparent auf komplexen Kontext zugreifen können. So kann es vorkommen, dass für eine Anwendung das Stattfinden eines Meetings von Interesse ist. Ob und in welchem Raum ein Meeting stattfindet, kann aus der Kombination von mehreren atomaren Kontextinformationen bestimmt werden. Hierzu könnte z. B. die Identität der sich im Raum befindenden Personen mit ihrem Terminkalender und der Lautstärke im Raum kombiniert werden. Durch die Interpretation dieser atomaren Informationen könnte dann eine abstraktere Kontextinformation generiert werden, die von mehreren Anwendungen verwendet werden kann. Die Verlagerung der Kontextinterpretation auf die Ebene der Plattform reduziert die Anwendungskomplexität, weil sich die Anwendungsentwickler nicht mehr darum kümmern müssen.

- **Transparente, verteilte Kommunikation**: Üblicherweise sind mehrere Sensoren für die Erfassung von Kontextinformationen zuständig. Diese Sensoren sind oft hochgradig in einer Umgebung verteilt. Die Verteilungsaspekte sollten sowohl für Anwendungen als auch für Sensoren transparent durch die Plattform verborgen werden. Sonst müssten sich Anwendungsentwickler um die Verteilung kümmern und dies würde zu komplexeren Anwendungen und Sensoren führen.

- **Permanente Verfügbarkeit der Kontexterfassung**: Weil *a priori* nicht vorhergesagt werden kann, wann Anwendungen auf Kontextinformationen zugreifen, müssen die kontexterfassenden Komponenten unabhängig von den Anwendungen jederzeit verfügbar sein. Umgekehrt kann der Ausfall der Kontextplattform die Funktionalität der Anwendungen negativ beeinflussen. Beispielsweise könnte die Telefonweiterleitung an das nächste Telefon in der Nähe des Benutzers nicht funktionieren, wenn die Anwendung nicht ermitteln kann, wo sich der angerufene Benutzer aktuell befindet.

- **Kontextspeicherung und -historie**: Jede kontexterfassende Komponente sollte eine Historie speichern. Dadurch ergeben sich weitere Möglichkeiten für Anwendungen wie die Auswertung von Statistiken und die Erstellung von Prognosen anhand dieser. Zudem könnten Anwendungen mit künstlicher Intelligenz das Verhalten von Benutzern lernen und sich besser ihren Gewohnheiten anpassen.

- **Ressourcenermittlung**: Schließlich sollte die Plattform Mechanismen anbieten, sodass Anwendungen sich nicht um die Auffindung von Sensoren kümmern müssen. Stattdessen sollten die Anwendungen spezifizieren können, welche Art von Kontext für sie von Interesse ist. Die Plattform gleicht dies dann mit den verfügbaren Sensoren ab und informiert die Anwendung entsprechend.

In dieser Arbeit werden als kontextbezogene Anwendungen eHome-Dienste betrachtet. Ein Teil des für personalisierbare Dienste relevanten Kontexts ist der *Personenkontext*, der hier von besonderem Interesse ist. Als Personenkontext können genau die Informationen aufgefasst werden, die weiter oben als Benutzerdaten bezeichnet wurden und durch ein Benutzermodell verwaltet werden (s. Kapitel 4). Dienste, deren Verhalten zumindest zum

Teil vom Personenkontext abhängt, werden als personalisierbare Dienste bezeichnet, wie in Abschnitt 2.2.4 erläutert wird.

2.1.3 Komponentenbasierte Softwareentwicklung

Befasste sich die Softwaretechnik anfangs noch größtenteils mit der Entwicklung von Programmen für einfache Probleme wie z. B. das Lösen numerischer Verfahren, die auf einem Rechner laufen sollten, besitzen heutige Softwaresysteme sehr komplexe und verteilte Strukturen. Diese Entwicklung formuliert Djkstra in [Dij72] folgendermaßen:

> "To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."

Die Entwicklung von komplexen monolithischen Softwaresystemen lässt sich heutzutage wenn überhaupt nur für Softwaresysteme rechtfertigen, die an spezifische Anforderungen eines Kunden angepasst sind (*Individualsoftware*). Einerseits ist die Entwicklung solcher Softwaresysteme sehr kostenintensiv, weil der Kunde die Gesamtkosten allein trägt. Andererseits sind ihre Wartung und Erweiterung mit großem Aufwand verbunden, wenn sich Anforderungen oder Systemumgebungen verändern.

Komplementär zur Individualsoftware ist die sogenannte *Standardsoftware*, die üblicherweise für einen Massenmarkt mit vielen Kunden entwickelt wird. Dadurch, dass mehrere Kunden die Kosten der Entwicklung gemeinsam tragen, kann die Software besonders günstig erworben werden. Der Hersteller sorgt gegebenenfalls für die Wartung und Weiterentwicklung der Software, deren neue Versionen den Kunden zugespielt werden können. Dem Vorteil der günstigen Kosten steht jedoch der Nachteil gegenüber, dass eine Anpassung an individuelle Kundenwünsche nicht möglich ist. Vielmehr müssen sich die Kunden an die Vorgaben der Standardlösung anpassen.

Einen Kompromiss zwischen diesen beiden Extremen Individual- und Standardsoftware stellt die *komponentenbasierte Softwareentwicklung* [Gri98] als Weiterentwicklung von modularen [Nag90] und objektorientierten Ansätzen dar. Das dabei verfolgte Ziel ist die Entwicklung von individuellen Softwaresystemen durch die „Komponierung und Konfigurierung“ [CH01] von vorgefertigten, wiederverwendbaren Softwarekomponenten.

Der Begriff der Softwarekomponente kommt in der Literatur vielfach mit verschiedenen Definitionen und unterschiedlichen Auffassungen vor [LW05]. Diese Unterschiede haben ihre Begründung darin, dass eine präzise Standarddefinition aufgrund der unterschiedlichen Anwendungsdomänen schwierig durchzusetzen ist. In der vorliegenden Arbeit werden die Definitionen von Councill und Heineman aus [CH01] verwendet:

Softwarekomponente Eine *Softwarekomponente* ist ein Stück Software, das konform zur Spezifikation eines Komponentenmodells ist und unabhängig von anderen Komponenten entwickelt und deployt werden kann. Weiterhin kann eine Komponente ohne Modifikation – entsprechend eines Kompositionsstandards – mit anderen Komponenten zusammengesetzt (komponiert) werden und mit diesen interagieren.

Komponentenmodell Ein *Komponentenmodell* definiert spezielle Interaktions- und Kompositionsstandards. Eine *Komponentenmodell-Implementation* ist eine Menge von geeigneten Softwareelementen, die nötig sind, um die Ausführung von Komponenten zu unterstützen, die entsprechend dem jeweiligen Komponentenmodell entwickelt wurden.

Der in obiger Definition vorkommende Begriff *Softwareelement* bezeichnet eine Sequenz von abstrakten Programmbefehlen, die entweder direkt oder erst nach Übersetzung in Maschinensprache, auf einem Rechner ausgeführt werden können. Mit *Deployment* wird der Vorgang beschrieben, in der fertige Komponenten in ihre Laufzeitumgebung eingebunden werden. Der *Kompositionsstandard* dagegen beschreibt, wie mehrere Komponenten zu einer größeren Struktur auf einer höheren Ebene zusammengesetzt oder wie alte Komponenten durch neue ersetzt werden können. Somit beschreibt der Kompositionsstandard den Kompositionsmechanismus eines Komponentenmodells. Ein grundlegendes Konzept von Komponenten sind klar definierte *Schnittstellen*. Der Aufbau und die Elemente einer Schnittstelle werden durch den *Interaktionsstandard* definiert.

Dadurch werden Implementierungsdetails der Komponente entsprechend dem *Geheimnisprinzip* (engl. Information Hiding, [Nag90]) gekapselt. Die Einhaltung dieses Prinzips gewährleistet größtmögliche Flexibilität eines komponentenbasiert erstellten Systems, weil Komponentenschnittstellen die Grundlage für die Interaktion mit anderen Komponenten und der Komponentenmodell-Implementation (auch *Komponentenplattform*) bilden. Zum einen muss eine Komponente Schnittstellen zur Interaktion mit der Komponentenplattform zur Verfügung stellen. Dadurch kann die Plattform den Lebenszyklus der Komponenten verwalten. Dazu zählen z. B. das Starten und Stoppen von Komponenten. Zum anderen findet die Komposition von Komponenten über ihre Schnittstellen statt. Dabei hat eine Komponente üblicherweise angebotene und benötigte Schnittstellen. *Angebotene* Schnittstellen werden von der Komponente realisiert und anderen Komponenten zur Verfügung gestellt. Oft muss zur Realisierung von angebotenen Schnittstellen auf andere Komponenten zurückgegriffen werden. Dies geschieht über den Import von *benötigten* Schnittstellen, die von anderen Komponenten realisiert werden. Die Komposition von Komponenten über Schnittstellen trägt signifikant zur Flexibilität ihrer Komponierbarkeit bei. Denn die Verwendung von Schnittstellen bietet die Grundlage für die nicht-invasive Austauschbarkeit von Komponenten, sofern sie die gleichen Schnittstellen anbieten. Umgekehrt erhöht die Austauschbarkeit von Komponenten die Flexibilität als auch Erweiterbarkeit eines Systems.

Bekannte Beispiele von Komponentenmodellen sind unter anderem das CORBA Component Model (CCM), die Enterprise Java Beans (EJB), das Common Object Model (COM+) und .NET sowie das im nächsten Abschnitt beschriebene OSGi.

2.1.4 Die OSGi-Service-Plattform

Die *OSGi Alliance* steht für eine Organisation bestehend aus mehr als 100 Unternehmen aus unterschiedlichen Branchen. Sie spezifiziert eine hersteller- und plattformunabhängige, dynamische *Dienstplattform* [WHKL08] auf Basis der Programmiersprache Java, die die

Modularisierung von Anwendungen konform zu einem Komponentenmodell ermöglicht. Inzwischen liegt die Spezifikation der *OSGi-Dienstplattform* (im Folgenden kurz OSGi) in der vierten Version [OSG09b] vor und wurde als „JSR 291: Dynamic Component Support for Java SE“ im Rahmen des Java Community Process (JCP) als offizielles dynamisches Komponentenmodell für Java angenommen.

War OSGi ursprünglich für den Einsatz in der Gebäudeautomatisierung vorgesehen, sind inzwischen weitere Einsatzgebiete vorhanden. Hierzu zählen insbesondere Bereiche wie *Automotive*, *mobile Endgeräte* oder *weitere eingebettete Systeme*, in denen Modularisierung, Verwaltung von Abhängigkeiten und dynamisches Deployment benötigt werden. Neuerdings wird OSGi auch im Desktopbereich eingesetzt. Ein prominentes Beispiel hierfür ist die integrierte Entwicklungsumgebung *Eclipse* [Ecl10a], die auf OSGi in Form des *Equinox*-Rahmenwerks aufsetzt. Auch in dieser Arbeit wird OSGi als Komponentenmodell für eHome-Dienste eingesetzt. Daher werden in diesem Abschnitt die Grundlagen von OSGi vorgestellt.

2.1.4.1 Struktur eines OSGi-Bundles

Komponenten in OSGi werden als *Bundles* bezeichnet. Sie haben die Form eines Java-Archivs (JAR-Datei), das neben Java-Klassen und einer Manifestdatei noch weitere Ressourcen wie Bilder oder HTML-Seiten beinhalteten kann [CG01]. Die Manifestdatei beschreibt den Inhalt der JAR-Datei und enthält Informationen über das Bundle [OSG09b]. Insbesondere werden in der Manifestdatei die Import- und Exportbeziehungen eines Bundles spezifiziert. Sie beschreiben die Java-Pakete, die von anderen Bundles importiert oder für andere Bundles exportiert werden. Über Schnittstellen, die in diesen Paketen enthalten sind, können Bundles zur Laufzeit dynamisch miteinander eine Verbindung herstellen und kommunizieren.

2.1.4.2 Lebenszyklus eines OSGi-Bundles

Ein Bundle kann nach seiner Installation auf der Dienstplattform mehrere Zustände durchlaufen. Eine Übersicht über die möglichen Zustände ist in Abbildung 2.1 in Form eines Lebenszyklus dargestellt. Nach der Installation nimmt ein Bundle zunächst den Zustand `INSTALLED` an. Es verbleibt in diesem Zustand, bis alle Importbeziehungen erfolgreich aufgelöst wurden, d. h., alle in der Manifestdatei importierten Bundles wurden gefunden. Erst dann geht ein Bundle in den Zustand `RESOLVED` über. Bundles müssen eine `Activator`-Klasse enthalten, die eine Start- und eine Stop-Methode enthält. Über diese Methoden kommuniziert die Dienstplattform mit dem Bundle. Wird die `Start`-Methode aufgerufen, geht das Bundle zunächst in einen Übergangszustand `STARTING` über, in der benötigte Ressourcen gebunden werden. Verläuft dieser Vorgang fehlerfrei, geht das Bundle in den Ausführungszustand `ACTIVE` über. In diesem Zustand kann ein Bundle Dienste bei der Dienstplattform registrieren oder deregistrieren. Durch den Aufruf der `Stop`-Methode werden zunächst gebundene Ressourcen im Zustand `STOPPING` freigegeben, anschließend geht das Bundle wieder in den Zustand `RESOLVED` über. In diesem Zustand können entweder Aktualisierungen des Bundles durchgeführt werden, bevor es neu gestartet wird, oder das Bundle wird deinstalliert und geht in den Zustand `UNINSTALLED` über.

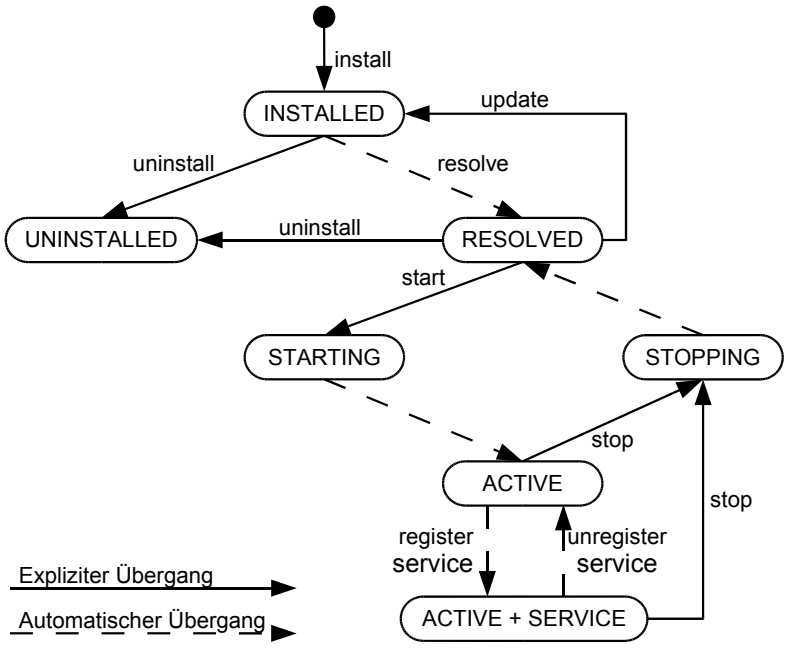


Abbildung 2.1: Lebenszyklus eines OGSi-Bundles (Quelle: [CG01]).

2.1.4.3 Aufbau der OSGi-Dienstplattform

Die Funktionalitäten und Richtlinien der Dienstplattform sind in verschiedene Schichten aufgeteilt, auf die sich Bundle-Entwickler stützen können [OSG09a]. Der Aufbau der Schichtenarchitektur ist in Abbildung 2.2 dargestellt. Die unterste Schicht bilden *Hardware und Betriebssystem*. Darauf setzt die *Ausführungsumgebung* in Form einer Java-Laufzeitumgebung (JVM) auf. Die eigentliche OSGi-Dienstplattform stellen die vier folgenden Schichten dar, die die von OSGi bereitgestellten *Basisdienste* implementieren:

- **Modulschicht**: Da die Java-Plattform nur eingeschränkte Unterstützung für das Packen, Deployen und Validieren von Java-basierten Komponenten anbietet, entgegnet dem die OSGi-Dienstplattform mit einem generischen, standardisierten Modulkonzept. Die oben beschriebenen Bundles ergänzen Java um ein Modularisierungskonzept. Die *Modulschicht* legt fest, wie die Bundles aufgebaut sein müssen, und ist für die Verwaltung der Bundle-Abhängigkeiten und deren Auflösung zuständig.

- **Lebenszyklusschicht**: Die *Lebenszyklusschicht* stellt eine Programmierschnittstelle (API) für Anwendungen zur Verfügung, über die die Lebenszyklus- und Sicherheitsoperationen eines Bundles gesteuert werden können. Die in Abbildung 2.1 dargestellten Zustände werden von dieser Schicht definiert und verwaltet.

- **Dienstschicht**: OSGi definiert ein Dienstmodell, das von der *Dienstschicht* umgesetzt wird. Bundles können ein oder mehrere Dienste bei der OSGi *Service Registry* anmelden, die dann von anderen Diensten aufgefunden und verwendet werden können. Das eng mit der Lebenszyklusschicht verknüpfte Dienstmodell stellt Mechanismen

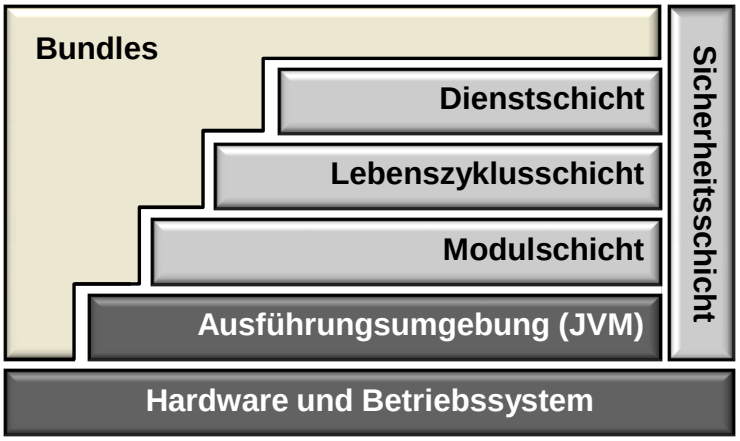


Abbildung 2.2: Schichtenarchitektur der OSGi-Dienstplattform (angelehnt an [OSG09a]).

bereit, die Bundles erlauben, einfacher mit der Laufzeitdynamik zurechtzukommen. Dadurch kann die Bindung von Diensten erst zur Laufzeit eines fertigen Systems erfolgen. Insofern geht das OSGi-Modell über einen rein komponentenbasierten Ansatz hinaus und setzt ein kooperatives, *serviceorientiertes* Modell um, das wegen seines Charakters auch als Find-And-Bind-Model bezeichnet wird. Der Dienst-Begriff geht im OSGi-Kontext jedoch kaum über den allgemeinen Schnittstellen-Begriff einer Komponente hinaus: Ein OSGi-Dienst ist im Prinzip ein einfaches Java-Objekt, das eine bestimmte Funktionalität umsetzt, die über eine Schnittstelle definiert und anderen Diensten bereitgestellt wird. Während das Paradigma der serviceorientierten Architekturen (SOA) Ortstransparenz und Zugriffstransparenz zur unternehmensweiten oder -übergreifenden Strukturierung von Systemlandschaften erfordert, unterstützt OSGi die Entwicklung verteilter Systeme nicht direkt. Es ist jedoch möglich, SOA-Systeme auf Basis von OSGi zu realisieren. Zusammenfassend kann festgehalten werden, dass sich OSGi zwischen komponentenbasierten und serviceorientierten Ansätzen befindet, aber durchaus für eigene Zwecke erweitert werden kann.

- **Sicherheitsschicht**: Die OSGi-*Sicherheitsschicht* stellt ein auf Java basierendes Sicherheitskonzept bereit, das im Wesentlichen aus der Java-2-Sicherheitsarchitektur besteht. Es bietet aber auch weitere Konzepte bzw. Mechanismen wie die sogenannten *Service-Permissions* oder *Package-Permissions* an. Durch Package-Permissions, die eine feingranulare Rechtevergabe ermöglichen, können Importe und Exporte für Bundles eingeschränkt werden. Durch Service-Permissions kann hingegen die Nutzung von Services eingeschränkt werden, sodass Bundles nur bestimmte Services anbieten oder nutzen können.

2.1.4.4 Eclipse Rich Client Platform

Eclipse ist eine frei verfügbare, in Java implementierte integrierte Entwicklungsumgebung (IDE). Die Eclipse-Open-Source-Gemeinschaft wird von der Eclipse Foundation [Ecl10a],

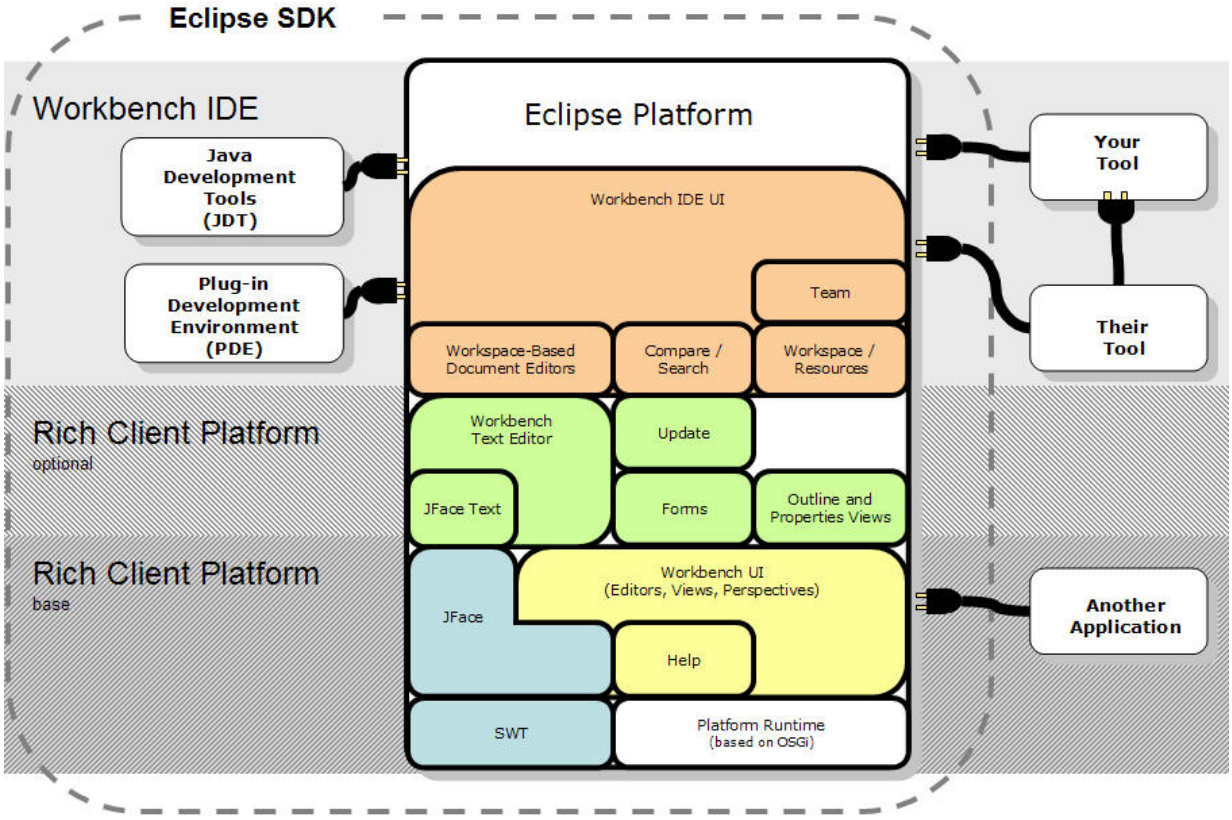


Abbildung 2.3: Die Eclipse-Architektur (Quelle: [SDF+04]).

einer gemeinnützigen Organisation mit mehreren prominenten Mitgliedern aus der Wirtschaft, geleitet. Obwohl Eclipse ursprünglich für die Java-Entwicklung vorgesehen war, ist es dank der Plug-In-Architektur sehr einfach für weitere Anwendungsmöglichkeiten erweiterbar. Beispielsweise existieren bereits diverse Plug-Ins für C++, Perl, PHP und weitere Programmiersprachen.

Das Eclipse-Plug-In-Komponentenmodell basiert seit der Version 3.0 auf einer Implementation einer OSGi-Dienstplattform namens Equinox [ML05, OSG09a]. Aus diesem Grund können dynamisch zur Laufzeit der Plattform Plug-Ins nachgeladen und entladen werden. Dabei besteht kein Unterschied zwischen den Konzepten der Eclipse-Plug-Ins und OSGi-Bundles. Diese Begriffe werden aus historischen Gründen synonym genutzt.

Abbildung 2.3 zeigt eine Übersicht über die Eclipse-Architektur, in die auch die OSGi-Dienstplattform eingebettet ist. Diese Architektur erlaubt die Interaktion und Kollaboration von Eclipse-Plug-Ins. Weil in Eclipse alles aus Plug-Ins besteht, gibt es eine Basismenge von Plug-Ins, die für die Entwicklung von Anwendungen notwendig sind. Diese Plug-Ins werden unter dem Begriff Eclipse *Rich Client Plattform* (RCP) zusammengefasst. Die RCP besteht aus der OSGi-implementierenden Laufzeitumgebung, SWT und JFace für die Entwicklung von grafischen Benutzerschnittstellen und der Workbench, die die Benutzerschnittstelle von Eclipse bereitstellt. Zusätzlich gibt es weitere optionale Komponenten, auf die bei der Entwicklung von Anwendungen zurückgegriffen werden kann, sodass sich ihre Entwicklungszeit

und ihr Entwicklungsaufwand verkürzen lassen. Plug-Ins sind in der Abbildung als „Tools“ gekennzeichnet.

2.2 Struktur von eHome-Systemen

In der vorliegenden Arbeit spielen intelligente, vernetzte Umgebungen eine zentrale Rolle. In der Literatur werden für solche Umgebungen Begriffe wie *Ambient Home*, *Smart Home*, *Intelligent Home* oder *Active Space* verwendet. Im Rahmen der Arbeiten im *eHome-Projekt* hat sich der *eHome*-Begriff etabliert. Obwohl diese Begriffe synonym verwendet werden können, wird in dieser Arbeit hauptsächlich auf den eHome-Begriff zurückgegriffen. Dadurch wird verdeutlicht, dass es sich bei den jeweiligen Ausführungen um Techniken, Konzepte und Bezeichnungen handelt, die dem eHome-Projekt zuzuordnen sind und in diesem Abschnitt einführend vorgestellt werden.

2.2.1 eHomes

Als eHome wird ein Haus oder eine Wohnung bezeichnet, in der elektronische, netzwerkfähige Geräte ganz oder teilweise über ein Netzwerk von einer zentralen Hardwareplattform, dem sogenannten *Residential-Gateway* [AMW99], gesteuert werden (s. Abbildung 2.4). Das Residential-Gateway kann beispielsweise ein gewöhnlicher PC sein und fungiert als zentrale Ausführungseinheit in einem eHome. Es realisiert die physikalische Anbindung der eingesetzten Kommunikationsprotokolle und stellt die Schnittstellen auf Hardwareebene bereit. In Abbildung 2.4 sind beispielsweise Kameras, Bewegungsmelder, Thermometer, Lampen usw. abgebildet, die über Protokolle wie X10 oder USB mit dem Residential-Gateway verbunden sind. Kirchhof definiert ein Residential-Gateway als [Kir05]:

> „[...] ein Rechner oder ein integriertes System, das mit Hardware-Schnittstellen zu unterschiedlichen physikalischen Medien und Netzwerktechnologien ausgestattet ist. Es leistet eine Protokollkonvertierung auf der Netzwerkschicht des ISO-Referenzmodells [ISO94].“

Die Menge von Kommunikationsprotokollen, die in den letzten Jahren hervorgekommen sind, ist vielfältig. Diese Menge kann wie in [Kir05] durchgeführt in zwei Kategorien unterteilt werden. Die erste Kategorie umfasst die *eHome-lokalen* Protokolle, zu denen unter anderem *HomePlug*, *HomePNA*, *HomeRF*, *IrDA*, *X.10*, *Bluetooth*, *CEBus*, *HAVi*, *LonWorks*, *EHS*, *EIB*, *Jini* und *UPnP* zählen. Diese Protokolle werden ausschließlich innerhalb eines eHomes eingesetzt. Die zweite Kategorie umfasst die *Weitbereichsprotokolle*, die für die Verbindung des eHomes an das Internet geeignet sind. Hierzu zählen *DSL*, *ISDN*, *analoge Telefonie*, *Kabel-Modems* und *Powerline*. Ein Residential-Gateway sollte in der Lage sein, eine geeignete Teilmenge von Protokollen aus beiden Kategorien an einem Punkt zusammenzuführen.

Offensichtlich sind diese Protokolle jedoch so heterogen, dass eine vereinheitlichte Kommunikation von Geräten und Diensten über unterschiedliche Protokolle ohne Weiteres nicht möglich ist. Dazu wird in eHomes eine „*Software-Abstraktionsschicht*“ auf Basis

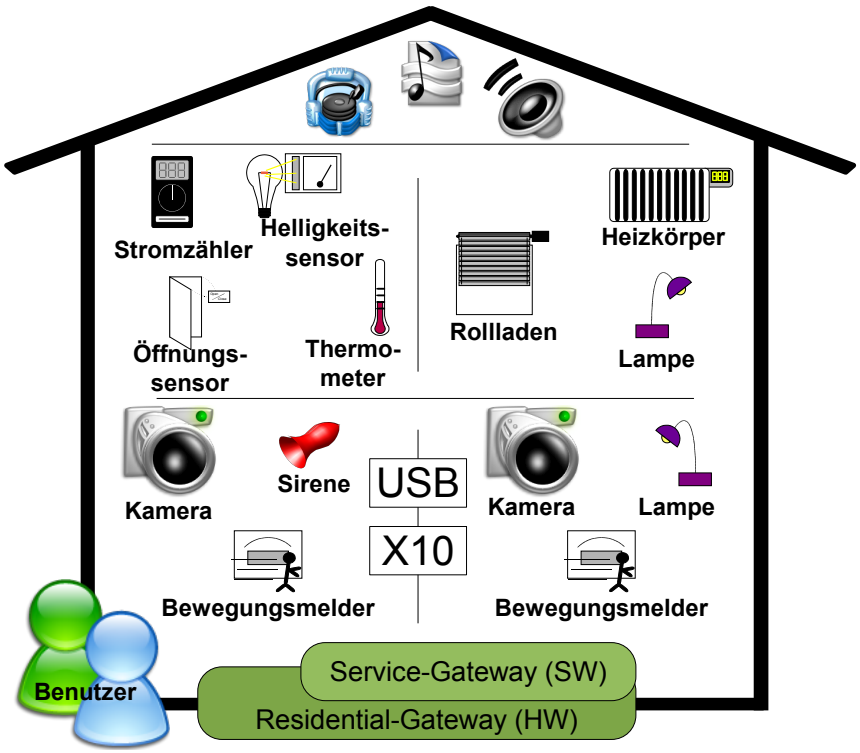


Abbildung 2.4: Beispielhafte Skizze eines eHomes.

des Residential-Gateway bereitstellt, die in Form einer Komponentenplattform (s. Abschnitt 2.1.3) Softwarekomponenten ausführen kann. Die relevanten Komponenten sind einerseits solche, die Softwareabbildungen von Kommunikationsinfrastrukturen und Geräteschnittstellen darstellen, und andererseits solche, die Dienstfunktionalitäten realisieren. Diese Abstraktionsschicht wird als *Service-Gateway* [Gon01a] bezeichnet und ermöglicht somit eine Koexistenz von unterschiedlichen Netzwerk- und Automatisierungstechnologien. Kirchhof definiert in [Kir05] ein Service-Gateway als

> „[...] eine Software-Infrastruktur auf Basis eines Residential-Gateway und bietet damit eine Ausführungsumgebung für Software-Komponenten an. Innerhalb dieser Umgebung werden Komponenten zur Adaption und Integration von Kommunikationsprotokollen ausgeführt sowie weitere Komponenten, die eHome-Dienste zur Erbringung von Nutzen für den Anwender realisieren.“

Obwohl beide Arten von Gateways unterschiedliche Aufgaben haben und Kirchhof die strikte Trennung zwischen Residential-Gateway und Service-Gateway empfiehlt, ist diese Trennung für die vorliegende Arbeit nicht wesentlich. Daher wird im Rest dieser Arbeit hauptsächlich der Begriff *Gateway* als Bezeichnung für die Kombination der beiden verwendet.

In [AKR07] wurde ein Ansatz zur Verteilung von Diensten innerhalb eines eHomes über mehrere Gateways vorgestellt. Dieser Ansatz ermöglicht die *transparente* Interaktion von Diensten über mehrere Gateways und wirkt Problemen wie der eingeschränkten

Konnektivität (physikalische Anschlüsse können nicht grenzenlos erweitert werden) eines einzelnen Gateways entgegen. Dabei wurde zwischen einem Haupt-Gateway und weiteren unterschieden. Das Haupt-Gateway muss dabei ständig in Betrieb sein, während die weiteren Gateways bedarfsorientiert betrieben werden können. In dieser Arbeit ist hauptsächlich die Interaktion von Benutzern mit dem Haupt-Gateway von Interesse, sodass die Verteilung innerhalb eines eHomes hier nicht weiter betrachtet wird.

Geräte können im Allgemeinen in Sensoren und Aktoren unterteilt werden. *Sensoren* sind solche Geräte, die Kontextinformationen wie Beleuchtungsintensität oder Temperatur erfassen. *Aktoren* dagegen verändern meist den Kontext. Beispielsweise verändert eine Heizung die Umgebungstemperatur und eine Lampe die Beleuchtungsintensität. In Abbildung 2.4 ist eine repräsentative aber nicht vollständige Menge von Geräten dargestellt, die verschiedenen Anwendungsdomänen zugeordnet werden können. Bewegungsmelder, Kameras, Sirenen und Lampen können etwa der Domäne der Sicherheitsanwendungen zugeordnet werden. Der Domäne der Verbrauchserfassung und -optimierung lassen sich Lampen, Heizungen, elektrische Rollläden, Thermometer, Beleuchtungssensoren, Tür- und Fensteröffnungssensoren und Strommessgeräte zuordnen. Lautsprecher und Fernsehgeräte lassen sich der Domäne Infotainment zuordnen.

2.2.2 eHome-Systeme

In [AK06, Kir05] werden sogenannte *eHome-Systeme* als Erweiterung von eHomes betrachtet. Erweiterung ist hier in dem Sinne zu verstehen, dass zusätzlich zu einem eHome noch weitere Akteure wie Benutzer und Anbieter betrachtet werden. Die *eHome-Benutzer*, im Folgenden abgekürzt als *Benutzer* (engl. *User*) bezeichnet, sind Akteure wie Bewohner, die sich in einem eHome aufhalten und Dienste in Anspruch nehmen. Die Interaktion mit den Diensten kann dabei entweder über Geräteschnittstellen oder über zusätzliche Interaktionsgeräte wie Displays oder PDAs geschehen. Der *Anbieter* (engl. *Provider*) ist verantwortlich für die Erbringung von verschiedenen Dienstleistungen. Hierzu gehören die Bereitstellung und der Betrieb eines eHomes genauso wie die Entwicklung und Installation von eHome-Diensten sowie die Versorgung der Kunden, also der Benutzer, mit weiteren Inhalten wie Nachrichten, Daten usw. Die Anbindung zwischen einem eHome und dem Anbieter wird über ein Weitbereichsnetzwerk realisiert, wobei auf eHome-Seite das Gateway den Kommunikationsendpunkt darstellt.

Die Prozesse, die zwischen Anbieter und Kunden stattfinden, spielen in dieser Arbeit keine wesentliche Rolle. Hier wird vielmehr angenommen, dass diese Prozesse wie in den obigen Arbeiten zufriedenstellend mit Werkzeugen unterstützt werden. Daher wird der Begriff eHome-Systeme im Rest der Arbeit selten verwendet. Stattdessen wird hier der Fokus auf den eHome-Begriff gelegt, weil die Interaktion von Benutzern mit eHomes und eHome-Diensten in dieser Arbeit von besonderem Interesse ist.

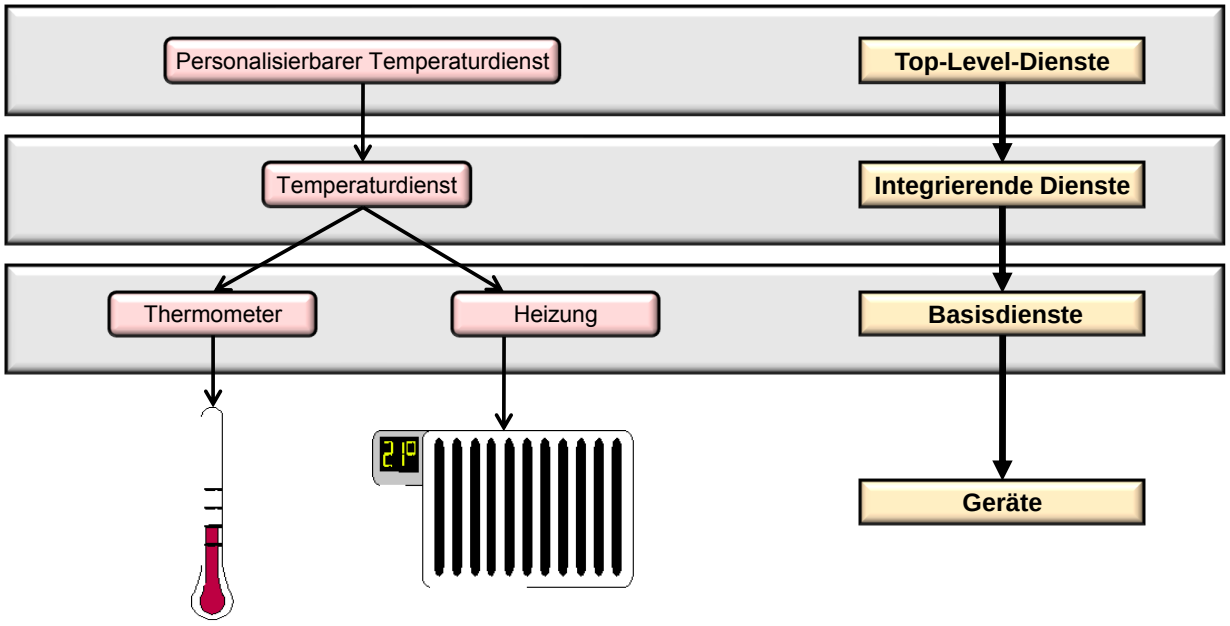


Abbildung 2.5: Hierarchische Dienstarchitektur am Beispiel des `Temperaturdienstes`.

2.2.3 eHome-Dienste

Die Vernetzung von Geräten in einem eHome allein reicht nicht, um Benutzern einen Mehrwert zu bieten. Die Vorteile eines eHomes kommen erst durch die Kombination einzelner Gerätefunktionalitäten zutage. Solch eine Kombination wird von komponentenbasierten, kontextbezogenen *eHome-Diensten* realisiert. In Anlehnung an die Vorgängerarbeiten im eHome-Projekt wird auch in dieser Arbeit eine hierarchische Dienstarchitektur verwendet, die in Abbildung 2.5 in Form einer Dienstkomposition (s. Abschnitt 2.1.3) beispielhaft dargestellt ist. Dabei werden drei Arten von Diensten unterschieden, die im Folgenden erläutert werden.

2.2.3.1 Basisdienste

Die unterste Schicht bilden Dienste, die hauptsächlich der Abstraktion der Geräte- und Kommunikationsinfrastruktur im eHome dienen. Die Dienste dieser Schicht werden als *Basisdienste* bezeichnet. Sie steuern meist Geräte (*Treiberdienste*) oder bieten Zugriff auf spezielle Kommunikationsprotokolle. Sie realisieren also die im letzten Abschnitt erwähnten Softwareabbildungen. Durch die Abstraktion von Geräte- und Protokollspezifika sind eHome-Dienste unabhängig von speziellen Herstellern oder Techniken. Auf diese Weise wird der Austausch von Geräten vereinfacht. Dies begünstigt die Erweiterbarkeit und Wartbarkeit von eHomes, zwei wichtige Qualitätseigenschaften, die von jedem Softwaresystem erfüllt werden sollten [Nag90].

Zwei Beispiele von Basisdiensten, die als Treiberdienste fungieren, sind in Abbildung 2.5 dargestellt. Der Dienst `Thermometer` liest über den Thermometer (Sensor) die aktuelle

Umgebungstemperatur und teilt es anderen Diensten mit. Der Dienst `Heizung` dagegen empfängt Anweisungen von anderen Diensten und regelt das Heizungsventil (Aktor) der entsprechenden Heizung.

Basisdienste unterscheiden sich von den beiden anderen Arten von Diensten insofern, dass sie für die Realisierung ihrer Funktionalitäten keine weiteren Dienste nutzen. Sie können ihre Funktionalitäten aber anderen Diensten zur Verfügung stellen und von diesen benutzt werden. Nach dieser Definition können in einem eHome auch weitere Basisdienste existieren, die nicht nur von Geräte- oder Protokollspezifika abstrahieren. Beispiele sind ein `Wetterdienst` bzw. ein `Routenplaner`, die – bis auf das Internet – auf keine anderen eHome-Dienste zugreifen, um Ihre Funktionalität zu realisieren.

2.2.3.2 Integrierende Dienste

Anders sieht es bei den *integrierenden* Diensten aus. Solche Dienste benutzen weitere Dienste (*Unterdienste*) zur Realisierung ihrer eigenen Funktionalitäten. Dadurch können sie neue, komplexere Funktionalitäten realisieren. Dabei können sie sowohl auf Basisdienste als auch auf weitere integrierende Dienste zurückgreifen. Infolgedessen kann sich die Schicht der integrierenden Dienste über mehrere Ebenen erstrecken. Um seine Unterdienste verwenden zu können, muss ein integrierender Dienst zuvor mit den Unterdiensten komponiert werden.

Integrierende Dienste werden oft auch als *Mehrwertdienste* bezeichnet [Kir05], weil durch die Kombination mehrerer Geräte ein Nutzen gewährt wird, der über das hinausgeht, was der Benutzer durch die getrennte Verwendung einzelner Geräte erzielen würde. Damit bilden Mehrwertdienste die Hauptmotivation für eHomes, weil erst mit ihnen das Potenzial einer vernetzten, automatisierten Umgebung ausgeschöpft werden kann.

Abbildung 2.5 zeigt als Beispiel den integrierenden `Temperaturdienst`. Dieser Dienst vergleicht anhand der von dem `Thermometer` gelieferten Kontextinformationen die aktuelle Raumtemperatur mit einem voreingestellten Wert. Weichen diese voneinander ab, wird die Heizung entsprechend geregelt. Es sei hier angemerkt, dass der `Temperaturdienst` ein kontextbezogener Dienst ist, weil sein Verhalten von der aktuellen Raumtemperatur beeinflusst wird. Umgekehrt verändert der Dienst durch seine Funktionalität auch den Umgebungskontext, weil er die Temperatur im Raum regeln kann.

2.2.3.3 Top-Level-Dienste

Die oberste Schicht der obigen Architektur wird aus den Diensten gebildet, die von keinem anderen Dienst mehr verwendet werden. Solche Dienste werden als *Top-Level-Dienste* bezeichnet. Ihre Funktionalitäten sind in einem eHome direkt wahrnehmbar, d. h., die Benutzer können diese Funktionalitäten direkt verwenden.

Ein Beispiel für ein Top-Level-Dienst ist der `Personalisierbare Temperaturdienst`, der die Funktionalität des integrierenden `Temperaturdienstes` so erweitert, dass die Raumtemperatur nicht mehr auf einen für den Raum spezifischen Wert angepasst wird, sondern an die persönlichen Präferenzen der sich im Raum aufhaltenden Benutzer. Nicht dargestellt ist die Art, wie der Dienst Benutzerpräferenzen, also den *Personenkontext* ermittelt. Diese kann er entweder durch Interaktion mit dem Benutzer oder aber über ein Benutzermodell

ermitteln. Beide Möglichkeiten werden im Rahmen dieser Arbeit unterstützt. Details dazu werden später in Kapitel 4 erläutert.

Dem Beispiel aus Abbildung 2.5 kann ferner entnommen werden, dass Top-Level-Dienste eigentlich auch integrierende Dienste sind, weil sie auch andere Dienste benutzen. Top-Level-Dienste unterscheiden sich von Integrierenden jedoch dadurch, dass integrierende Dienste von anderen Diensten verwendet werden, während das bei Top-Level-Diensten nicht der Fall ist. Eine Zuordnung von Diensten in eine der beiden Kategorien ist im Voraus nicht möglich, weil die *aktuelle* Dienstkomposition in einem eHome für diese Zuordnung entscheidend ist. Und die Kompositionen können von eHome zu eHome unterschiedlich sein. Angenommen, in einem eHome ist der `Personalisierbare Temperaturdienst` nicht installiert, sondern nur der `Temperaturdienst`. In diesem Fall würde der `Temperaturdienst` zu den Top-Level-Diensten gezählt werden. Also kann eine Zuordnung von Diensten zur Kategorie der Top-Level-Dienste erst zur Laufzeit eines eHomes erfolgen. Je nach Implementation können in seltenen Fällen auch Basisdienste zur Kategorie der Top-Level-Dienste zugeordnet werden, wenn ihre Funktionalitäten ohne weiteres von Benutzern genutzt werden können. Dieser Fall tritt jedoch sehr selten auf und wird daher in dieser Arbeit nicht betrachtet.

Während obige Klassifikation die in einem eHome installierten Dienste abhängig von der aktuellen Dienstkomposition zu einer der drei Kategorien zuordnet, existiert eine weitere Klassifizierung von eHome-Diensten, die im nächsten Abschnitt diskutiert wird.

2.2.4 Personalisierbare eHomes und eHome-Dienste

Eine weitere Möglichkeit der Unterteilung von eHome-Diensten ergibt sich durch Betrachtung der *Benutzeradaptivität* [Sch01] (s. auch Abschnitt 2.1.1). Benutzeradaptive Dienste, die ihre Funktionalitäten an ihre Benutzer anpassen können, werden in dieser Arbeit als *personalisierbar* bezeichnet. Sie unterscheiden sich von *nicht-personalisierbaren* Diensten dadurch, dass ihr Verhalten von Benutzerdaten beeinflusst werden kann, während dies bei nicht-personalisierbaren Diensten nicht der Fall ist.

Bereits in Abbildung 2.5 wurde mit dem `Personalisierbaren Temperaturdienst` ein Beispiel für einen personalisierbaren Dienst vorgestellt, der die aktuelle Raumtemperatur an die Präferenzen seines Benutzers anpassen kann. Weitere Beispiele für personalisierbare Dienste werden in Abschnitt 2.3 vorgestellt.

Ähnlich wie im obigen Beispiel können viele nicht-personalisierbare Dienste durch eine entsprechende Erweiterung ihrer Funktionalität in personalisierbare Dienste überführt werden. In welcher Art und Weise personalisierbare Dienste Benutzerdaten erlangen, ist momentan unbedeutend. Prinzipiell sind alle in Abschnitt 2.1.1 erwähnten Verfahren möglich. Der im Rahmen dieser Arbeit verfolgte Ansatz wird in Kapitel 4 vorgestellt.

In diesem Zusammenhang wird der Personalisierungsbegriff auch auf eHomes ausgeweitet. Ein eHome wird genau dann als personalisierbar bezeichnet, wenn es seinen Benutzern mindestens einen personalisierbaren Dienst anbietet. Weil die Personalisierung eines eHomes automatisch die Personalisierung mindestens eines Dienstes impliziert, wird in dieser Arbeit nur dann explizit zwischen der Personalisierung von Diensten oder eHomes unterschieden, wenn es an der jeweiligen Stelle für das Verständnis wichtig ist.

2.3 Anwendungsbeispiele für eHome-Dienste

In diesem Abschnitt werden Beispiele für Anwendungen (Top-Level-Dienste) beschrieben, die in eHomes einzeln oder zusammen zum Einsatz kommen können. Ihre Beschreibung soll einerseits einen Eindruck von möglichen Tagesabläufen in eHomes geben. Andererseits werden einige dieser Dienste im weiteren Verlauf der Arbeit zur Erläuterung der erarbeiteten Konzepte herangezogen.

Bei den folgenden Beispielen, von denen einige auch schon in der Motivation erwähnt wurden, handelt es sich mit Ausnahme des `Sicherheitsdienstes` um personalisierbare Dienste. Anders als im obigen Beispiel des `Personalisierbaren Temperaturdienstes` wird im weiteren Verlauf der Arbeit bei den Dienstnamen auf den Zusatz `personalisierbar` verzichtet, wenn dies aus dem Kontext hervorgeht.

2.3.1 Lichtdienst

Ähnlich wie der oben beschriebene *Personalisierbare Temperaturdienst* kann der `Lichtdienst` aus dem Anwendungsbereich Komfort den Umgebungskontext an die Präferenzen seiner Benutzer anpassen. In diesem Fall handelt es sich nicht um die Temperatur, sondern um die Beleuchtungsintensität. Eine mögliche Erweiterung dieses Dienstes ist die Anpassung der Lichtverhältnisse an die Aktivitäten der Benutzer. Beispielsweise sind beim Fernsehen andere Lichtverhältnisse erwünscht als beim Arbeiten. Eine weitere Variante dieses Dienstes ist die „mobile“ Version, die dem Benutzer von Raum zu Raum folgt. Begibt sich der Benutzer in einem anderen Raum, wird das Licht im alten Raum aus- und im neuen Raum, entsprechend der Präferenzen des Benutzers, angeschaltet. Offensichtlich können hier Konflikte auftreten, wenn mehrere Benutzer einen Raum gleichzeitig betreten. Das Problem der Benutzerkonflikte ist jedoch nicht Thema dieser Arbeit. Daher wird es hier nicht weiter betrachtet.

Dieser Dienst verwendet als Aktoren eine oder mehrere dimmbare Lampen. Als Sensor könnte ein Dreh- oder Schieberegler zum Einsatz kommen, mit dem sowohl die Funktionalität des Dienstes an- und ausgeschaltet als auch dem Dienst die gewünschte Beleuchtungsintensität mitgeteilt werden kann. Alternativ könnte der Dienst die Präferenzen des Benutzers auch anders ermitteln, beispielsweise über ein Benutzermodell (s. Abschnitt 2.1.1).

2.3.2 Musikdienst

Der `Musikdienst` gehört zum Anwendungsbereich Multimedia und spielt die Lieblingsmusik des Benutzers in dem Raum, in dem er sich aufhält. Er kann die Lautsprecher im eHome so steuern, dass die Musik dem Benutzer folgt, wenn er sich von einem Raum in einen anderen begibt. Er kann diese Funktionalität je nach Bedarf ein- und ausschalten.

Um seine Funktionalität realisieren zu können, benötigt dieser Dienst drei Arten von Benutzerdaten. Erstens, die Information, ob der Benutzer den Dienst momentan ein- oder ausgeschaltet hat. Zweitens, die Information über die Art der Musik, die der Benutzer hören möchte. Drittens, der Raum, in dem sich der Benutzer momentan befindet. Er muss immer

den aktuellen Stand dieser Daten kennen, um zeitnah auf Änderungen der Daten reagieren zu können.

2.3.3 Lifesytle-Dienst

Der `Lifesytle-Dienst` aus dem Anwendungsbereich Komfort ermöglicht seinen Benutzern die Definition unterschiedlicher Lifestyle-Profile, die Präferenzen bzgl. Beleuchtung, Temperatur und Musik enthalten. Diese Präferenzen können dann in bestimmten Situationen oder zu bestimmten Aktivitäten wie *Weckzeit*, *Party*, *Lesen*, *Fernsehen*, oder *Abwesend* aktiviert werden. Die Aktivierung kann entweder manuell durch den Benutzer erfolgen oder automatisch durch andere eHome-Dienste vorgenommen werden. Beispielsweise könnte der im Folgenden beschriebene Weckdienst die Präferenzen für die *Weckzeit* aktivieren.

2.3.4 Weckdienst

Die Hauptaufgabe des `Weckdienstes` aus dem Anwendungsbereich Komfort ist die Bestimmung der optimalen *Weckzeit* des Benutzers unter Berücksichtigung verschiedener Einflussfaktoren. Dazu gehören die benötigte Zeit zum Frühstücken und Duschen genauso wie die Uhrzeit des ersten Termins am Tag sowie die benötigte Fahrzeit zu diesem. Die Fahrzeit kann von den aktuellen Wetterverhältnissen (Schnee, Nebel usw.) oder von der Verkehrssituation auf der Standardroute (Stau, Umleitungen usw.) beeinflusst werden. Dieser Dienst hat mehr Funktionalitäten als nur die Berechnung der optimalen Weckzeit. Beispielsweise kann er abhängig von der Weckzeit das Badezimmer auf die gewünschte Temperatur vorheizen. Durch die Anpassung an die variable Weckzeit kann Energie gespart werden. Darüber hinaus kann der Benutzer mit seiner Lieblingsmusik und Licht aufgeweckt werden. Das automatische Hochfahren von Jalousien ist auch eine Möglichkeit. Eine etwas spielerische Erweiterung sieht die automatische Kaffeezubereitung zu vor.

Um diese Funktionalitäten zu realisieren, benötigt auch dieser Dienst Benutzerdaten. Neben den Präferenzen für Licht, Temperatur, Kaffee und Musik werden auch Informationen aus dem Terminkalender für die Uhrzeit und Ort des ersten Termins sowie Dusch- und Frühstücksdauer benötigt. Ferner muss dieser Dienst auch technische Informationen wie die Heiz- und Kaffeezubereitungsdauer berücksichtigen. Externe Kontextinformationen wie die aktuelle Verkehrssituation oder realistische Prognosen über die zu erwartende Verkehrssituation ergänzen die von dem Weckdienst einzuplanenden Informationen. Solche externen Informationen können über Basisdienste über das Internet ermittelt werden.

2.3.5 Klingeldienst

Ein innovativer Dienst aus den Anwendungsbereichen Komfort und Medizin ist der `Klingeldienst`. Dieser Dienst kann im Gegensatz zur klassischen Klingel für die Signalgebung nicht nur den Klingelsummer verwenden, sondern auch den Fernseher oder übliche Lautsprecher. Aber auch eine visuelle Signalgebung durch das Blinken der Lampen ist möglich. Dadurch, dass der Dienst personalisierbar ist, kann ein Benutzer einstellen, ob er lieber

visuell, akustisch oder kombiniert benachrichtigt werden möchte. Insbesondere für gehörlose oder sehbehinderte Benutzer bietet diese Art der Personalisierbarkeit große Vorteile. Die audiovisuelle Benachrichtigung von Benutzern ist nicht nur auf die Klingelfunktionalität beschränkt, sondern könnte auch für weitere Anwendungsfälle eingesetzt werden.

Die von diesem Dienst benötigten Benutzerdaten umfassen beispielsweise die Präferenzen bzgl. der präferierten Art der Signalgebung. Aber auch Informationen über den medizinischen Zustand des Benutzers können die Funktionalität dieses Dienstes beeinflussen.

2.3.6 Medizindienst

Ein Forschungsgebiet, das in den letzten Jahren an Bedeutung gewonnen hat, ist *eHealth* [AZA08, LSB09]. Dabei werden Möglichkeiten erforscht, Patienten durch Verwendung aktueller Technik im Alltag möglichst gut zu unterstützen. Die Anwendungsgebiete reichen von Krankenhäusern und Altenheimen bis hin zu privaten Wohnungen. Im häuslichen Umfeld könnte ein `Medizindienst` Vitalwerte von Benutzern wie Blutdruck, Puls und Atemfrequenz überwachen und in Notfallsituationen angemessene Aktionen ausführen. Beispielsweise könnten Bekannte oder medizinisches Personal benachrichtigt oder dem Benutzer Empfehlungen gegeben werden, sich zu beruhigen oder anders zu ernähren. Bei älteren Personen ist die Erkennung von Stürzen von besonderer Bedeutung, weil „beispielsweise im Alter von über 65 etwa ein Drittel der Männer und zwei Drittel der Frauen mindestens einmal im Jahr stürzt“ [LSB09]. Durch rechtzeitige und angemessene Reaktion auf solche Stürze können schwerwiegende Folgen gemildert werden. Auch die Erinnerung an und die Kontrolle der regelmäßigen Einnahme von Medikamenten kann mit heutiger Technik von einem `Medizindienst` durchgeführt werden.

Die von diesem Dienst benötigten Benutzerdaten umfassen neben den Vitalwerten der Patienten auch Informationen darüber, welche Aktionen in bestimmten Situationen ausgeführt werden sollen.

2.3.7 Sicherheitsdienst

Ein Beispiel für einen `Sicherheitsdienst` wurde in [Kir05] beschrieben. Das Ziel dieses Dienstes ist die kontinuierliche Überwachung eines eHomes. Wird eine Gefahrensituation wie Einbruch, Feuer oder Rohrbruch erkannt, informiert der Dienst die Bewohner und gibt optional Alarm. Im Prinzip handelt es sich um eine Erweiterung der klassischen Gebäudeüberwachung um Funktionalitäten, die über die pure Alarmierung hinaus gehen. Beispielsweise können Bewohner per SMS, MMS oder E-Mail über eine Situation informiert werden, wenn sich niemand zu Hause befindet. Es macht zudem Sinn, den Dienst vorsichtshalber auch nachts zu aktivieren.

Zur Erkennung von Gefahrensituationen werden verschiedene Sensoren wie Bewegungsmelder, Glasbruchsensoren, Tür- und Fensteröffnungssensoren, Rauchsensoren, Kameras usw. eingesetzt. Kameras können darüber hinaus zur Dokumentation der Situation in Form eines Videos verwendet werden. Alarm kann mit Sirenen gegeben werden. Vor einem sich androhenden Einbruch könnte über Lautsprecher auch Musik ausgegeben werden, um An-

wesenheit im eHome zu simulieren. Der Dienst könnte auch die Möglichkeit anbieten, über eine Webschnittstelle ausgeschaltet zu werden, wenn ein Alarm fälschlicherweise ausgegeben wurde. Die Erweiterungsmöglichkeiten sind vielfältig.

2.4 Digitale Identität und Identitätsmanagement

Dieser Abschnitt widmet sich der grundsätzlichen Frage. Es wird beispielhaft gezeigt, woraus eine Identität bestehen kann und in welchem Zusammenhang diese Begrifflichkeit Verwendung findet. Ferner wird auf die Verwaltung von Identitäten eingegangen und anschließend gezeigt, welche Bedeutung dem Schutz von Identitäten zukommt und wie dies gewährleistet werden kann.

2.4.1 Was ist eine Identität?

Die Beschäftigung mit der Frage, was eine Identität ist, führt zwangsläufig in viele unterschiedliche Wissensgebiete, angefangen mit der eigenen physischen Identität, die Identität im Recht und in der Mathematik bis hin zu philosophischen Betrachtungsweisen. Die Philosophie definiert den Begriff der Identität meist als Gleichheit, als eine vollkommene Übereinstimmung in allen denkbaren Eigenschaften. Ähnlich dazu ist die technische Sicht, die etwas als identisch bezeichnet, wenn es ein und das Gleiche ist. Den Zusammenhang eines Individuums mit sich selbst beschreibt die psychologische Betrachtungsweise. Gemeint ist hierbei die Summe der Merkmale, anhand derer ein Individuum von anderen unterschieden werden kann. Jede dieser wenigen Unterscheidungen führt zu vielen weiteren Fragestellungen. Beispielsweise bezeichnet die Logik zwei Dinge als identisch, wenn sie in allen Eigenschaften übereinstimmen. Im Falle einer Dame, die wir zum zweiten Mal an einem bestimmten Tag erblicken und behaupten, dieser heute schon einmal begegnet zu sein, gehen wir von vergleichbaren Äußerlichkeiten aus, wie z. B. Kleidung und Haarfarbe. Bei genauerer Betrachtung könnte man aber sehr wohl zu dem Schluss kommen, dass sich diese Eigenschaften der Äußerlichkeiten auch bei einem weiteren Individuum vorfinden lassen könnten (z. B. bei eineiigen Zwillingsschwestern). Das einzig zuverlässige Kriterium wäre demnach die kontinuierliche Existenz im Raum. Im Sinne der Logik wären die Damen nur identisch, wenn sie zur gleichen Zeit am gleichen Punkt des Raumes gewesen wären.

Dies zeigt die Komplexität des Themas und die Unmöglichkeit, diese zuerst recht simpel anmutende Frage nach der Identität in einer kurzen Einführung abschließend zu beantworten. Im Kontext dieser Arbeit wird mit Identität eine Menge von Attributen und den Ausprägungen (genauer: *Identitätsattribute*) bezeichnet, die gewisse *Eigenschaften* einer Person wiedergeben. Manche Eigenschaften wie Geburtsdatum und Augenfarbe haben für eine Person immer die gleiche Ausprägung. Andere Eigenschaften sind variabel, je nachdem welche Identität eine Person gerade einnimmt [PH09]. Ein Beispiel dafür ist die Adresse. Ist eine Person privat unterwegs, wird sie, wenn nötig, ihre Heimadresse angeben. Ist dieselbe Person geschäftlich unterwegs, kann sie auch ihre Firmenanschrift als Ausprägung ihre Adresse ausgeben.

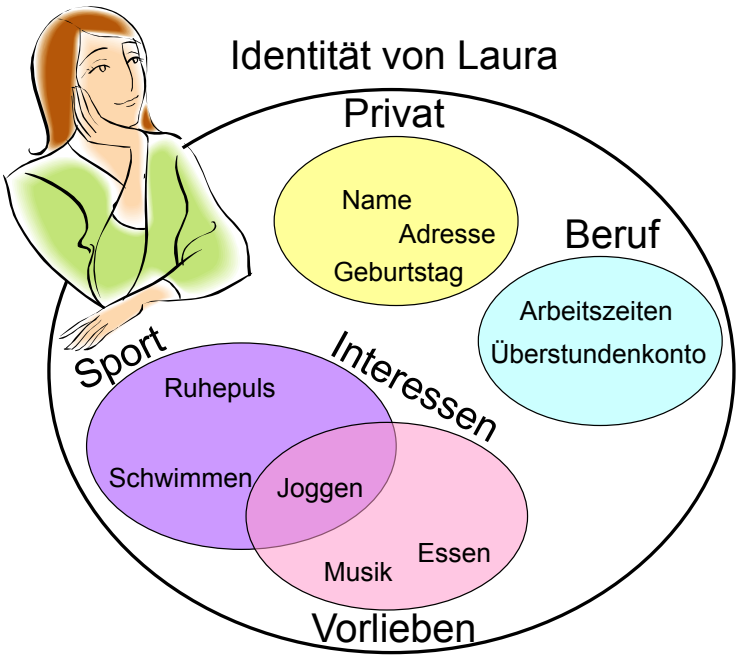


Abbildung 2.6: Beispiel für die Identität einer Person.

Attribute und -werte können sich dabei im Laufe der Zeit ändern. Da nie alle Eigenschaften einer Person erfasst werden können, sind Identitäten immer auf eine Menge von Eigenschaften reduziert, die im entsprechenden Kontext benötigt werden.

Abbildung 2.6 zeigt beispielhaft eine Identität einer bestimmten Person. Diese Identität besteht aus einer Menge von Eigenschaften wie z. B. ihrem Namen, ihrer Adresse, ihrer Arbeitszeiten usw. Weil diese Person in verschiedenen Kontexten nur unterschiedliche Eigenschaften von sich bekannt geben möchte, lässt sich diese Menge von Eigenschaften in Untermengen aufteilen, die wiederum verschiedene Identitäten darstellen. Dadurch kann eine Person unterschiedliche Identitäten (z. B. *beruflich*, *privat*) annehmen. In diesem Beispiel gibt es fünf Identitäten. Neben den Identitäten *Privat* und *Beruf* hat die Person noch die Identitäten *Sport* und *Vorlieben*, die sich in einem Attribut überschneiden und zusammen die Identität *Interessen* bilden.

Wenn im Umfeld einer elektronischen Speicherung und Verarbeitung Eigenschaften einer Person gespeichert werden, wird bei den daraus resultierenden Daten von *digitalen Identitäten* gesprochen, die in Abschnitt 2.4.2 näher beschrieben werden. Als Beispiel hierfür dient die vom Staat gegebene verwaltungstechnische Identität. Die Person wir hierbei auf einige Eigenschaften reduziert, die diese eindeutig identifizierbar machen sollen. Diese Identität wird der Person durch Abgabe eines Identitätsausweises z. B. in Form des Personalausweises gegeben [WM06, WJM+03].

Im Kontext von Computernetzen und elektronisch erfassten Identitäten ist eine Auswahl von Identitäten durch die Person denkbar. Bei der Wahl solch einer Identität wird auch von einer *Teilidentität* gesprochen. Das weiter unten beschriebene *Identitätsmanagement* impliziert die Möglichkeit einer Person, ihre Identität mit der sie ihrem Kommunikati-

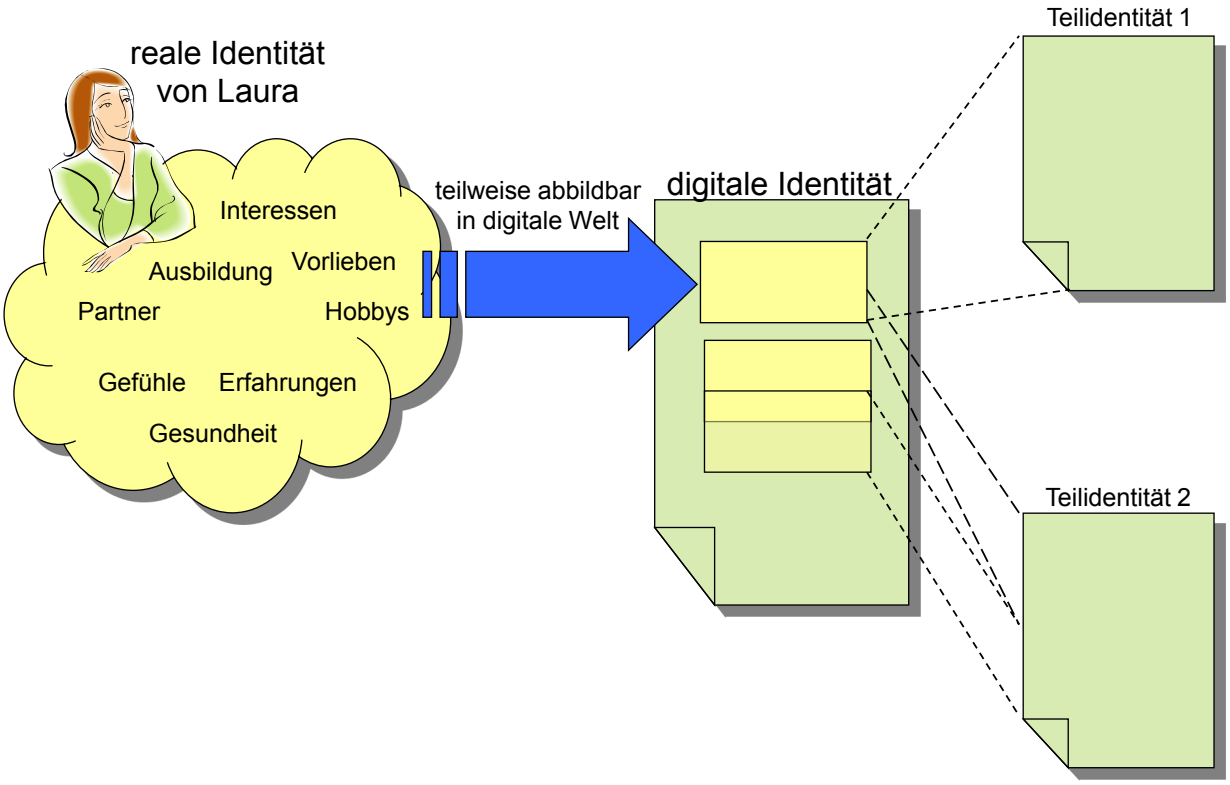


Abbildung 2.7: Abbildung realer Identität auf digitale Identität.

onspartner gegenüber auftreten möchte, zu wählen und zu verwalten. Mit der Wahl der Identität wird gleichzeitig eine Auswahl über die Eigenschaften der Person getroffen, die dem Kommunikationspartner gegenüber offenbart werden dürfen [DR06, JM01, Köh99].

2.4.2 Digitale Identitäten und Teilidentitäten

Eine digitale Identität ist eine *Abbildung* von Teilen der realen Identität einer Person in der realen Welt auf eine digitale Identität in der digitalen Welt. Damit wird nicht die gesamte reale Identität abgebildet, sondern nur ein bestimmter Teil wie z. B. Name, Adresse, Interessen usw. Eine digitale Identität kann elektronisch gespeichert und automatisch verarbeitet werden.

Dieser Zusammenhang wird in Abbildung 2.7 dargestellt. Dort ist die reale Identität der Person Laura zu sehen. Diese Identität, die aus einer Menge von Lauras Eigenschaften besteht, soll nun auf eine digitale Identität abgebildet werden. Eigenschaften wie *Gefühle*, *Erfahrungen* und *Gesundheit* können nicht abgebildet werden. Eigenschaften wie beispielsweise *Interessen* und *Ausbildung* können hingegen von der realen in die digitale Welt abgebildet werden. Eine digitale Identität ist also ein Teil einer realen Identität und kann bereits als eine *Teil-* oder *Pseudoidentität* bezeichnet werden.

Der primäre Sinn solcher Identitäten besteht darin, in bestimmten Situationen nur bestimmte Informationen bereitzustellen. Beispielsweise dienen ausschließlich die Attri-

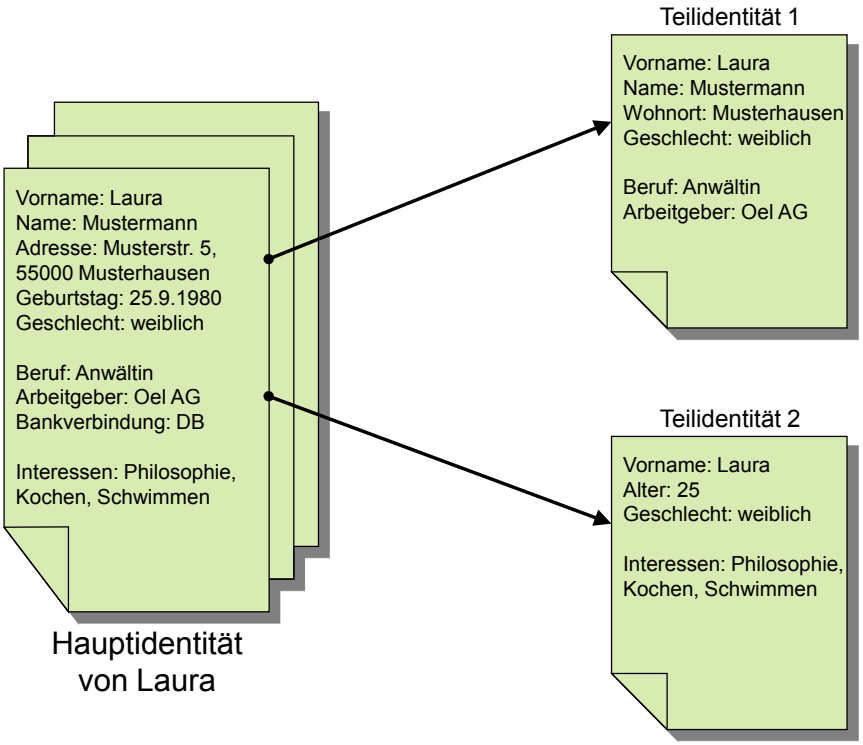


Abbildung 2.8: Beispiel für Teilidentitäten.

bute *Spitzname* und *Passwort* dazu, sich bei einem Betriebssystem zu *authentifizieren*. In einer anderen Situation könnte es ausreichen, lediglich die Telefonnummer und einige Interessengebiete anzugeben.

Abbildung 2.8 zeigt, wie eine digitale Identität aussehen könnte. Die reale Identität der Person Laura ist in eine digitale Identität abgebildet, die hier als *Hauptidentität* bezeichnet wird. Mit Hauptidentität ist die Menge *aller* abgebildeten Eigenschaften einer Person gemeint, im Gegensatz zu einer *Teilidentität*, die nur eine *Teilmenge* von Eigenschaften enthält. Auf die reale Identität der Person bezogen, stellt die Hauptidentität jedoch eine Teilidentität der realen Identität dar. Von dieser Hauptidentität aus lassen sich neue Teilidentitäten durch Auswahl einer Menge von Attributen abbilden. Im Beispiel sind es die Teilidentitäten *Teilidentität 1* und *Teilidentität 2*. Beide beinhalten eine Teilmenge der Attribute der *Hauptidentität* und dienen zur Repräsentation der Person in unterschiedlichen Kontexten.

Als Nächstes werden die Begriffe Anonymität und Pseudonymität eingeführt und mit Identitäten in Bezug gestellt.

2.4.3 Anonymität

Unter *Anonymität* (namenlos, aus dem Griechischen) wird die Geheimhaltung der Identität eines Subjekts verstanden. Dabei kann Subjekt eine natürliche Person (im Kontext dieser Arbeit interessant), eine juristische Person oder ein Computer sein. Mit anderen Worten bedeutet Anonymität, dass ein Angreifer ein Subjekt innerhalb einer Gruppe nicht hinrei-

chend identifizieren kann. Diese Gruppe wird als *Anonymitätsmenge* bezeichnet [PH09]. Je größer die Anonymitätsmenge ist und je ähnlicher das Verhalten der Mitglieder ist, desto stärker ist die Anonymität. Ferner ist in diesem Zusammenhang die *perfekte* Anonymität gegeben, wenn die Wahrscheinlichkeit, dass ein Gruppenmitglied eine bestimmte Aktion durchgeführt hat, für alle Mitglieder der Gruppe gleich ist [PH09, KP03, Ros03].

Anonymität kann in unterschiedlichen Stärken realisiert werden. Flinn und Maurer haben in [FM95] folgende Stufen von Anonymität beschrieben, in denen ein Benutzer unterschiedlich anonym ist:

- **Level 5: Eindeutige Identifikation (super identification)**: In diesem Fall muss der Benutzer eindeutig identifiziert werden. Es darf keine Möglichkeit bestehen, dass Benutzer verwechselt werden oder sich als andere Personen ausgeben. Eine solche Identifikation ist bei sensiblen Aktionen wie Bankgeschäften nötig. Beide Partner einer Kommunikation müssen sicherstellen können, dass der Gegenüber derjenige ist, für den er sich ausgibt.

- **Level 4: Normale Identifikation (usual identification)**: Der Benutzer weist seine Identität durch den Nachweis der Kenntnis eines Geheimnisses, beispielsweise ein Passwort, nach. Dies ist der normale Weg bei der Anmeldung an einem Betriebssystem.

- **Level 3: Mögliche Identifikation (latent idenfitication)**: Jeder Benutzer ist dem System über Pseudonyme bekannt. Die Pseudonyme sind einmalig und werden nur von einer Person benutzt. Nutzer können andere Benutzer nicht identifizieren, obwohl das System die Identität zu jedem Pseudonym kennt. Auf diese Weise funktionieren beispielsweise Chiffreanzeigen.

- **Level 2: Pseudonymität (pen-name identification)**: Ein Benutzer verwendet ein oder mehrere Pseudonyme (Künstlername) und die dazugehörigen Passwörter, um sich zu authentifizieren. Im Gegensatz zum Level 3 kennt das System aber nicht die wahre Identität des Benutzers. E-Mails-Adressen können beispielsweise zu diesem Zweck eingesetzt werden.

- **Level 1: Anonyme Identifikation (anonymous identification)**: Ein Benutzer meldet sich anonym bei einem System an. Das System kennt den Benutzer aber nicht als wieder erkennbares Individuum. Der Nutzer ist damit nicht adressierbar, kann aber dennoch von anderen anonymen Nutzern innerhalb einer Sitzung unterschieden werden. Dadurch können die Aktionen stets dem richtigen Benutzer zugeordnet werden. Sobald er sich abmeldet, kann er nicht mehr kontaktiert werden.

- **Level 0: Keine Identifikation (no identification)**: In diesem Level ist ein Benutzer dem System nicht bekannt. Dies ist der Fall bei öffentlich zugänglichen Computern, etwa in einer Bibliothek. Es können jedoch intelligente Programme auf dem PC vorhanden sein, die Historien von Benutzeraktionen aufzeichnen, sodass daraus zusätzliche Informationen über Benutzerverhalten abgeleitet werden können. In diesem Zusammenhang kann von echter Anonymität erst dann gesprochen werden, wenn die Existenz von Aufzeichnungen über Benutzeraktionen ausgeschlossen werden kann.

Diese Unterteilung impliziert schon, dass Pseudonyme eine wichtige Rolle bei der Realisierung von Anonymität spielen. Daher werden sie im Folgenden genauer beschrieben.

2.4.4 Pseudonymität

Verwandt mit der Anonymität ist die *Pseudonymität*, die auf die Verwendung von Pseudonymen hinweist. Ein *Pseudonym* ist dabei ein Bezeichner für die Identifikation einer Person, der sich von dem echten Namen (bürgerlicher Name) der Person unterscheidet [PH09]. Es ermöglicht die Zuordnung und Verkettung (s.u.) von Handlungen einer bestimmten Person, ohne ihre wahre Identität bekannt geben zu müssen. Eine Person kann auch mehrere Pseudonyme verwenden, deren Zusammenhang für Außenstehende nicht erkennbar ist.

Pseudonyme sind deshalb so interessant, weil sie eine Betrachtung von Anonymität in verschiedenen Ausprägungen unterstützen. Während nämlich Anonymität und Zurechenbarkeit (s. Abschnitt 2.5.1) zwei Extreme der Verkettbarkeit von Handlungen von Personen bilden, deckt Pseudonymität die gesamte Palette zwischen diesen beiden Extremen ab. In diesem Zusammenhang lassen sich unterschiedliche Klassen von Pseudonymen unterscheiden (s. [KP03, PH09]):

- **Personenpseudonym**: Ein Personenpseudonym ist ein Substitut für den bürgerlichen Namen einer Person und kann über lange Zeit hinweg für viele Geschäftsbeziehungen verwendet werden. Sie bieten die größte Verkettbarkeit von Aktionen in unterschiedlichen Kontexten. Beispiele hierfür sind die Sozialversicherungsnummer, die Telefonnummer oder die E-Mail-Adresse.

- **Rollenpseudonym**: Die Verwendung eines Rollenpseudonyms ist dagegen auf spezifische Rollen beschränkt. Solch ein Pseudonym wählt die Person für die Kommunikation mit ihrer Umwelt immer dann, wenn sie sich in der zugehörigen Rolle befindet. Ein berühmtes Beispiel hierfür ist der Künstlername, den Buchautoren verwenden, um nicht ihre wahre Identität preis zu geben.

- **Beziehungspseudonym**: Während obige Pseudonyme für die Kommunikation mit mehreren Partnern genutzt werden können, wird ein Beziehungspseudonym immer dann gewählt, wenn mit demselben Partner kommuniziert wird. Die Verwendung solch eines Pseudonyms ist unabhängig von der gewählten Rolle. Beispielsweise kann eine Person bei der Deutschen Bahn online mit dem gleichen Nicknamen Fahrkarten sowohl für private als auch für geschäftliche Zwecke erwerben.

- **Rollenbeziehungspseudonym**: Sollen jedoch für jede Rolle unterschiedliche Pseudonyme eingesetzt werden, kommen Rollenbeziehungspseudonyme in Spiel. Hier wird für jeden Kommunikationspartner und jede Rolle ein eigenes Pseudonym verwendet. Somit kann der Kommunikationspartner nicht herausfinden, ob zwei unterschiedliche Pseudonyme zum gleichen Benutzer gehören.

- **Transaktionspseudonym**: Soll die mögliche Verkettbarkeit noch weiter eingeschränkt werden, kann für jede Transaktion ein anderes Pseudonym verwendet werden. Bei-

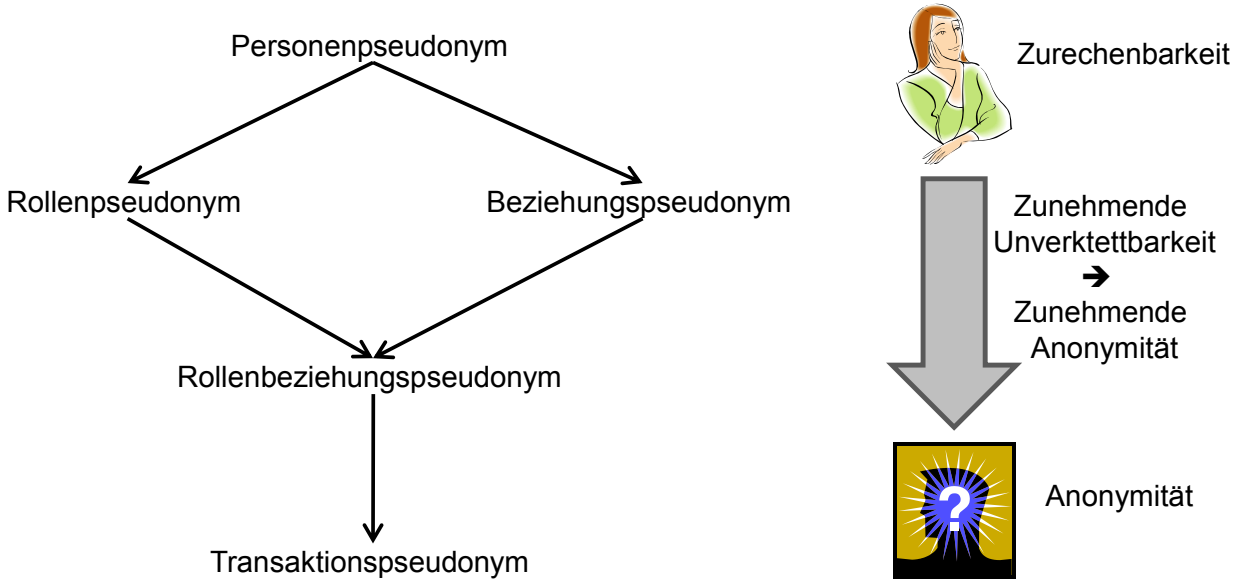


Abbildung 2.9: Unterschiedliche Arten von Pseudonymen und ihr Zusammenhang zur Anonymität (angelehnt an [KP03]).

spielsweise können Zufallszahlen mit Transaktionen im Onlinebanking verknüpft werden.

Die Stärke der Anonymität, die durch die obigen Klassen der Pseudonymität erreicht werden können, ist in Abbildung 2.9. Dabei bedeutet $A \rightarrow B$, dass B eine stärkere Anonymität ermöglicht als A. Also bieten Rollen- und Beziehungspseudonyme eine stärkere Anonymität als Personenpseudonyme. Werden Rollenbeziehungspseudonyme eingesetzt, kann die Anonymität noch mal verstärkt werden. Die stärkste Anonymität bieten Transaktionspseudonyme, vorausgesetzt, es können nicht anderweitig, beispielsweise über die Transaktion selbst, Informationen über die Person gesammelt werden.

Wenn ein Pseudonym einer bestimmten Person nicht eindeutig zuzuordnen ist, so ist damit nicht zwangsläufig die *Aufdeckbarkeit* dieser Person, also deren Identifizierung, zwingend verhindert. Ein Problem ist das Hinterlassen von Spuren mit Pseudonymen und die etwaige *Verkettbarkeit* dieser Pseudonyme untereinander. Abbildung 2.10 zeigt solch einen Fall. Die *Hauptidentität* von Laura kann eindeutig einer Person zugeordnet werden. Mit dieser Identität ist die Person eindeutig identifiziert, da dort der vollständige Name und die Adresse enthalten sind. Von dieser Identität sind zwei weitere Teilidentitäten abgebildet worden, die *Teilidentität 1* und *Teilidentität 2*. Beide Teilidentitäten für sich genommen lassen zunächst keine eindeutige Identifizierung der Person zu. Aber was passiert, wenn eine Beziehung zwischen diesen Teilidentitäten gefunden werden kann? Angenommen, Laura ist in zwei unterschiedlichen Institutionen mit *Teilidentität 1* und *Teilidentität 2* unterwegs gewesen. Weiter sei angenommen, dass diese beiden Institutionen zusammenarbeiten und ihre Daten abgleichen. Durch Vergleich der einzelnen Attribute der beiden Teilidentitäten lässt sich ein Zusammenhang zwischen ihnen herstellen. In beiden findet sich für die Attribute

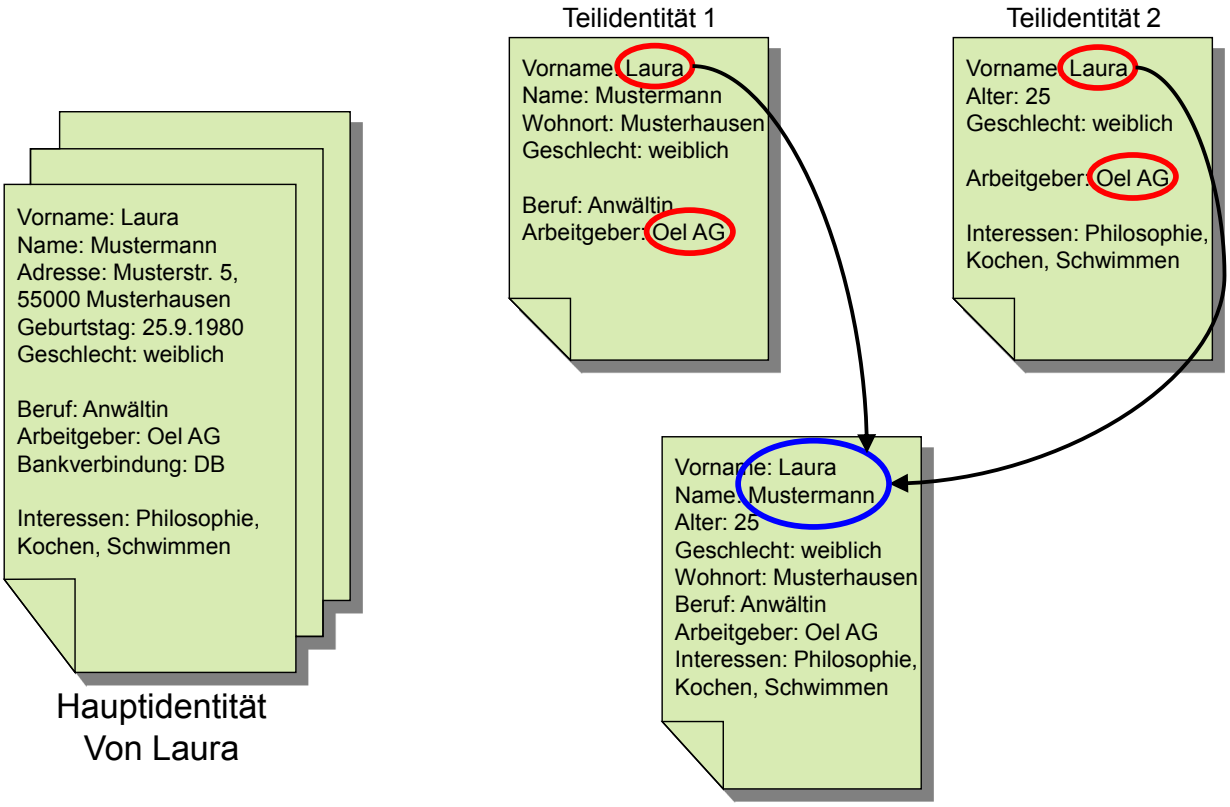


Abbildung 2.10: Beispiel für die Verkettbarkeit von Teilidentitäten.

`Vorname` und `Arbeitgeber` als Ausprägungen `Laura` und `Oel AG`. Die Wahrscheinlichkeit ist damit sehr hoch, dass beide Pseudonyme von der gleichen Person genutzt werden. Beide Identitäten sind somit verkettbar geworden. Die Institutionen können nun ihre Daten zusammenfügen, welches eine zusätzliche beidseitige Informationsgewinnung nach sich zieht.

2.4.5 Identitätsmanagement

Wenn eine Person verschiedenen Kommunikationspartnern gegenüber mit unterschiedlichen Teilidentitäten auftritt, kann sie durch *Identitätsmanagement* unterstützt werden. Konkret dient Identitätsmanagement der Pflege von Identitätsattributen (Erstellen, Löschen, Ändern) und der Verwaltung von Teilidentitäten. Insbesondere gehört hierzu die Zuordnung von Attributen und Pseudonymen zu Teilidentitäten sowie die Auswahl einer Teilidentität für die Kommunikation mit dem aktuellen Kommunikationspartner. Dadurch behält der Benutzer die *Kontrolle* über die Preisgabe seiner Daten, sodass er von seinem *Recht auf informationelle Selbstbestimmung* Gebrauch machen kann. Mit *Identitätsmanagementsystemen* hingegen lässt sich das Recht auf informationelle Selbstbestimmung weitgehend *technisch* in die digitale Welt abbilden [MJM01a, HB03, PH09].

Bereits in den 80er Jahren wurden die ersten Vorschläge zu einem technisch basierten Identitätsmanagement gemacht. Seitdem werden immer wieder neue Ansätze für generelle

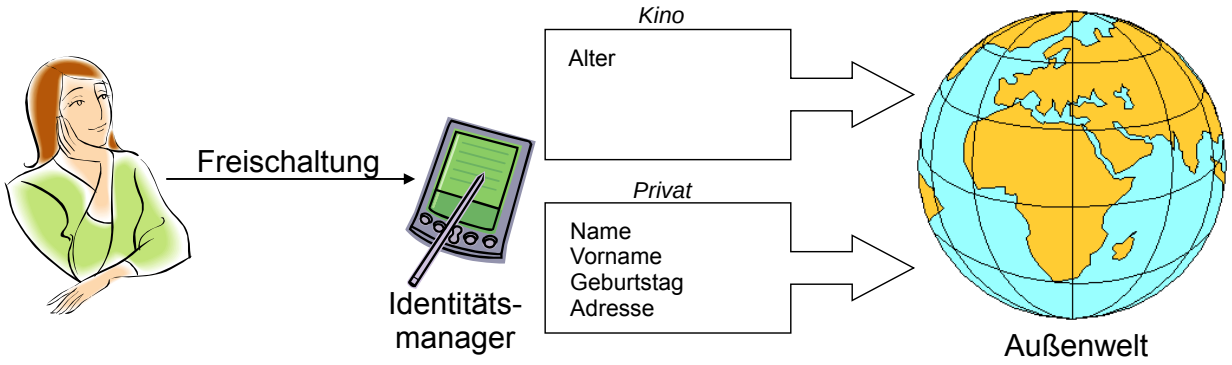


Abbildung 2.11: Beispiel für Identitätsmanagement.

Modelle erstellt, und es finden sich mittlerweile eine Reihe von Implementationen, da die Verwaltung von Identitäten gerade im Bereich des *Datenschutzes* und der *Privatsphäre* eine zunehmende Rolle spielt [HKRG03, Ros04, MJ01, CH02]. Ein umfassendes Identitätsmanagementsystem lässt den Nutzer steuern, welche Kommunikationspartner welche Informationen erhalten und wie diese verwendet werden dürfen. Im Sinne *mehrseitiger Sicherheit* bedeutet dies gegebenenfalls die *Aushandlung* der Informationsweitergabe, z. B. Umfang und Art der Daten und Bedingungen an die Verarbeitung dieser Daten. Wo eine automatische Steuerung für die Herausgabe von Informationen durch das Identitätsmanagementsystem nicht möglich ist, soll das System dem Nutzer darlegen, wer welche Informationen wofür haben möchte und eine Bestätigung oder Ablehnung erfragen.

Das Kernstück für Identitätsmanagement bilden Verfahren zur Gewährleistung von Anonymität und Pseudonymität, die weiter oben erläutert wurden [PBP06, Bra04, HR03]. *Methoden des Identitätsmanagements*, die zum Selbstdatenschutz der Person eingesetzt werden können, sind:

- **Aushandlungskomponenten** für Sicherheitsfunktionen eines Kommunikationsnetzes, das Sicherheitsfunktionen einschließlich Anonymität bereitstellt.
- **Regelbasierte Bestimmung** über die Herausgabe von Benutzerdaten und über die Anforderungen an die Verwendung dieser Daten.
- **Funktionen für Einwilligung, Widerspruch, Auskunft, Berechtigung** oder **Löschung** der eigenen Daten beim Verarbeiter.
- **Nutzerseitige Unterstützung bei der Pseudonym- und Benutzerdatenverwaltung** durch Speichern und Darstellen von Kontext und Kontextwechseln, z. B. durch Protokollierung der Daten und der Kontexte, in denen sie herausgegeben wurden (z. B. Zeitpunkt oder Kommunikationspartner).

Abbildung 2.11 verdeutlicht, wie Identitätsmanagement in der Praxis aussehen könnte. Eine Person bewegt sich innerhalb der realen Welt mit einem Handheld, auf welchem

sich ein Identitätsmanagementsystem befindet. Dieser verwaltet und schützt die auf dem Handheld mitgeführten Benutzerdaten. Bei Eintritt in eine Umgebung, die persönliche Daten benötigt, kann die Person über ihr Handheld eine Teilidentität auswählen. In der Abbildung sind exemplarisch zwei Teilidentitäten aufgeführt. Die Teilidentität *Kino* stellt nur die Information `Alter` zur Verfügung, wohingegen die Teilidentität *Privat* mit wesentlich mehr Informationen aufwartet.

2.5 Datenschutz und Privatsphäre

Privatsphäre wird häufig mit Datenschutz verwechselt. Daher wird im Folgenden näher auf diese beiden Begrifflichkeiten und deren Bedeutung eingegangen. Darüber hinaus werden Charakteristika von Schutzzielen für mehrseitige Sicherheit in Hinblick auf die Privatsphäre untersucht. Damit wird das Ziel verfolgt, Anforderungen an ein technisches System für Privatsphäre und Zugriffskontrolle zu erarbeiten.

Datenschutz meint den Schutz persönlicher Belange bzw. des *Persönlichkeitsrechts* und das Recht auf *informationelle Selbstbestimmung* (s. auch Abschnitt 1.3.4). Es werden also die Rechte an den Daten geschützt und nicht die Daten selbst [Roß07, FP01]. Das Recht auf informationelle Selbstbestimmung kann aus dem Grundgesetz abgeleitet werden:

> „Jeder hat das Recht auf die freie Entfaltung seiner Persönlichkeit, soweit er nicht die Rechte anderer verletzt und nicht gegen die verfassungsmäßige Ordnung oder das Sittengesetz verstößt.“[1]

Der Ursprung dieses Gesetzes stammt aus dem Jahre 1983 und ist eine Folge der damaligen Volkszählung. Seitens der Bevölkerung gab es große Bedenken, ob die erhobenen Daten nicht noch anderen Zwecken als dem eigentlichen Zweck der Volkszählung zugeführt würden. Die Bevölkerung fürchtete sich vor Missbrauch ihrer persönlichen Daten, und nach vielen Demonstrationen schlug sich dies in dem sogenannten Volkszählungsurteil (s.u.) und dem daraus resultierendem Gesetz nieder. Das Recht auf informationelle Selbstbestimmung gewann durch dieses neue Gesetz erheblich an Bedeutung.

Bei der *Kommunikation in digitalen Netzen* spielt der Datenschutz eine erhebliche Rolle. In solchen Netzen ist es viel leichter, personenbezogene Daten zu sammeln und zu verarbeiten als es in der realen (analogen) Welt möglich wäre. Die Gefahr eines Missbrauchs, der Daten ist hier wesentlich größer. Daher muss in den digitalen Welten um so mehr auf Datenschutz geachtet werden [Lan05, Lan02, Lan01b, Pfi06a].

Nach dem Bundesdatenschutzgesetz besteht die Verpflichtung, auf *Datensparsamkeit* zu achten. Dies bedeutet, dass nicht mehr Daten gesammelt werden als nötig. Dieser Grundsatz schlägt sich im folgenden Gesetz nieder:

> „Gestaltung und Auswahl von Datenverarbeitungssystemen haben sich an dem Ziel auszurichten, keine oder so wenig personenbezogene Daten wie möglich zu

[1] Nachteilsverbot: §9.Abs. 1 Satz 2 VZG 1983.

erheben, zu verarbeiten oder zu nutzen. Insbesondere ist von den Möglichkeiten der Anonymisierung und Pseudonymisierung Gebrauch zu machen, soweit dies möglich ist und der Aufwand in einem angemessenen Verhältnis zu dem angestrebten Schutzzweck steht.“[2]

In der Realität sieht es mit der Berücksichtigung dieses Gesetzes allerdings nicht so gut aus. Dies liegt vor allem daran, dass Dienstleister damit argumentieren können, dass eine umfangreichere Datenerhebung eine Dienstverbesserung für den Kunden nach sich zieht. Dies kann sogar soweit gehen, dass intelligente Umgebungen wie eHomes sogar Informationen über die sozialen Beziehungen von Benutzern sammeln, um ihre Funktionalitäten besser den Bedürfnissen der Benutzer anzupassen [ZO05].

Privatsphäre meint etwas anderes als Datenschutz, auch wenn Privatsphäre eng mit dem Datenschutz zusammenhängt. Unter Privatsphäre ist zuerst einmal der Bereich zu verstehen, in dem Privatpersonen unbeobachtet leben, handeln und kommunizieren können. Eine Privatperson soll in ihrem privaten Bereich selbst darüber entscheiden dürfen, wer sie sieht und wer mit ihr in eine Kommunikationsbeziehung eintreten darf [PH09, Roß07].

Bei der Privatsphäre sollte daher im Vordergrund stehen, dass eine Person nicht automatisch unsichtbar wird, sondern die Kontrolle darüber behält, wer welche Eigenschaften von ihr sehen soll. Die Unterscheidung zwischen Privatsphäre und Datenschutz sollte besonders bei der Entwicklung von Identitätsmanagementsystemen Beachtung finden. Wichtig ist, dass die Person *jederzeit überwachen und bestimmen kann*, welche persönlichen Daten wie verwendet werden und keine Informationen ohne ihr Einverständnis weitergegeben werden. Dies macht die Verwaltung von Daten notwendig [Rod03, NWET04, Chr04].

Um konkrete Anforderungen für Privatsphäre in digitalen Netzen zu entwickeln, können die im folgenden Abschnitt aufgeführten *Schutzziele* herangezogen werden.

2.5.1 Schutzziele mehrseitiger Sicherheit und Privatsphäre

Unter mehrseitiger Sicherheit wird die Berücksichtigung der Sicherheitsanforderungen *aller* an einer Kommunikation beteiligten Parteien, insbesondere auch der Benutzer eines Systems, verstanden [Pfi06b, Gol07]. Die Kommunikationspartner können verschiedene Interessen verfolgen und ein unterschiedliches Interesse an Privatsphäre und Sicherheit aufweisen. Daher muss ein Abgleich dieser eventuell konkurrierenden Interessen vorgenommen werden. Beispielsweise möchte eine eHome möglichst viel über ihre Benutzer erfahren, während diese möglichst wenig von sich preisgeben oder sogar anonym auftreten möchten. Fehlendes Vertrauen der Kommunikationspartner, wie auch die Gefahr von Angriffen Dritter, sollen dabei durch Schutzmechanismen wie z. B. Verschlüsselung oder Anonymität bei der Kommunikation ausgeglichen werden. Dafür wurden Schutzziele entwickelt, um aus diesen die nötigen Anforderungen an ein System abzuleiten (s. [WP00, Pfi06b]).

- **Vertraulichkeit**: Der Inhalt einer Nachricht wird geheim gehalten. Kein unbefugter Dritter kann den Inhalt der Nachricht erkennen.

[2] BDGS §3a (Datenvermeidung und Datensparsamkeit)

- **Verdecktheit**: Versteckt die Übertragung einer Nachricht, kein Dritter soll die Existenz einer Nachricht erkennen können.
- **Unbeobachtbarkeit**: Ein Benutzer kann Dienste oder Ressourcen nutzen, ohne dass andere beobachten können, dass der Dienst oder die Ressource genutzt wird.
- **Anonymität**: Kommunikationspartner und Dritte erfahren nicht die Identität eines Benutzers.
- **Pseudonymität**: Die Nutzung einer Ressource ist einem Benutzer zurechenbar, ohne dass dieser seine reelle Identität offenbaren muss.
- **Zurechenbarkeit**: Das Senden (bzw. Empfangen) von Informationen kann gegenüber Dritten bewiesen werden.
- **Integrität**: Der Empfänger kann (unbefugte) Modifikationen einer Nachricht erkennen.
- **Verbindlichkeit**: Ein Nutzer kann belangt werden, um seine Zusagen innerhalb einer angemessenen Zeit zu erfüllen.
- **Verfügbarkeit**: Sichert die Nutzbarkeit von Diensten und Ressourcen für einen Benutzer.
- **Erreichbarkeit**: Sichert, dass bei Bedarf mit einer Ressource (z. B. auch mit einem anderen Nutzer) Kontakt aufgenommen werden kann.

Diese Ziele sind gegenüber dem Kommunikationspartner als auch gegenüber potenziellen Dritten zu betrachten. Dabei sind die einzelnen Schutzziele nicht unabhängig voneinander, sondern können Wechselwirkungen aufweisen. Zum Beispiel folgt Anonymität aus Unbeobachtbarkeit. Verdecktheit verstärkt Anonymität, wobei wiederum Anonymität komplementär zu Zurechenbarkeit ist [WP00].

Da es sich bei den zu schützenden Daten im Zusammenhang mit Privatsphäre um (personenbezogene) *Benutzerdaten* handelt, werden im Folgenden einige der oben beschriebenen Schutzziele im Hinblick auf ihre Bedeutung für Benutzerdaten aufgegriffen (s. auch [Wör03]):

- **Bedeutung von *Vertraulichkeit* für Benutzerdaten**: Nicht autorisierte Dritte dürfen keinen Zugriff auf Benutzerdaten erlangen. Dies betrifft die Speicherung und Übertragung der Daten sowie die Verwendung und Weitergabe dieser.
- **Bedeutung von *Verdecktheit* für Benutzerdaten**: Die Übertragung von Benutzerdaten zu einem Dienst soll verdeckt stattfinden.
- **Bedeutung von *Unbeobachtbarkeit* für Benutzerdaten**: Die Kommunikation mit einem Dienst soll nicht ersichtlich sein, da die Möglichkeit besteht, dass ein unbefugter Dritter hieraus Informationen gewinnt. Beispielsweise könnte aus der

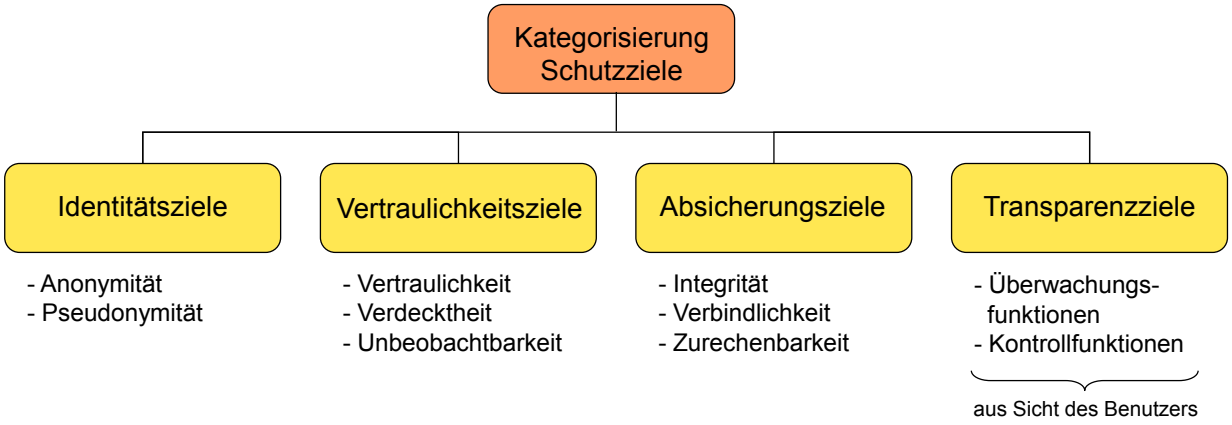


Abbildung 2.12: Kategorisierung von Schutzzielen.

Nutzung einer Informationsquelle zu gesundheitlichen Fragen auf eine Krankheit einer Person geschlossen werden.

- **Bedeutung von *Anonymität* für Benutzerdaten**: Benutzer sollen Funktionalitäten personalisierbarer Dienste verwenden und ihnen ihre Daten zur Verfügung stellen können, ohne dass die Dienste ihre Identität kennen.

- **Bedeutung von *Pseudonymität* für Benutzerdaten**: Die Nutzung von manchen personalisierbaren Dienstleistungen ist nicht immer völlig anonym möglich, sodass in diesen Fällen die Interaktion mit einem Pseudonym wünschenswert ist.

- **Bedeutung von *Verbindlichkeit* für Benutzerdaten**: Betrifft z. B. die Forderung, dass keine Weitergabe der Benutzerdaten an Dritte durch den Dienst geschieht. Es geht hier um die Einhaltung von Zusagen durch den Dienst, was die Verwendung der Daten angeht.

- **Bedeutung von *Verfügbarkeit* für Benutzerdaten**: Personalisierbare Dienste sollten stets Zugriff auf benötigte Benutzerdaten haben.

Die verschiedenen Schutzziele haben also unterschiedliche Bedeutung für den Schutz von Benutzerdaten bzw. von digitalen Identitäten. Diese können in verschiedene Kategorien einordnet werden, um die Hauptaspekte weiter herauszuarbeiten [TP01, KP03]. Anonymität und Pseudonymität sind miteinander verwandt und lassen sich in die Kategorie „*Identitätsziele*“ einordnen. Vertraulichkeit, Verdecktheit und Unbeobachtbarkeit sind miteinander verknüpft und finden unter der Kategorie „*Vertraulichkeitsziele*“ ihren Platz. Integrität, Verbindlichkeit und Zurechenbarkeit lassen sich unter „*Absicherungsziele*“ zusammenfassen. Ein weiterer Aspekt betrifft die Frage, wie eine Person erfahren kann, welche Daten von ihr zu welchem Zeitpunkt gesammelt werden und unter welchen Umständen dies geschieht [Wör03, KK06, HBCDF06]. Dies kann als „*Transparenzziel*“ bezeichnet werden

	Objekt 1 Textdatei	*Objekt 2* Editor	*Objekt 3* Mozilla
Subjekt 1 Editor	read, write	kill	-
Subjekt 2 Mozilla	-	-	kill

Tabelle 2.1: Beispiel einer Zugriffskontrollmatrix

und ergibt sich auch aus den rechtlichen Rahmenbedingungen für den Datenschutz. Diese Kategorisierung ist in Abbildung 2.12 grafisch dargestellt.

Um den oben aufgestellten Schutzzielen gerecht zu werden, müssen unterschiedliche Anforderungen an den Umgang mit Benutzerdaten gestellt werden. Um diesen Anforderungen gerecht zu werden, müssen verschiedene Mechanismen angewandt werden. Hierzu gehört neben Identitätsmanagement und Anonymisierung auch ein System zur Gewährleistung der Zugriffskontrolle, das dem Benutzer erlaubt, festzulegen, wer wie auf welche Benutzerdaten zugreifen darf [Sch01, DVFS04, PAN05, Wör03]. Im nächsten Abschnitt werden verschiedene Konzepte zur Zugriffskontrolle vorgestellt, von denen zwei im Rahmen dieser Arbeit für die Zugriffskontrolle für Benutzerdaten und eHome-Dienste angewendet werden.

2.6 Zugriffskontrolle

Seit dem Bestehen von Computersystemen mit gemeinsamer Ressourcennutzung durch verschiedene Personen und Prozesse ist die *Zugriffskontrolle* ein wichtiges Thema. Viele *Zugriffskontrollmodelle* haben ihren Ursprung bei Sicherheitsdiensten und dem Militär. Mittlerweile gibt es eine Reihe von Ansätzen für die Zugriffskontrolle, die unterschiedlichen Bedürfnissen gerecht werden, darunter auch die Gewährleistung der Privatsphäre.

Bei den Zugriffskontrollmechanismen wird zwischen Subjekten und Objekten unterschieden, wobei immer Subjekte auf Objekte zugreifen. Die folgenden Abschnitte geben einen kurzen Überblick über grundlegende Zugriffskontrollmodelle.

2.6.1 Zugriffsmatrixmodell und Zugriffskontrollliste

Das älteste und einfachste Zugriffsmodell ist das *Zugriffsmatrixmodell*, welches auch unter dem Begriff *Referenzmonitormodell* bekannt ist [Lam74, SS94]. Das Problem der Rechtezuordnung wird hier als Zugriffsmatrix betrachtet, wobei die Spalten den Objekten und die Zeilen den Subjekten zugeordnet sind. Das entsprechende Feld einer Subjekt-Objekt-Kombination gibt die Rechte des Subjekts an dem Objekt wieder. Matrixänderungen kann nur ein Benutzer durchführen, der die entsprechenden Rechte dazu hat (z. B. Administrator oder Eigentümer).

Tabelle 2.1 ist zu entnehmen, dass `Subjekt 1` (ein Editorprogramm) das `Objekt 1` (eine Textdatei) lesen und schreiben darf, `Objekt 2` (das Editorprogramm) beenden kann und

bezüglich des `Objekt 3` (dem Browserprogramm) keine Rechte besitzt. `Subjekt 2` hingegen hat nur bezüglich des `Objekt 3` Rechte, nämlich das Browser-Programm zu beenden.

Das Modell ist durch seine Einfachheit flexibel einsetzbar, die Matrix kann aber schnell sehr komplex und unhandlich werden. Zur Modellierung z. B. einer großen Menge von Subjekten, wie sie in großen Systemen vorkommt, eignet sich dieser Ansatz nicht.

Soll dieses Konzept trotzdem angewendet werden, eignen sich *Zugriffskontrolllisten* (engl. Access Control List (ACL)) für die Realisierung solcher Matrizen. Dabei wird jedem Objekt eine ACL zugewiesen, in der die Rechte aller Subjekte an dem Objekt eingetragen werden. Es handelt sich also um eine spaltenweise Speicherung der Matrix. Um Rechte von Subjekten an einem Objekt zu prüfen, reicht es, die zugehörige ACL zu überprüfen. Auch die Änderung der Rechte ist durch das Ersetzen der alten ACL durch ein neue einfach möglich.

ACLs werden in der vorliegenden Arbeit zur Kontrolle von Zugriffen durch Dienste auf Benutzerdaten angewendet (s. Kapitel 6).

2.6.2 Weitere Zugriffskontrollmodelle

Neben dem Zugriffsmatrixmodell existieren noch weitere Zugriffskontrollmodelle. In der Literatur werden meist drei Grundmodelle beschrieben, *Discretionary Policies* (*DAC*), *Mandatory Policies* (*MAC*) und *Role-Based Policies* (*RBAC*) [SS94, LABW92]. Diese werden im Folgenden beschrieben.

2.6.2.1 Discretionary Access Control (DAC)

Discretionary heißt „beliebig“ und bedeutet, dass die zulässigen Zugriffsarten für jedes Subjekt oder für jede Gruppe von Subjekten und jedes Objekt vom Eigentümer des Objekts vergeben wird. Der ursprüngliche Besitzer jedes Objekts ist der Erzeuger, bis dieser die Eigentümerschaft an ein anderes Subjekt überträgt. Bei jedem Zugriff werden die Rechte des Subjekts überprüft. Hierbei werden offene und geschlossene Richtlinien unterschieden, je nachdem ob jede nicht explizit erfasste Kombination von Subjekt, Objekt und Zugriffsart als Berechtigung oder als Verbot interpretiert wird. Dieser Ansatz stammt aus dem Bereich der Dateisysteme. Die Zugriffsarten hierbei sind „Lesen“, „Schreiben“ und „Ausführen“. Der Nachteil dieses Modells ist die fehlende Überprüfung des Datenflusses. Es wird nicht festgelegt, was ein Subjekt mit den Informationen machen darf, wenn es einmal Zugriff darauf erhalten hat. So kann es Informationen an unberechtigte Objekte übertragen.

2.6.2.2 Mandatory Access Control (MAC)

Bei dieser Richtlinie werden Zugriffsrechte in Form von Regeln verbindlich festgelegt. Dieser Ansatz eignet sich zur Kontrolle von Informationsflüssen und hat seinen Ursprung im militärischen Anwendungsbereich. Es findet eine Klassifizierung von Subjekten und Objekten nach deren Sicherheitsstufe statt. Diese Klassifizierung drückt dabei den Grad der Vertrauenswürdigkeit bzw. der Sensibilität der Daten aus. Wer Besitzer der Daten ist, spielt hier keine Rolle. Ein Objekt darf von einem Subjekt nur dann gelesen werden,

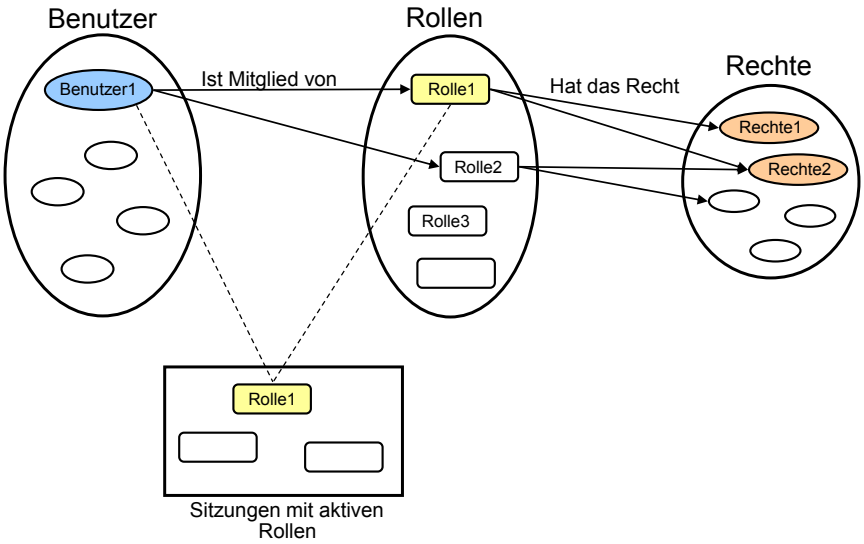


Abbildung 2.13: Rollenbasierte Zugriffskontrolle (RBAC).

wenn das Subjekt die gleiche oder eine höhere Sicherheitsstufe besitzt als das Objekt. Ein Subjekt darf ein Objekt nur dann schreiben, wenn es die gleiche oder eine niedrigere Stufe besitzt. Dadurch wird verhindert, dass ein Subjekt Informationen aus seiner Stufe in Objekte einer niedrigeren Stufe schreibt, denn für die niedrige Stufe sind die Informationen nicht freigegeben. Dadurch eignet sich dieses Modell zur Überprüfung des Datenflusses.

2.6.2.3 Role-Based Access Control (RBAC)

Der Ansatz der rollenbasierten Zugriffskontrolle hat sich aus Anwendungen entwickelt, wo der DAC-Ansatz zu wenig restriktiv ist und wo komplexere Aktivitäten der Subjekte auftreten, die sich auch nicht einfach mit der Einordnung in Sicherheitsstufen erfassen lassen. Rollen beinhalten hier die Berechtigungen für bestimmte Aktionen, die zu einer Aufgabe gehören. Subjekte sind Inhaber einer oder mehrerer Rollen und melden sich im System mit jener Rolle an, die ihrer aktuellen Aufgabe entspricht. Das Prinzip der minimalen Rechte lässt sich sehr gut mit Rollen modellieren. Mittlerweile haben sich aus dem ursprünglichen Rollenkonzept viele Variationen entwickelt.

Die Grundstrukturen beim RBAC (engl. *Role-Based Access Controll*) sind *Rolle*, *Subjekt*, *Recht* und die grundlegenden Relationen zwischen diesen. Die Art der Rechtedefinition ist nicht genau spezifiziert, da sie anwendungsabhängig ist. Die Rechte werden direkt an Rollen geknüpft. Ist ein Benutzer Mitglied einer Rolle, besitzt er automatisch die Rechte dieser Rolle. Ist der Benutzer Mitglied in mehreren Rollen, entscheidet er, mit welcher Rolle er sich beim System anmeldet [SCFY96].

Folgende Zuordnungen und Funktionen geben einen Überblick über mögliche Relationen zwischen Subjekten und Rollen sowie Rollen und Rechten:

- Jedes Subjekt kann mehrere Rollen annehmen, und jede Rolle kann mehreren Subjekten zugeordnet sein (m:n-Relation, engl. *User Role Assigment* (URA)).

- Jede Rolle kann mehrere Rechte beinhalten, und jedes Recht kann in verschiedenen Rollen enthalten sein (m:n-Relation, engl. *Permission Role Assigment* (PRA)).
- Eine Sitzung definiert eine Funktion, die ein Subjekt für einen Zeitraum an eine oder mehrere Rollen bindet, in denen es agiert und für die es autorisiert ist.

Abbildung 2.13 zeigt die grundsätzliche Idee von rollenbasierten Zugriffsmodellen. Hier besitzt beispielsweise Benutzer `b1` die beiden Rollen `e1` und `e3`. Er bekommt aber nur die Rechte von Rolle `e1`, nämlich `r1` und `r2`, zugeteilt, weil er in diesem Beispiel mit dieser Rolle im System angemeldet ist.

2.6.3 Authentifizierung als Voraussetzung der Zugriffskontrolle

Eine wichtige Voraussetzung für eine funktionsfähige Zugriffskontrolle ist der Nachweis einer Eigenschaft (oft die Identität) eines Subjekts, der auf ein Objekt zugreifen möchte. Im Englischen wird der Gesamtprozess als *Authentication* bezeichnet, im Deutschen wird hingegen zwischen Authentifizierung und *Authentisierung* unterschieden. Während eine Partei durch die Authentisierung etwas behauptet, verifiziert die zweite Partei diese Behauptung durch die Authentifizierung. In dieser Arbeit wird diese Unterscheidung nicht strikt befolgt.

Es existieren vielfältige Verfahren der Authentifizierung, weil die Anforderungen anwendungsspezifisch variieren können. Für sensible Aktionen ist eine sicherere Authentifizierung erforderlich als bei gewöhnlichen Aktionen. Daher gibt es auch unterschiedlich starke Verfahren. Die Bandbreite erstreckt sich von einer Authentifizierung mit Benutzernamen und Passwort bis hin zur Überprüfung biometrischer Merkmale.

Die Authentifizierung steht im direkten Zusammenhang mit der Anonymität. Bei der Authentifizierung wird die Identität einer Person festgestellt. Das läuft aber der Geheimhaltung der Identität zuwider, die wiederum für den Schutz der Privatsphäre wichtig ist. Es muss demnach ein Verfahren eingesetzt werden, welches eine Person ausreichend identifiziert, um eine Zugriffskontrolle durchführen zu können, darüber hinaus aber keine weitere Identifikation zulässt. Eine Möglichkeit, diesen Anspruch umzusetzen, sind anonyme Credentials, die in Abschnitt 2.8 vorgestellt werden.

2.7 Kryptografie

Verschlüsselung ist ein Verfahren, bei dem ein lesbarer Klartext in einen unlesbaren Geheimtext umgewandelt wird. Die Umwandlung in umgekehrter Richtung, also vom Geheimtext zum Klartext, wird *Entschlüsselung* genannt. Von entscheidender Bedeutung hierfür ist der Schlüssel. Dieser Parameter bestimmt, auf welche Weise der Klartext umgewandelt wird.

Die *Kryptografie* ist das wissenschaftliche Forschungsgebiet, das sich mit der Verschlüsselung beschäftigt. Dabei werden hauptsächlich Ziele Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit verfolgt (s. Abschnitt 2.5.1).

Es wird dabei zwischen zwei Arten von Verschlüsselungsverfahren unterschieden, die anhand der verwendeten Schlüssel in symmetrische und asymmetrische Verfahren unterteilt werden, wie im Folgenden beschrieben wird.

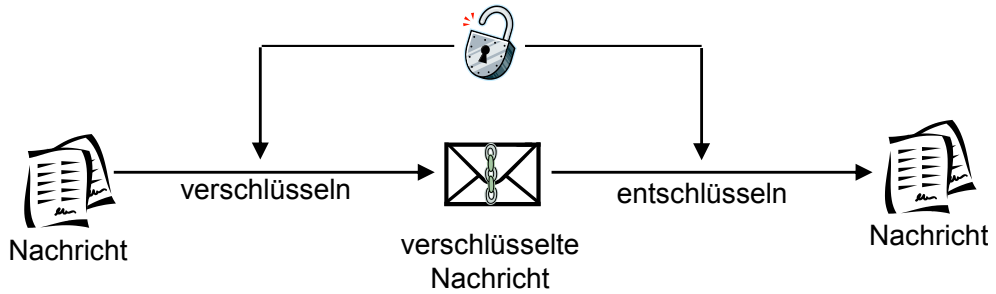


Abbildung 2.14: Symmetrische Verschlüsselung: Konzept.

2.7.0.1 Symmetrische Verschlüsselung

Eines der Dogmen, von denen in der Kryptologie gesprochen wird, besagt, dass nicht die Geheimhaltung der Funktionalität einen guten Verschlüsselungsalgorithmus ausmacht, sondern die mit der bloßen Mathematik (Logarithmusproblem, Primzahlfaktoren) einhergehende zeitliche „Unmöglichkeit“ der Berechnung.

Das in dieser Arbeit verwendete asymmetrische Verschlüsselungsverfahren ist eine Weiterentwicklung des *symmetrischen Verschlüsselungsverfahrens*. Hierunter wird ein Algorithmus verstanden, bei dem zur Ver- und Entschlüsselung der gleiche Schlüssel verwendet wird, d. h., Sender und Empfänger besitzen den gleichen Schlüssel.

Dadurch wird sowohl für die Verschlüsselung als auch für die Entschlüsselung der gleiche Schlüssel verwendet. Symmetrische Verfahren werden in Blockchiffren und Stromchiffren aufgeteilt. Blockchiffren arbeiten mit einer festen Blocklänge. Alle in diesen Block passenden Zeichen werden in einem Schritt ver- oder entschlüsselt. Stromchiffren hingegen verarbeiten die Eingabe Zeichen für Zeichen.

Abbildung 2.14 zeigt das Grundprinzip symmetrischer Verschlüsselung. Eine Nachricht wird mit einem Schlüssel verschlüsselt, der allen an der Kommunikation Beteiligten bekannt ist. Die verschlüsselte Nachricht wird an den oder die Empfänger weitergeleitet, und kann mit dem gleichen Schlüssel wieder entschlüsselt werden.

Symmetrischen Verfahren haben den Vorteil, dass sie in der Regel wesentlich schneller sind als asymmetrische Verfahren. Ihr großer Nachteil ist hingegen ihre Symmetrie. Da derselbe Schlüssel sowohl für die Ver- als auch für die Entschlüsselung benutzt wird, muss dieser über einen sicheren Kanal ausgetauscht werden. Dies stellt sich häufig als Problem dar.

Es existiert eine große Vielfalt von symmetrischen Verfahren. Der *Advanced Encryption Standard (AES)* [DR02] ist ein verbreiteter Algorithmus. Verwendung findet er beispielsweise in dem *Secure Shell (SSH)* Protokoll [BKN06].

2.7.0.2 Asymmetrische Verschlüsselung

Im Gegensatz zu symmetrischen Verfahren werden bei *asymmetrischen Verfahren* unterschiedliche Schlüssel für die Ver- und Entschlüsselung eingesetzt. Jeder Teilnehmer besitzt ein Schlüsselpaar, bestehend aus einem *geheimen* und einem *öffentlichen* Schlüssel. Der

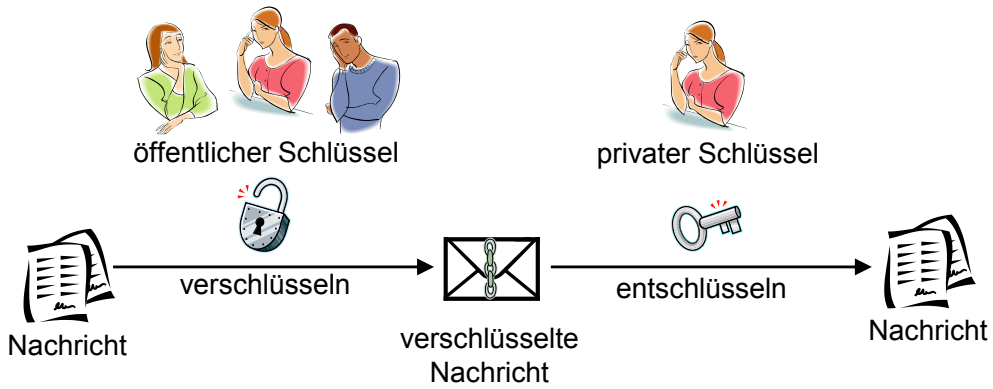


Abbildung 2.15: Asymmetrische Verschlüsselung: Konzept.

öffentliche Schlüssel des Empfängers dient zum Verschlüsseln einer Nachricht. Nur mit dem dazugehörigen privaten Schlüssel kann diese Nachricht entschlüsselt werden. Dadurch kann eine Nachricht immer nur an einen Empfänger geschickt werden. Für mehrere Empfänger muss die Nachricht mehrmals mit unterschiedlichen Schlüsseln verschlüsselt werden. Ein Austausch der Schlüssel stellt sich als ausgesprochen einfach dar. In der Regel wird der öffentliche Teil über sogenannte Schlüsselserver bekannt gemacht. Dort kann ihn jeder abrufen und benutzen, um eine Nachricht an den Inhaber zu verschlüsseln.

Abbildung 2.15 zeigt das Grundprinzip asymmetrischer Verschlüsselung. Der Empfänger der Nachricht erzeugt einen öffentlichen und einen privaten Schlüssel. Den öffentlichen Schlüssel gibt er allen Sendern bekannt, die ihm eine Nachricht zukommen lassen wollen. Ein Sender verschlüsselt die Nachricht mit dem öffentlichen Schlüssel des Empfängers. Der Empfänger kann die Nachricht mit seinem privaten Schlüssel entschlüsseln. Falls ein Angreifer den Kommunikationsweg abhören sollte, wäre dieser nicht in der Lage, die Nachricht zu entschlüsseln, da ihm der private Schlüssel nicht bekannt ist. Der wohl bekannteste Algorithmus dieser Art ist das RSA-Verfahren[RK05]. Ein Nachteil asymmetrischer Verfahren liegt darin, dass sie im Vergleich zu symmetrischen Verfahren sehr langsam sind.

Die asymmetrische Verschlüsselung kann für die *Authentifizierung* verwendet werden. Abbildung 2.15 zeigt dieses Verfahren an einem Beispiel. Laura und Nick machen sich gegenseitig ihre öffentlichen Schlüssel bekannt. Dann verschlüsselt Laura die Nachricht zunächst mit ihrem privaten Schlüssel und danach mit dem öffentlichen Schlüssel von Nick. Nachdem die Nachricht Nick zugestellt wurde, entschlüsselt er diese zuerst mit seinem privaten Schlüssel und dann mit dem öffentlichen Schlüssel von Laura. Letzteres kann allerdings nur erfolgreich gelingen, wenn die Nachricht wirklich von Laura kam. Ist dies nicht der Fall, passt der verwendete Schlüssel nicht. Dadurch ist sichergestellt, dass

- nur Nick die Nachricht lesen kann, da nur er den zu seinem öffentlichen Schlüssel passenden privaten Schlüssel besitzt,
- nur Laura die Nachricht erstellt haben kann und
- die Nachricht auf dem Weg nicht verändert wurde.

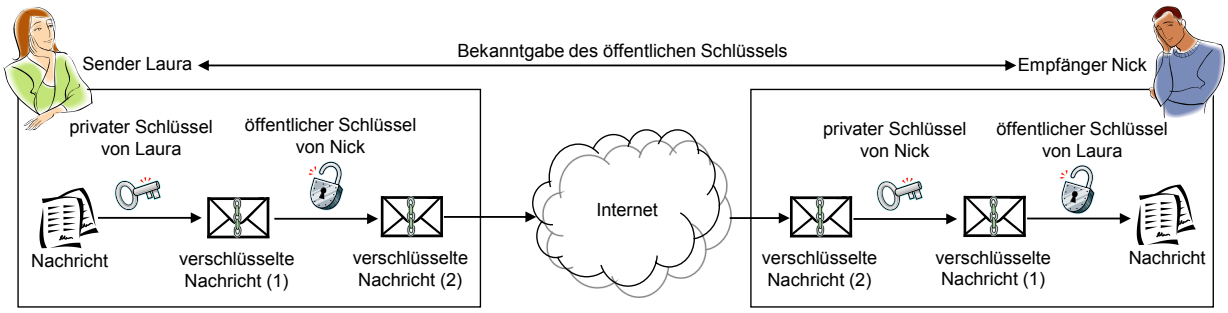


Abbildung 2.16: Asymmetrische Verschlüsselung: Beispiel für Authentifizierung.

Aus Effizienzgründen sieht es in der Praxis allerdings so aus, dass eine Nachricht nicht komplett mit dem privaten Schlüssel des Senders verschlüsselt wird, sondern eine *Hash-Funktion* zum Erzeugen eines eindeutigen *Fingerabdrucks* verwendet wird. Die Details würden hier jedoch zu weit führen. Daher sei hier auf weiterführende Literatur wie etwa [PP09] verwiesen.

2.7.0.3 Hybride Verfahren

Da beide Verfahren ihre Vor- und Nachteile haben, sind *hybride* Verfahren als Kombination beider die logische Konsequenz. Die eigentliche Verschlüsselung erfolgt über einen symmetrischen Algorithmus. Damit wird der Geschwindigkeitsvorteil ausgenutzt. Der symmetrische Schlüssel wird über ein asymmetrisches Verfahren ausgetauscht, indem der symmetrische Schlüssel mit einem öffentlichen asymmetrischen Schlüssel verschlüsselt wird. Dann wird kein sicherer Kanal benötigt, um den Schlüssel auszutauschen.

Hybride Verfahren sind sehr weit verbreitet. Hybride Verschlüsselung wird zum Beispiel im E-Mail-Verkehr eingesetzt. Das bekannte übliche *(Open-)PGP* [PGP07] verwendet einen einmaligen symmetrischen Sitzungsschlüssel, um die E-Mail zu verschlüsseln. Mit dem öffentlichen asymmetrischen Schlüssel des Empfängers wird der Sitzungsschlüssel verschlüsselt und der E-Mail beigefügt. Der Empfänger muss nun zu erst den Sitzungsschlüssel entschlüsseln, um anschließend die Nachricht entschlüsseln zu können.

Auch das Netzwerkprotokoll *SSL* (Secure Socket Layer) benutzt ein hybrides Verschlüsselungsverfahren. Eine Menge von Webseiten werden per SSL verschlüsselt übertragen. Erkennbar ist dies an der URL, die mit *https* beginnt. Wenn solch eine Seite aufgerufen wird, passiert Folgendes: Der Browser erzeugt kurzfristig einen privaten und einen öffentlichen Schlüssel. Der öffentliche Schlüssel wird dem Webserver übermittelt. Dieser verwendet den öffentlichen Schlüssel, um einen neu erzeugten symmetrischen Schlüssel zu verschlüsseln und zum Browser zurück zu übermitteln. Der Browser empfängt den symmetrischen Schlüssel und kann fortan die symmetrisch verschlüsselten Webseiten empfangen und entschlüsseln. Diese Technik ist zielführend, da die komplette asymmetrische Verschlüsselung aller Webseiten zu rechenintensiv wäre.

2.8 Anonyme Credentials

Eine wichtige Entwicklung im Gebiet der Computersicherheit bilden sogenannte *digitale Credentials*. Sie bilden ein Analogon zu analogen Credentials, es wird auch von *natürlichen Credentials* gesprochen, der Realen Welt wie etwa Führerscheine, Ausweise oder Kinokarten. Üblicherweise werden analoge Credentials von einer Behörde oder Organisation ausgestellt und berechtigen den Besitzer, gewisse Ressourcen zu benutzen. Der Begriff der digitalen Credentials wurde erstmals von David Chaum in [Cha85] als „statements concerning an individual that are issued by organizations, and are in general shown to other organizations“ eingeführt. Diese *digital signierten* Aussagen können ähnlich zu analogen Credentials Informationen über den Namen, das Alter, die Staatsbürgerschaft usw. enthalten. Durch den Nachweis über den Besitz von Credentials kann der Besitzer Zugriff auf Dienste und Ressourcen anfordern [Cha92, Cha90, Cha85].

2.8.1 Eigenschaften von Credentials

Durch Anwendung der asymmetrischen Verschlüsselung können digitale Credentials mit unterschiedlichen Merkmalen ausgestattet werden werden: [Bis02, CH02, GEB99, Bra02, Bra00, BBC08]:

- **Ablauf**: Credentials können mit einem Ablaufdatum versehen werden. So kann verhindert werden, dass sie über einen bestimmten Zeitpunkt hinaus verwendet werden.

- **Verbrauchbarkeit**: Hier wird festgelegt, wie oft ein Credential verwendet werden darf. Beispielsweise ist die Kinokarte nur einmalig und meist auch nur zu einem bestimmten Zeitpunkt nutzbar. Auch elektronische Tickets können in der Häufigkeit ihrer Verwendbarkeit eingeschränkt werden. Nur einmalig verwendbare Credentials werden auch als One-Show-Credentials bezeichnet.

- **Übertragbarkeit**: Credentials sind übertragbar oder nicht übertragbar. Nicht-übertragbare Credentials, wie z. B. der Personalausweis, sind mit einer bestimmten Person verbunden. Übertragbare Credentials können an andere Personen weitergegeben werden, wie z. B. eine Eintrittskarte fürs Kino. Falls eine Nichtübertragbarkeit bei einem anonymen Credential gewünscht ist, muss dies mit speziellen Mitteln erreicht werden.

- **Teilbarkeit**: Eine Erweiterung der Übertragbarkeit eines Credentials ist die Teilbarkeit. Hier kann das Credential bei fortwährendem Besitz des Credentials weitergegeben werden. Ein Beispiel hierfür ist der Wohnungsschlüssel, der vervielfältigt und weitergegeben werden kann.

- **Anonymität**: Anonyme Credentials lassen keine eindeutige Identifizierung des Besitzers zu. Somit ist es möglich, dass der Besitzer dem Kommunikationspartner zwar gewisse Aussagen über sich (Eigenschaften oder Rechte) nachweist, aber seine reelle Identität nicht preis gibt. Beispielsweise weist Geld die Anonymitätseigenschaft auf,

weil bei der Barzahlung im Supermarkt nicht die Identität des Bezahlenden bekannt gegeben werden muss. Anders ist dies beim elektronischen Zahlen per Kredit- oder EC-Karte. Letztere haben eine dem Benutzer zugeordnete Nummer, mit der detailliert aufgezeichnet werden kann, wann der Besitzer wo was eingekauft hat. Die Anonymität kann zum Beispiel durch den Einsatz von Pseudonymen und Verschlüsselungstechniken realisiert werden. Digitale Credentials mit dieser Eigenschaft werden als *anonyme (digitale) Credentials* bezeichnet.

- **Aufdeckbarkeit**: Manchmal kann es sein, dass der Kommunikationspartner verlangt, dass die Identität des Besitzers eines anonymen Credentials aufgedeckt werden kann, wenn dieser sich nicht an die verabredeten Regeln bei der Nutzung von Ressourcen oder Diensten hält. Diese Eigenschaft wird meist vor Beginn der Kommunikationsbeziehung ausgehandelt, während zum Aufdecken der reellen Identität oft eine dritte Partei benötigt wird.

- **Widerruf**: Es kann auch vorkommen, dass ein einmal ausgestelltes Credential aufgrund bestimmter Ereignisse widerrufen werden muss. Beispielsweise kann es sein, dass einer Person der Führerschein entzogen wird, weil sie die Verkehrsregeln wiederholt missachtet hat. Die Ankündigung des Widerrufs kann beispielsweise über Sperrlisten geschehen.

- **Fälschungssicherheit**: Es muss sichergestellt werden, dass Credentials nicht gefälscht werden können. Zu dieser Eigenschaft zählt auch, dass eine Person ihr Credential nicht an andere Personen weitergeben kann. Um dies zu verhindern, wird jedes Pseudonym einer Person mit dieser verknüpft. Da Credentials für Pseudonyme ausgestellt werden, sind auch diese mit der Person verknüpft. Dies kann durch die Nutzung der asymmetrischen Verschlüsselung erreicht werden.

- **Getrennte Schlüsselgenerierung**: Diese Eigenschaft erlaubt den Benutzern und Organisationen die getrennte Generierung von Schlüsseln. Dadurch wird die Schlüsselverwaltung vereinfacht und ein flexibles Zusammenarbeiten der Teilnehmer möglich.

- **Effizienz**: Um den Einsatz auch auf mobilen Geräten zu ermöglichen, sollten möglichst effiziente Verfahren zur Generierung und zum Nachweis von Credentials eingesetzt werden.

- **Selbstständigkeit/Widerspruchsfreiheit**: Dies bezieht sich darauf, ob ein Credential alleine Gültigkeit besitzt oder nur in Kombination mit anderen. Beispielsweise kann ein Studententicket nur in Kombination mit einem Lichtbildausweis als Fahrticket für Bus und Bahn verwendet werden. Dabei muss sichergestellt werden, dass die Credentials demselben Besitzer gehören und nicht unterschiedliche Personen ihre Credentials kombinieren.

- **Selektive Freigabe**: Dies ist eines der wichtigsten Eigenschaften von digitalen Credentials, die sie von analogen Credentials abhebt. Erfüllt ein Credential diese Eigenschaft,

ist der Besitzer in der Lage, nur eine Teilmenge der in dem Credential enthaltenen Aussagen nachzuweisen, also nur diejenigen, die in der aktuellen Situation notwendig sind. Beispielsweise reicht es, wenn ein Kinobesucher nachweist, dass er das für den Film erforderliche Mindestalter hat. Er muss aber nicht weitere Informationen wie seinen Namen oder sein konkretes Geburtsdatum preisgeben. Beim Vorlegen des Papierausweises können hingegen u. a. auch diese Informationen eingesehen werden. Diese Eigenschaft ermöglicht somit die Minimierung der frei gegebenen Daten und verbessert den Schutz der Privatsphäre.

Obige Eigenschaften vermitteln einen ersten Eindruck über die Möglichkeiten für den Einsatz von digitalen Credentials. Eine genaue Betrachtung dieser Eigenschaften zeigt, dass als Hauptanwendungsgebiete für (anonyme) digitale Credentials die *Authentifizierung* und der *Schutz der Privatsphäre* in Frage kommen. Im Folgenden werden einige bekannte Systeme vorgestellt, die sich die Handhabung von Credentials zugeschrieben haben. Für weiterführende Informationen, insbesondere für theoretische Grundlagen digitaler Credentials, sei dem geneigten Leser weiterführende Literatur wie etwa die Übersicht in [Bra02] empfohlen.

2.8.2 Bekannte Credential-Systeme

Ein klassisches Beispiel für ein System mit digitalen Credentials (*Credentials-System*) ist der Authentifizierungsdienst *Kerberos* [NT94]. Bei Kerberos muss sich eine Person nur einmal authentifizieren. Anschließend bekommt sie ein digitales Credential, auch *Ticket* genannt. Mit diesem Ticket kann sie Dienste nutzen, ohne sich erneut authentifizieren zu müssen. Kerberos weist jedoch nicht die Anonymitätseigenschaft auf, die für die Ziele dieser Arbeit von besonderer Bedeutung sind.

Die Wurzeln von *anonymen Credential-Systemen* gehen auf Chaum zurück. Er präsentierte in [Cha85] ein System, das Benutzern das Erlangen und Vorweisen von Credentials auf Basis von *Pseudonymen* ermöglicht, die untereinander *nicht verkettbar* sind.

Ein weiteres bekanntes anonymes Credential-System ist das von Brands entwickelte und inzwischen von Microsoft aufgekaufte *U-Prove* [BDD07, Bra00, Mic10]. Hier werden Credentials als *ID Token* bezeichnet. Es existieren zwei Versionen der ID Token. Sie unterscheiden sich in ihrer Lebensdauer: *Transient Tokens* werden bei Bedarf erzeugt und sind nur während einer Sitzung gültig. *Long-lived Tokens* können gespeichert werden und sind dauerhaft gültig. U-Prove weist unter anderem Eigenschaften wie Fälschungssicherheit, selektive Freigabe, Widerruf und Verbrauchbarkeit auf.

Auch IBM hat sich jahrelang mit anonymen Credential-Systemen beschäftigt. Das Ergebnis ist das im Züricher Forschungslabor unter der Federführung von Camenisch entwickelte *idemix* [CL01, CH02], welches als Kurzform für *Identity Mixer* steht. Es stellt Schnittstellen zur Erzeugung und Verwendung von anonymen Credentials bereit. Das Projekt besteht aus den entwickelten theoretischen Konzepten und einer Referenzimplementation. Die Referenzimplementation fand unter anderem in dem PRIME-Projekt der Europäischen Union Verwendung [HBPP05, CSS+05, PRI10a]. Dort wurde auch die Entwicklung

der Referenzimplementation vorangetrieben. Momentan wird die Entwicklung durch das Nachfolgeprojekt PrimeLife [Pri10b] fortgeführt. Die wichtigsten der im letzten Abschnitt aufgelisteten Eigenschaften wie Aufdeckbarkeit, Widerruf, Anonymität, Fälschungssicherheit, Nicht-Übertragbarkeit, Verbrauchbarkeit, Ablauf und selektive Freigabe weist idemix auf.

Für den Einsatz in dieser Arbeit kamen sowohl U-Prove als auch idemix infrage. Um zwischen diesen beiden Systemen zu entscheiden, wurden folgende Kriterien herangezogen: Laut [BDD07] ist U-Prove *effizienter* als idemix. Umgekehrt besitzt idemix jedoch deutlich bessere *Anonymisierungseigenschaften* [BBC08]. Dies liegt insbesondere daran, dass es mit idemix möglich ist, dasselbe Credential mehrfach derselben Organisation vorzuzeigen, ohne dass die Organisation das mehrfache Vorzeigen erkennen kann. Daher werden die Credentials bei idemix auch als *unverkettbare* Credentials bezeichnet (engl. *Multi-Show Unlinkable Credentials*). Daher eignet sich idemix besser zum Schutz der Privatsphäre. Dies war einer der Gründe, warum die Entscheidung zugunsten von idemix gefallen ist. Diese Entscheidung hat jedoch auch praktische Gründe. Denn idemix wurde von IBM der eHome-Gruppe freundlicherweise samt Quellcode zur Verfügung gestellt. Dadurch konnten, wenn nötig, Anpassungen vorgenommen werden. Eine einsetzbare Version von U-Prove stand dagegen nicht zur Verfügung, auch Anfragen an Microsoft und Credentica blieben unbeantwortet[3]. Im folgenden Abschnitt wird idemix daher näher beschrieben.

2.8.3 idemix

In diesem Abschnitt wird das zuvor eingeführte *idemix* im Detail beschrieben. Weil idemix die Unverkettbarkeit von Transaktionen mit anonymen Credentials gewährleistet, kann es in Anwendungen integriert werden, die Zugriffskontrolle mit dem Schutz der Privatsphäre kombinieren. Zunächst wird anhand eines Beispiels erläutert, wofür idemix eingesetzt werden kann. Danach werden die unterschiedlichen Rollen von Teilnehmern beschrieben, die idemix vorsieht. Am Anschluss daran werden die grundlegenden Protokolle von idemix beschrieben. Abschließend werden einige spezifische Eigenschaften von idemix-Credentials erläutert.

2.8.3.1 Ein einfaches Beispiel

Am deutlichsten wird die Idee hinter einem anonymen Credential-System wie idemix anhand eines Beispiels: Ein Kunde möchte ein Auto mieten. Dazu betritt er eine Autovermietung und legt seinen Führerschein vor. Der Führerschein stellt den Nachweis über die Berechtigung des Kunden zum Führen eines Fahrzeugs dar. Er enthält aber noch weitere Daten: Alter, Name und Wohnort. All diese Daten kann der Angestellte nun einsehen und benutzen. Dabei kann nicht bestimmt werden, wofür er die Daten benutzt. Im harmlosesten Fall legt er eine Kundenkarteikarte an, um den Kunden in Zukunft Angebote machen zu können. Sobald die Daten bekannt sind, können sie jedoch auch für andere Zwecke benutzt werden.

[3] Im März 2010 hat Microsoft eine erste Version von U-Prove für Testzwecke u. a. für Java (http://code.msdn.microsoft.com/uprovesdkjava) veröffentlicht. Zu dieser Zeit befand sich die vorliegende Arbeit jedoch schon in der Endphase, sodass U-Prove nicht mehr getestet werden konnte.

An dieser Problematik setzt idemix an. Mit idemix kann sich der Kunde vor dem Betreten der Autovermietung bei einer Zertifizierungsstelle ein Credential auf Basis seines Führerscheins ausstellen lassen. Darin können dieselben Information gespeichert werden wie auf dem Führerschein. Statt nun seinen Führerschein vorzeigen zu müssen, kann er das Credential benutzen, um seine Fahrerlaubnis nachzuweisen. Der Benutzer wählt selbst, welche Daten er aus dem Credential zeigen möchte. Im Beispiel benötigt die Autovermietung erst einmal den Nachweis, dass der Benutzer Auto fahren kann. Diese Eigenschaft kann ohne weitere Details des sonstigen Inhaltes des Credentials bewiesen werden. Die Autovermietung erhält keine weiteren Daten. Name, Alter und Wohnort sind nicht ersichtlich und bleiben daher geheim. Auf dieser Grundlage kann der Kunde ein Auto mieten, ohne dabei zu viele Daten preisgeben zu müssen. Im Falle eines Unfalls und der daraus resultierenden Schadensabwicklung benötigt die Vermietung den Namen des Kunden. Über das Credential und eine spezielle dritte Organisation kann die Identität in solchen Fällen offengelegt werden. Dazu ist aber im Vorfeld auch die Mitarbeit des Kunden notwendig.

Der Einsatzort von Idemix ist überall dort, wo gewisse Aussagen bewiesen werden müssen, die Preisgabe weitergehender Informationen aber gar nicht oder nur in speziellen Fällen nötig ist. Damit eignet es sich auch für den Einsatz in eHomes wie in Kapitel 6 gezeigt wird. Bei der Authentifizierung muss der Benutzer dem eHome beispielsweise nachweisen, dass er eine Berechtigung zum Betreten des eHomes und zum Nutzen der dort angebotenen Dienste hat. Gleichzeitig möchte der Benutzer aber oft seine reelle Identität nicht preisgeben und die Menge der freigegebenen Daten minimieren.

2.8.3.2 Rollen von beteiligten Teilnehmern

Das anonyme Credential-System idemix sieht folgende drei unterschiedliche *Rollen* für Teilnehmer vor. Dabei kann jeder Teilnehmer auch mehrere Rollen annehmen.

- **Benutzer**: Benutzer können mit Organisationen entweder anonym oder über ein Pseudonym interagieren. Sie können beliebig viele Pseudonyme bei einer Organisation registrieren lassen und sich unter diesen Pseudonymen Credentials ausstellen lassen. Eine Organisation stellt ein Credential also nicht direkt für den Benutzer, sondern für das vereinbarte Pseudonym aus. Im Normalfall sind die Pseudonyme eines Benutzers bei jeder Organisation unterschiedlich.

- **Organisationen**: Organisationen können Credentials sowohl ausstellen als auch verifizieren. Letzteres bedeutet, dass die Organisation überprüft, ob der von einem Benutzer vorgezeigte Nachweis über den Besitz eines Credentials richtig ist. Eine Organisation kann dabei beide Rollen, also *Aussteller* und *Verifizierer*, annehmen. Der Besitzer eines Credentials ist dem Aussteller unter einem anderen Pseudonym bekannt ist als einem möglichen Verifizierer. Das gilt auch dann, wenn der Aussteller und der Verifizierer gleich sind. Der Aussteller kann also das beim Vorzeigen des Credentials verwendete Pseudonym nicht dem Pseudonym zuordnen, dass beim Ausstellen desselben Credentials verwendet wurde.

- **Annullierer**: Am Beispiel wurde ersichtlich, dass es notwendig sein kann, die Identität eines Credential-Besitzers in bestimmten Situationen aufzudecken, in denen sich der Benutzer nicht an gewisse Regeln gehalten hat. Hierfür bietet idemix zwei optionale Möglichkeiten der *Annullierung der Anonymität*.

  Die *lokale* Annullierung ermöglicht die Aufdeckung des Pseudonyms, unter dem der Besitzer des Credentials beim *Aussteller* bekannt ist. Damit eine lokale Annullierung funktionieren kann, muss zusätzlich zum Benutzer und zum Verifizierer ein vertrauenswürdiger Dritter, ein sogenannter *Annullierer* (engl. *Revocation Manager*) existieren, der im Bedarf die Annullierung vornehmen kann. Ferner muss der Besitzer des Credentials dieser Möglichkeit zustimmen. Gibt es mehrere mögliche Annullierer, kann sich der Benutzer einen aussuchen.

  Die *globale* Annullierung funktioniert ähnlich. Hier wird jedoch vorausgesetzt, dass ein spezieller Aussteller von Credentials existiert (engl. *Root Pseudonym Authority*). Dieser stellt nur Credentials zu Pseudonymen aus, die mit der reellen Identität des Benutzers verknüpft werden können. Jeder Benutzer ist diesem Aussteller demnach unter einem *Root Pseudonym* bekannt und besitzt auch nur ein Credential von diesem Aussteller, das *Root Credential*. Somit ist bei der globalen Annullierung die reelle Identität des Benutzers aufdeckbar.

2.8.3.3 Grundlegende Protokolle

In diesem Abschnitt werden die grundlegenden Protokolle beschrieben, auf denen idemix basiert. Dabei werden die theoretischen Grundlagen nicht im Detail aufgegriffen. Vielmehr soll diese Beschreibung das notwendige Wissen zum Verstehen des später vorgestellten Ansatzes vermitteln, indem idemix zum Einsatz kommt. Für weiterführende Literatur zu idemix mit kryptografischen Details sei hier auf [MS09, CL01, CH02].

Zum Verständnis der folgenden Ausführungen müssen zunächst einige Begriffe und Zusammenhänge erläutert werden. Oben wurde bereits erwähnt, dass zwischen Organisationen und Benutzern unterschieden wird. Jeder Benutzer besitzt einen *geheimen Hauptschlüssel* (x_U), der bei jeder Transaktion verwendet wird. Er ist damit auch mit allen Pseudonymen und Credentials dieses Benutzers verknüpft. Ferner ist jede Organisation in Besitz eines Schlüsselpaares, bestehend aus einem öffentlichen und einem privaten Schlüssel. Dieses Schlüsselpaar wird bei beim Ausstellen und Verifizieren von Credentials eingesetzt.

Eine grobe Skizze der Protokolle ist in Abbildung 2.17 dargestellt. Dabei handelt es sich hier um *interaktive* Protokolle zwischen dem Benutzer und den Organisationen:

- **Pseudonym erstellen**: Bevor ein Benutzer (U) mit einer Organisation interagieren kann, müssen sie sich über ein Pseudonym N für U einigen. Um ein neues Pseudonym zu erstellen, tritt der Benutzer mit der Organisation in Verbindung. Gemeinsam errechnen die beiden Parteien einen Wert, der als Pseudonym bezeichnet wird. Ein Benutzer kann mit der gleichen Organisation mehr als ein Pseudonym aushandeln. Dabei wird immer der gleiche Hauptschlüssel verwendet, die Pseudonyme unterscheiden sich aber und sind nicht miteinander verkettbar.

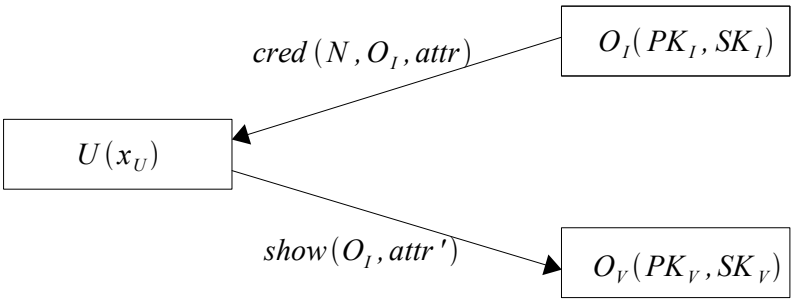


Abbildung 2.17: Grundlegende Protokolle von idemix (Quelle: [CH02]).

- **Credentials ausstellen**: Damit ein Benutzer ein Credential erlangen kann, muss er dem Aussteller (O_i) über ein gültiges Pseudonym (N) bekannt sein. Wenn N berechtigt ist, ein Credential zu erhalten, das ein oder mehrere Attribute *attr* enthält, erstellt O_i ein entsprechendes Credential C, signiert dieses mit seinem privaten Schlüssel und sendet es U. Diese Attribute enthalten gewisse Aussagen über den Benutzer wie etwa seinen Namen oder sein Alter. Normalerweise sind die Attribute beiden Parteien in Klartext bekannt. Ein Attribut kann auch so im Credential verankert werden, dass der Aussteller dieses Attribut nicht lesen kann. Dazu muss der Benutzer dem Aussteller die Korrektheit des Attributs durch einen *Zero-Knowledge-Beweis* nachweisen. Dadurch können auch Attribute im Credential verankert werden, die vor dem Aussteller geheim gehalten werden sollen.

- **Credentials vorzeigen**: Zum Vorzeigen eines Credentials tritt der Benutzer mit dem Verifizierer (O_v) über ein anderes Pseudonym in Kontakt. Der Benutzer ist in Besitz des Credentials C, das von einer anderen oder der gleichen Organisation ausgestellt wurde. Als Erstes vereinbaren die Parteien, welche Attribute des Credentials bewiesen werden sollen. Es ist möglich, ein Attribut vollständig offenzulegen oder nur Eigenschaften darüber zu beweisen. Beispielsweise kann beim Führerschein die Fahrzeugklasse offengelegt werden. Beim Alter kann hingegen bewiesen werden, dass der Benutzer mindestens 18 Jahre alt ist.

  Nachdem die Parteien vereinbart haben, welche Attribute bewiesen werden sollen, kann U das Credential C vorzeigen. Dabei erhält O_v tatsächlich keine genaue Kenntnis über den Inhalt von C. Stattdessen weist U mithilfe eines *Zero-Knowledge-Beweises* nach, dass er erstens im Besitz eines Credentials ist, dass von O_i ausgestellt wurde und die Attribute *attr* enthält, und zweitens Kenntnis über den Hauptschlüssel x_U hat. Wenn das Protokoll erfolgreich abgeschlossen ist, ist der Verifizierer davon überzeugt, dass der Benutzer die geforderten Eigenschaften besitzt.

  In einigen Situationen ist es nötig, zu verifizieren, ob das Credential zu einer bestimmten Person gehört. Dazu kann das Credential mit Bezug auf das Pseudonym, für das das Credential ausgestellt wurde, vorgezeigt werden. Bei diesem Ablauf des Protokolls wird der Organisation bewiesen, dass das beim Vorzeigen verwendete Pseudonym

den gleichen Hauptschlüssel benutzt wie das Pseudonym beim Ausstellen des Credentials. Diese Eigenschaft ist nötig, um die *Fälschungssicherheit* zu gewährleisten und die Weitergabe von Credentials zu erschweren. Wird diese Möglichkeit genutzt, ist klar, dass der Besitzer des Credentials auch derjenige ist, für den das Credential ausgestellt wurde. Denn nur dieser kennt den geheimen Hauptschlüssel. Ferner kann diese Eigenschaft verwendet werden, wenn ein Benutzer zwei verschiedene Credentials vorzeigen muss, etwa Studententicket und Lichtbildausweis. Hier muss er nämlich dann nachweisen, dass beide Credentials für denselben Benutzer (derselben Hauptschlüssel) ausgestellt wurden.

2.8.3.4 Weitere Bemerkungen

In diesem Abschnitt werden einige ergänzende Bemerkungen zu idemix gemacht. Eine dieser Bemerkungen betrifft die *Unverkettbarkeit* von Credentials und der Transaktionen zum Vorzeigen von Credentials. Durch den Einsatz von Zero-Knowledge-Beweisen können verschiedene Verifizierer nicht erkennen, wenn dasselbe Credential mehrfach vorgezeigt wird. Auch wenn dasselbe Credential mehrfach beim selben Verifizierer vorgezeigt wird, ist dieser nicht in der Lage, dies zu erkennen. Auch der zugehörige Aussteller kann die Unverkettbarkeit nicht aufheben, weil das Vorzeigen eines Credentials auch nicht mit dem Pseudonym verkettbar ist, für den das Credential ausgestellt wurde.

Natürlich hindert dies die Verifizierer nicht daran, eine Verkettung von Transaktionen mithilfe der nachgewiesenen Eigenschaften, also dem offengelegten Teil des Credentials, vorzunehmen. Um dies zu verhindern, sollten möglichst keine Informationen vorgezeigt werden, die eine eindeutige Identifizierung von Personen ermöglichen würden. Hierzu gehört etwa die Adresse oder das Geburtsdatum. Ob dies immer möglich ist, ist natürlich auch anwendungsabhängig.

Eine weitere Bemerkung betrifft die Eigenschaften der Fälschungssicherheit. Es ist nicht einfach, zu verhindern, dass ein Benutzer sein Credential weiter gibt. Es muss daher ein Weg gefunden werden, der die Weitergabe so unattraktiv macht, dass Personen eine Weitergabe nicht in Erwägung ziehen. Ein klassischer Weg ist die Verknüpfung einer wichtigen Information von *außen* zum Hauptschlüssel des Benutzers. So könnten etwa die Zugangsdaten zum Bankkonto des Benutzers offengelegt werden, wenn er seine Credentials weiter gibt. Eine andere Möglichkeit wird von idemix realisiert. Die sogenannte *All-Or-Nothing-Übertragbarkeit* impliziert, dass wenn der Benutzer auch nur ein Pseudonym oder Credentials von sich weiter gibt, gleichzeitig auch alle anderen Pseudonyme und Credentials weitergegeben werden. Also werden sofort alle Geheimnisse des Benutzers *innerhalb* des Systems offengelegt [CH02]. In beiden Fällen sollten die Benutzer ihre Credentials aus eigenem Interesse nicht weitergeben.

Die letze Bemerkung betrifft die Möglichkeit, die Häufigkeit der Verwendbarkeit eines Credentials einzuschränken. Ein Spezialfall sind *One-Show-Credentials*, die nur einmal verwendet werden können. Danach werden sie ungültig. Dem Verifizierer bietet idemix zwei Möglichkeiten, um die Gültigkeit eines Credentials zu überprüfen. Bei der *Online-Überprüfung* kontaktiert der Verifizierer den Aussteller des Credentials unmittelbar beim

Vorzeigen, um die Gültigkeit sicherzustellen. Bei der *Offline-Überprüfung* wird der Aussteller periodisch kontaktiert, also nachdem das Credential schon verwendet wurde. Im letzteren Fall können Missbrauchsfälle erst a-posteriori geahndet werden.

2.9 Zusammenfassung

In diesem Kapitel wurden grundlegende Begriffe, Konzepte und Techniken vorgestellt, die für das Verständnis des Lösungsansatzes benötigt werden.

Zuerst wurden in Abschnitt 2.1 einige allgemeine Begriffe und Konzepte vorgestellt. Hierbei wurde gezeigt, dass immer dann wenn Personalisierung stattfindet, *Benutzermodelle* zum Einsatz kommen sollten. Ferner wurde der Begriff des *kontextbezogenen Computing* eingeführt und es wurden Anforderungen an Rahmenwerke aufgezählt, die kontextbezogene Anwendungen unterstützen. Weiterhin wurde der Begriff der *komponentenbasierten Softwareentwicklung* beschrieben. Dieser Abschnitt wurde mit der Beschreibung der im Rahmen dieser Arbeit eingesetzten Komponentenplattform *OSGi* abgeschlossen.

In Abschnitt 2.2 wurden *eHomes* und *eHome-Systeme* vorgestellt. Dabei wurde zunächst die Systemstruktur von eHomes beschrieben. Daraufhin wurden am Beispiel einer typischen Dienstkomposition verschiedene Arten von eHome-Diensten diskutiert. Dabei wurde unterschieden zwischen Basisdiensten, integrierenden Diensten und Top-Level-Diensten. Eine weitere Unterscheidung erlaubt die Unterteilung von personalisierbaren und nicht-personalisierbaren Diensten. Danach wurden in Abschnitt 2.3 eine Reihe von Beispielen für personalisierbare eHome-Dienste gegeben.

Als Nächstes wurde in Abschnitt 2.4 die Thematik der *Identitäten* behandelt. Es wurde gezeigt, was Identitäten sind bzw. sein können und wie sich digitale Identitäten von reellen Identitäten ableiten lassen. Es wurde weiterhin der Zusammenhang zwischen *Anonymität*, *Pseudonymität* und *Zurechenbarkeit* erläutert. Abschließend zu diesem Themenschwerpunkt fand das *Identitätsmanagement* seine Darstellung, welches einen integralen Bestandteil bei der Zielsetzung darstellt, dem Benutzer die Kontrolle über die Herausgabe seiner Daten zu geben.

Im folgenden Abschnitt 2.5 wurden die Begriffe *Datenschutz* und *Privatsphäre* voneinander abgegrenzt. Insbesondere wurde das Recht auf *informationelle Selbstbestimmung* und dessen Bedeutung für den Einzelnen erläutert. Weiterhin wurden *Schutzziele* formuliert, denen bei der Modellierung von sicheren Systemen eine bedeutende Stellung zukommt. Ferner wurde die Integration solcher Schutzziele in das Konzept der *mehrseitigen Sicherheit* beschrieben.

Anschließend wurde in Abschnitt 2.6 das Problem der gemeinsamen Ressourcennutzung durch verschiedene Personen und Prozesse aufgegriffen. Hierbei spielen Zugriffskontrollmechanismen bezüglich Datensicherheit eine wichtige Rolle. Aber auch bei der Gewährleistung von Privatsphäre ist die Zugriffskontrolle ein wichtiger Bestandteil. Neben einer allgemeinen Einführung wurden unterschiedliche Zugriffskontrollmodelle beschrieben. Es wurde auch darauf hingewiesen, dass sowohl die *rollenbasierte Zugriffskontrolle* als auch *Zugriffskontrolllisten* in dieser Arbeit Verwendung finden werden. Abgeschlossen wurde dieser Abschnitt

mit der Erläuterung des Begriffs *Authentifizierung*, weil es eine wichtige Voraussetzung für eine funktionierende Zugriffskontrolle ist.

Ein kurzer Überblick über gängige Verschlüsselungsverfahren, mit denen Sicherheitsziele wie *Vertraulichkeit*, *Integrität* oder *Authentizität* erreicht werden können, wurde in Abschnitt 2.7 gegeben.

Den Abschluss dieses Grundlagenkapitels bildet Abschnitt 2.8. In diesem Abschnitt wurden *anonyme Credentials* vorgestellt. Durch ihre *Eigenschaften* können sie zwei scheinbar widersprüchliche Ziele miteinander vereinbaren, nämlich Authentifizierung und Schutz der Privatsphäre. Nach einem Überblick über bekannte Credential-Systeme wurde *idemix* detaillierter vorgestellt, weil es im Rahmen dieser Arbeit eingesetzt wird.

Kapitel 3

Einordnung der Arbeit in den Projektkontext

Dieses Kapitel widmet sich der Einordnung der vorliegenden Arbeit in den Kontext des eHome-Projekts an der Rheinisch-Westfälischen Technischen Hochschule Aachen. Vor Beginn dieser Arbeit wurden in den Arbeiten von Michael Kirchhof [Kir05] und Ulrich Norbisrath [Nor07] grundlegende Konzepte für die Entwicklung von eHome-Diensten sowie für die Einrichtung und den Betrieb von eHomes entwickelt. In einer weiteren Arbeit, die parallel zur vorliegenden durchgeführt wurde, hat sich Daniel Retkowitz [Ret10] mit der Adaptierung von eHome-Diensten beschäftigt. Im Folgenden werden diese Arbeiten näher beschrieben. Anschließend werden sie bzgl. der Erfüllung der Anforderungen bewertet, die in Abschnitt 1.3 diskutiert wurden. Ausgehend von dieser Bewertung werden in den nächsten Kapiteln die im Rahmen dieser Arbeit entwickelten Konzepte vorgestellt.

3.1 Kirchhof: Prozesse und Infrastrukturen

Bislang werden eHomes eher prototypisch in Forschungsprojekten betrachtet und sind selten im privaten Umfeld im Einsatz. Ein Grund für ihre geringe Verbreitung trotz kostengünstiger Geräte, die schon auf dem Markt vorhanden sind, ist ihre *Heterogenität*, die sich in Form unterschiedlicher Ausprägungen von Geräten, Gebäudearchitekturen und Benutzeranforderungen bemerkbar macht. Daraus entsteht ein nicht zu vernachlässigender Aufwand für die Entwicklung und den Betrieb spezifischer Softwarelösungen, der für jedes eHome getrennt entwickelt werden. Dieser Aufwand schlägt sich auch auf die Kosten von eHomes nieder, weswegen der Normalverbraucher bisher nicht in den Genuss von eHomes gekommen ist.

Ausgehend von dieser Situation wurde das eHome-Projekt mit dem Ziel gestartet, Konzepte und Methoden zu entwickeln, die den Weg von eHomes in einen zukünftigen Massenmarkt ebnen.

In einer ersten Arbeit [Kir05] im eHome-Projekt hat sich Michael Kirchhof im Wesentlichen mit zwei Aspekten von eHome-Systemen beschäftigt. Zunächst hat er Basisinfrastrukturen entwickelt, um eine bessere Unterstützung für Dienstentwickler zu erwirken. Zu

diesen Infrastrukturen gehört erstens der Vorschlag, Dienste komponentenbasiert und von Gerätespezifika abstrahierend zu entwickeln, um sie in unterschiedlichen eHomes wiederverwenden zu können. Hierfür hat er das dreischichtige Architekturmodell *PowerArchitecture* entwickelt, das beispielsweise in Abbildung 2.5 dargestellt ist. Zusätzlich hat er einen Ansatz zur deklarativen Entwicklung von eHome-Diensten auf Basis von *Event-Condition-Action-Regeln* entwickelt, der eine einfachere Abbildung von Tagesabläufen auf Dienstfunktionalitäten erlaubt. Dieser Ansatz wurde um einen Mechanismus zur Konflikterkennung erweitert, sodass Benutzer über unterschiedliche Konflikte informiert und zu deren interaktiver Auflösung aufgefordert werden können, wenn mehrere Dienste die gleichen Ressourcen verwenden wollen. Außerdem hat Kirchhof die *Verteilung* von eHome-Diensten unter zwei Gesichtspunkten untersucht. Einerseits hat er die Verteilung innerhalb eines eHomes ermöglicht, um mehrere Gateways in einem eHome einsetzen zu können (s. Abschnitt 2.2). Andererseits hat er die Verteilung zwischen einem Anbieter und eHomes ermöglicht, um eHomes über ein Webportal des Anbieters fernsteuern zu können. Während die erste Art der Verteilung a-posteriori unabhängig von der Dienstentwicklung durchgeführt werden kann, muss die zweite Art der Verteilung schon a-priori von Dienstentwicklern berücksichtigt werden.

Ferner hat Kirchhof den Geschäftsprozess, der zwischen Anbietern und Kunden abläuft, analysiert und formalisiert. Während Anbieter Unternehmen sind, die eHome-Dienste vertreiben, sind Kunden die Besitzer oder Bewohner von eHomes, die eHome-Dienste für eine bestimmte Nutzungsdauer entgeltlich abonnieren und nutzen. Dabei umfasst der Geschäftsprozess, auch als *eHome-Prozess* bezeichnet, den gesamten Lebenszyklus von eHome-Diensten, von der Entwicklung über die Auslieferung bis hin zur Deinstallation. Das Hauptziel war die Entwicklung eines Prozesses, der von der Heterogenität von eHomes abstrahiert und für alle eHomes anwendbar ist. Für die Durchführung des eHome-Prozesses wurden zudem Werkzeuge und ihre Funktionalitäten identifiziert, die zur Unterstützung von Kunden und Anbietern in ihren Aktivitäten eingesetzt werden können.

3.2 Norbisrath: SCD-Prozess

Einen Ansatz zur Reduktion des Aufwandes für die Entwicklung und Wartung von eHomes durch die Wiederverwendung von eHome-Diensten hat Norbisrath in seiner Arbeit [Nor07] verfolgt. Hierfür hat er einen Prozess beschrieben, der den klassischen Entwicklungsprozess für eHomes ersetzen soll. Dieser Prozess wird als *SCD-Prozess* bezeichnet und umfasst die Phasen *Spezifizierung* von Diensten und eHomes sowie die *Konfigurierung* und das *Deployment* von Diensten. Im Folgenden wird erläutert, wie sich der SCD-Prozess vom klassischen Entwicklungsprozess unterscheidet. Anschließend werden die von Norbisrath entwickelten Werkzeuge beschrieben, die zur Unterstützung dieses Prozesses dienen.

3.2.1 Vom klassischen Entwicklungsprozess zum SCD-Prozess

Der klassische Entwicklungsprozess für eHomes zeichnet sich dadurch aus, dass dieser für jeden Kunden individuell durchgeführt wird (s. Abbildung 3.1). In der *Anforderungsana-*

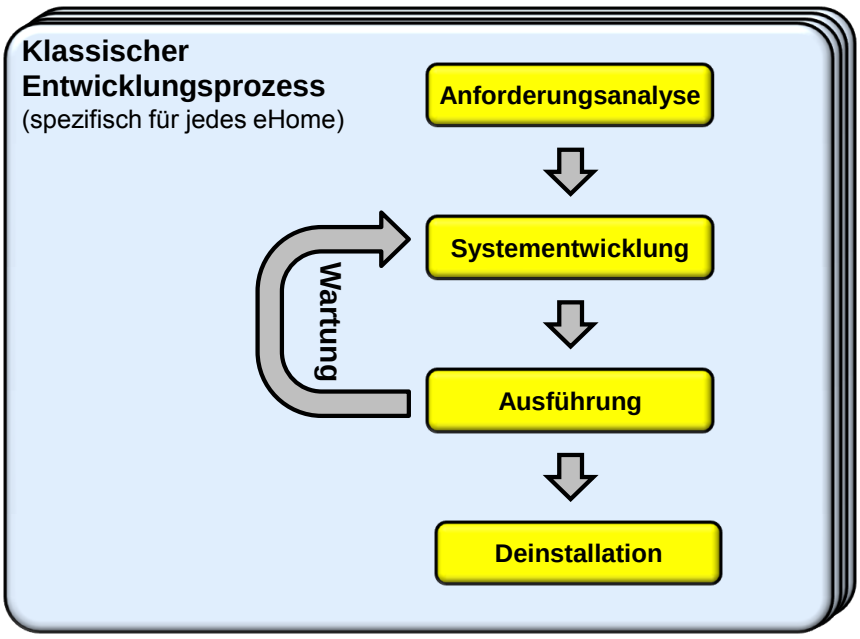


Abbildung 3.1: Klassischer Entwicklungsprozess (angelehnt an [Nor07]).

lyse werden die Kundenwünsche und Gegebenheiten des eHomes (Gebäudearchitektur, vorhandene Geräte usw.) mithilfe eines Spezialisten ermittelt. Nach Erstellung der Anforderungsspezifikation, die gewöhnlich für jeden Kunden unterschiedlich ausfällt, wird in der Phase Systementwicklung eine Lösung entwickelt. Diese Lösung besteht zum einen aus der Vernetzung vorhandener und eventuell noch zu beschaffender Geräte. Zum anderen besteht sie aus Software, die auf Basis der Kopplung der Funktionalitäten dieser Geräte Mehrwertdienste realisiert. Das Ergebnis ist ein System, das sich nach der Inbetriebnahme in Ausführung befindet.

Diese Vorgehensweise zur Entwicklung kundenspezifischer Lösungen führt zu festverdrahteten Systemen, die auf einzelne eHomes zugeschnitten sind und, aufgrund der Heterogenität, nur mit großem Aufwand und hohen Kosten auf andere eHomes portierbar sind. Dadurch findet kaum Wiederverwendung von Software über eHome-Grenzen hinweg statt. Die Entwicklung monolithischer Systeme (s. Abschnitt 2.1.3) erschwert zudem die Wartung des Systems, wenn etwa neue Funktionalitäten dazu kommen oder Geräte ausgetauscht werden.

Um einen möglichst hohen Grad an Wiederverwendung zu erzielen, werden eHome-Dienste komponentenbasiert entwickelt. Als Komponentenmodell wird das weitverbreitete OSGi (s. Abschnitt 2.1.4) eingesetzt. Der komponentenbasierte Ansatz ermöglicht den Übergang vom klassischen Entwicklungsprozess zum komponenten- und konfigurierungsbasierten *SCD-Prozess*. Der Name leitet sich, wie auch oben erwähnt, aus den drei Hauptphasen des Prozesses ab, nämlich *Spezifizierung*, *Konfigurierung* und *Deployment*. Die Voraussetzung für diesen Prozess, nämlich die *Abstraktion* von Geräte- und Protokollspezifika mithilfe von Standardkomponenten und -schnittstellen, ermöglicht die Bewältigung der Heterogenität. Infolgedessen können (integrierende) eHome-Dienste kostengünstig in einer Vielzahl von eHomes eingesetzt werden. Im Folgenden werden die Phasen des SCD-Prozesses beschrieben.

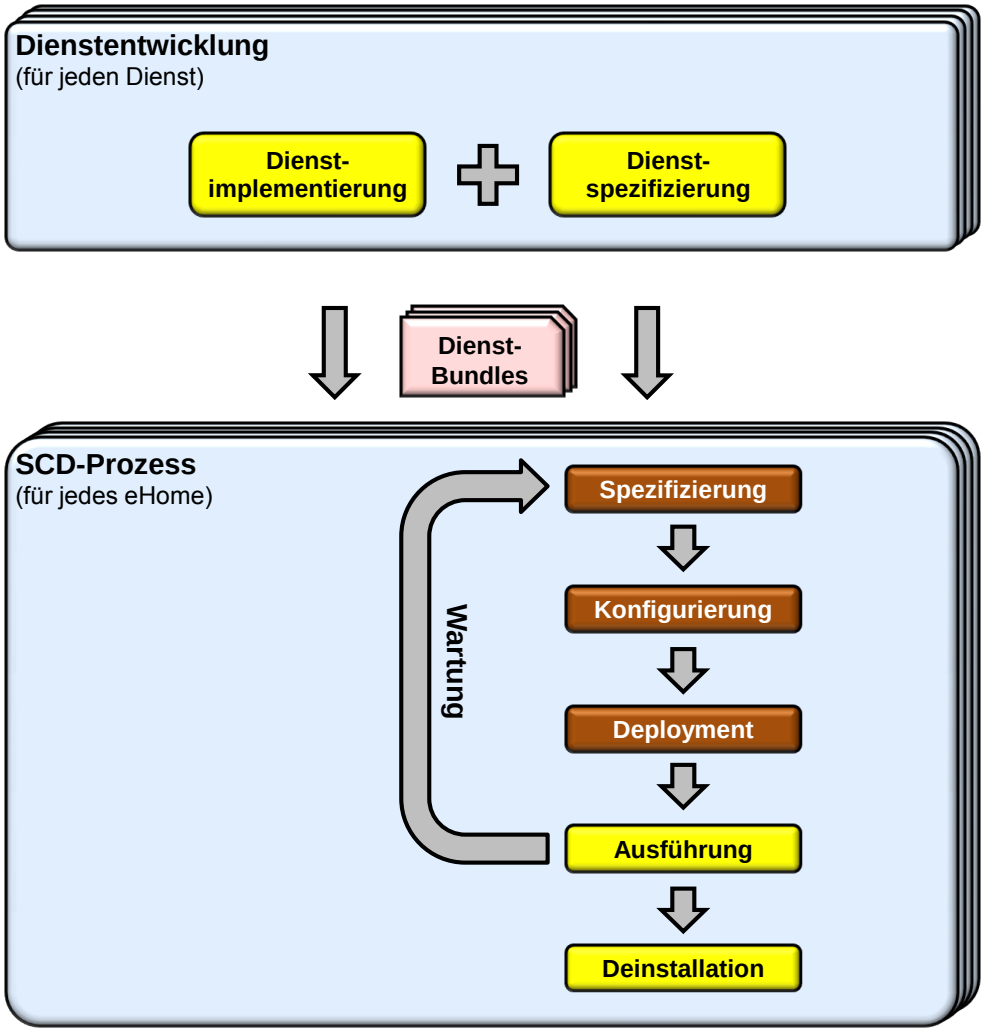


Abbildung 3.2: SCD-Prozess nach Norbisrath (angelehnt an [Nor07]).

3.2.1.1 Dienstentwicklung

Abbildung 3.2 zeigt den SCD-Prozess nach Norbisrath. Im oberen Teil ist die Phase der *Dienstentwicklung* dargestellt, die sowohl die Dienstimplementierung als auch die Dienstspezifizierung enthält. In dieser Phase werden Softwarekomponenten von einem oder mehreren Anbietern entwickelt, die später als eHome-Dienste eingesetzt werden. Diese Komponenten realisieren also die Dienstfunktionalitäten für eHomes. Die Dienstentwicklungsphase ist dienstspezifisch, wird also für jeden Dienst individuell durchgeführt.

Der neue Prozess zeichnet sich dadurch aus, dass die Dienstentwicklung im Gegensatz zum klassischen Prozess von den spezifischen Ausprägungen einzelner eHomes entkoppelt ist. Dadurch erhalten die Dienste einen allgemeineren Charakter und können ohne Anpassung in mehreren eHomes wiederverwendet werden. Um eHome-spezifische Anforderungen trotzdem zu erfüllen, können die Dienste, die zur Realisierung der gewünschten Funktionalitäten geeignet sind, ausgewählt und komponiert werden. Das Ergebnis ist eine Komposition von Diensten, die wie in Abbildung 2.5 dargestellt aussehen kann.

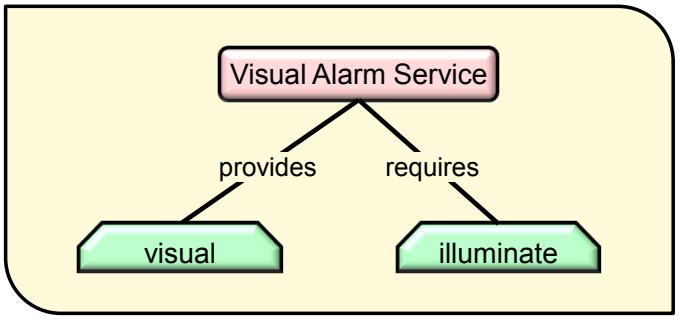


Abbildung 3.3: Beispiel einer Dienstspezifikation (angelehnt an [Nor07]).

Eine wichtige Voraussetzung für die Wiederverwendung von Diensten ist der Einsatz der gleichen Plattform in allen eHomes. In dieser Arbeit wird wie im gesamten Kontext des eHome-Projekts angenommen, dass OSGi als Komponentenplattform eingesetzt wird. Daher werden die Dienstimplementationen in OSGi-Bundles verpackt (s. Abschnitt 2.1.4 und Abbildung 3.2). Bevor die Dienste in einem eHome in Betrieb genommen werden können, müssen sie möglicherweise mit weiteren Komponenten komponiert werden. Die Komponierung könnte manuell durchgeführt werden. In diesem Fall würde aber der Aufwand für die Komponierung eine weitere Hürde für eine Verbreitung von eHomes darstellen, obwohl durch die Wiederverwendung der Entwicklungsaufwand für die Komponenten nur einmalig anfällt.

Um dieser Problematik zu begegnen, hat sich Norbisrath mit der Entwicklung von Werkzeugen beschäftigt, die die automatische Komponierung von Diensten ermöglichen. Damit eine automatische Komponierung durchgeführt werden kann, muss jeder Dienst mit einer *Dienstspezifikation* ausgeliefert werden. In solch einer Spezifikation sind beispielsweise Informationen über die *Funktionalitäten* enthalten, die ein Dienst benötigt (*requires*) und anbietet (*provides*). Abbildung 3.3 zeigt ein Beispiel für eine Dienstspezifikation. Dargestellt ist ein Dienst namens `Visual Alarm Service`, der selbst die Funktionalität `visual` anbietet und für ihre Realisierung die Funktionalität `illumination` benötigt. Dieser Dienst kann also einen visuellen Alarm abgeben, um die Benutzer über bestimmte Ereignisse zu informieren. Seine Funktionalität könnte z. B. von dem `Klingeldienst` (s. Abschnitt 2.3.5) benutzt werden, um die Bewohner visuell darüber zu benachrichtigen, dass geklingelt wurde.

Die Dienstspezifikation wird in der Phase *Dienstspezifizierung* mithilfe des von Norbisrath entwickelten Werkzeugs *eHomeConfigurator* erstellt und zusammen mit der Dienstimplementation in dem zugehörigen OSGi-Bundle ausgeliefert. Diese Dienstspezifikation dient später als Grundlage für die automatische Komponierung mit anderen Diensten.

Die Entkopplung der Dienstentwicklung vom eHome-spezifischen Teil des Prozesses hat auch Einfluss auf die Art der Anforderungsermittlung. Während im klassischen Prozess die Anforderungen für jedes eHome separat ermittelt wurden, ist dies im SCD-Prozess nicht mehr der Fall. Im letzteren Fall ist auch die Anforderungsermittlung zweigeteilt. In der dienstspezifischen Phase werden allgemeine Anforderungen für Funktionalitäten erhoben, die generell in eHomes benötigt werden könnten. Die Ermittlung der kundenspezifischen

Anforderungen, zu denen auch die Gebäudearchitektur und die Art der gewünschten Funktionalitäten zählen, findet hingegen im nachgelagerten eHome-spezifischen Teil statt.

3.2.1.2 Spezifizierung

Diese Phase zur Ermittlung kundenspezifischer Anforderungen wird als *Spezifizierung* bezeichnet und bildet die erste Phase des eHome-spezifischen Teils des SCD-Prozesses (s. Abbildung 3.2). Sie sollte nicht mit der vorgelagerten Dienstspezifizierung verwechselt werden. Zunächst werden in dieser Phase die physikalischen Umstände des eHomes ermittelt. Hierzu zählen Informationen über den Grundriss und über die vorhandenen Türen und Fenster in dem eHome. Diese Informationen werden auch als statischer Umgebungskontext bezeichnet. Dynamischer Umgebungskontext wie Temperatur, Beleuchtung und Luftfeuchtigkeit werden zur Laufzeit von Sensoren erfasst und sind daher nicht Gegenstand dieser Phase. Ferner wird in dieser Phase ermittelt, welche netzwerkfähigen Geräte in der Umgebung schon vorhanden sind und in welchen Räumen sich diese Geräte konkret befinden (*Gerätekontext*). Als Nächstes werden die Funktionalitäten ermittelt, die der Kunde in seinem eHome nutzen möchte. Dies kann durch die Auswahl von verfügbaren Top-Level-Diensten aus dem Sortiment des Anbieters sowie die Angabe geschehen, in welchen Räumen die Funktionalitäten verfügbar sein sollen. Ausgehend von diesen Informationen kann dem Kunden empfohlen werden, eventuell fehlende Geräte zu beschaffen. Die ermittelten Informationen werden in der sogenannten *eHome-Spezifikation* zusammengefasst, die im Wesentlichen die eHome-spezifischen Anforderungen enthält.

Auch die Erstellung der eHome-Spezifikation wird durch den eHomeConfigurator unterstützt. Gemeinsam bilden die mit den ausgewählten Diensten gelieferten Dienstspezifikationen und die eHome-Spezifikation die Grundlage für die nächste Phase.

3.2.1.3 Konfigurierung

Der Spezifizierung des eHomes folgt die Phase der *Konfigurierung*. Die Konfigurierung lässt sich weiter unterteilen in Komponierung und Parametrisierung. Die Aufgabe dieser Phase ist die größtenteils automatische Erstellung einer Konfiguration. Sie enthält alle Informationen, die für die Inbetriebnahme des eHomes nötig sind und als Grundlage für die Ausführung der ausgewählten Dienste dienen.

Im Rahmen der *Komponierung* werden die Abhängigkeiten der ausgewählten Top-Level-Dienste bis auf die Geräteebene aufgelöst. Dazu wird analysiert, welche Funktionalitäten von den ausgewählten Top-Level-Diensten benötigt werden. Als Nächstes werden Dienste ermittelt, die diese Funktionalitäten anbieten. Falls solche Dienste gefunden wurden, werden sie mit dem Top-Level-Dienst verbunden. Dieser Vorgang wird im nächsten Schritt für die neu ermittelten Dienste wiederholt, falls in deren Spezifikation auch benötigte Funktionalitäten angegeben sind. Erst wenn alle Pfade bis zur untersten Ebene der Basisdienste durchlaufen sind, ist die Komponierung der Dienste beendet. Dabei sollte beachtet werden, dass Basisdienste die einzigen Dienste sind, die konkret die im eHome existierenden Geräte steuern und ihre Funktionalitäten auf einer abstrakten Ebene anderen Diensten zur Verfügung stellen. Zusammengefasst beschreibt das Ergebnis der Komponierung, die *Dienstkomposition*, wie

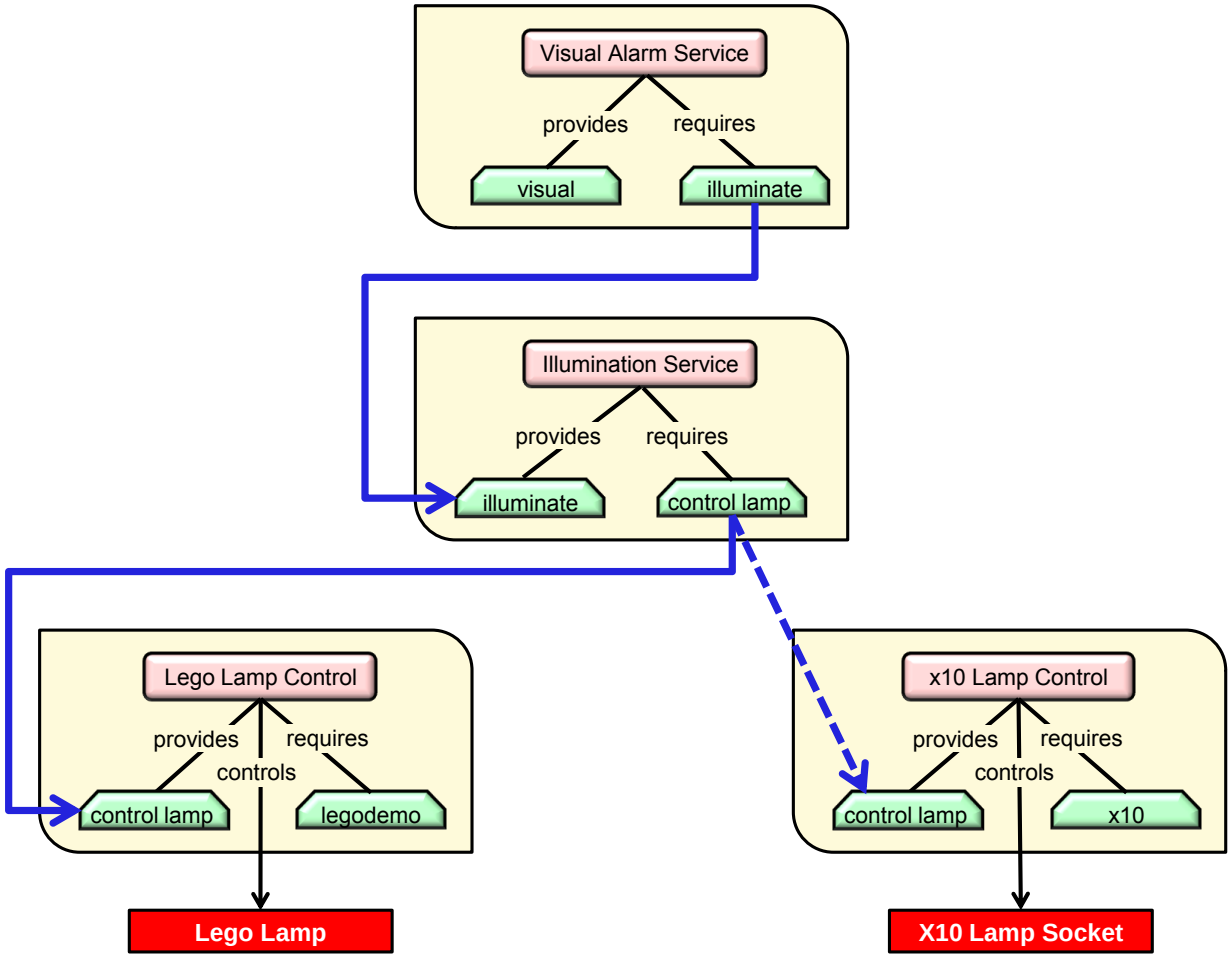


Abbildung 3.4: Beispiel einer Dienstkomponierung (angelehnt an [Nor07]).

die Dienste miteinander verbunden sind, welchen Räumen sie zugeordnet sind und welche Basisdienste welche Geräte ansteuern.

Ein Beispiel für die Komponierung von Diensten auf Grundlage von Dienstspezifikationen ist in Abbildung 3.4 dargestellt. Als Top-Level-Dienst ist der oben beschriebene Dienst `Visual Alarm Service` abgebildet. Auf der nächsten Ebene ist der integrierende Dienst `Illumination Service` dargestellt, der die von dem `Visual Alarm Service` benötigte Funktionalität `illuminate` anbietet. Der `Illumination Service` wiederum benötigt die Funktionalität `control lamp`, die von zwei Basisdiensten, nämlich `Lego Lamp Control` und `x10 Lamp Control`, angeboten wird. In dieser Situation kann keine automatische Komponierung ohne Benutzerinteraktion durchgeführt werden. In so einem Fall bestimmt der Benutzer interaktiv, welche Alternative ausgewählt werden soll. Im obigen Beispiel wurde der Basisdienst `Lego Lamp Control` ausgewählt, indiziert durch die durchgezogene Linie, weil in diesem Beispiel eine Komponierung für einen aus Lego gebauten Demonstrator durchgeführt wurde (s. Abschnitt 8.3.2 und [Nor07]). Die Auswahl von alternativen Unterdiensten könnte auch automatisch auf Basis von *Quality-of-Service*-Kriterien durchgeführt werden. Solch

ein Ansatz wurde im eHome-Projekt bisher jedoch nicht verfolgt, ist aber ein mögliches Thema für zukünftige Arbeiten.

Als Nächstes folgt die *Parametrisierung* der komponierten Dienste. Dadurch werden die Funktionalitäten der Dienste an die Bedürfnisse und Wünsche der Benutzer angepasst. Die Parameter, über die das Dienstverhalten beeinflussbar ist, müssen zu diesem Zweck in der Dienstspezifikation angegeben werden (*Formalparameter*, s. [Nag90]). Sowohl vor der Inbetriebnahme als auch zur Laufzeit können diese Parameter mit dem eHomeConfigurator bearbeitet, also mit benutzerspezifischen Werten belegt werden (*Aktualparameter*). Beispielsweise kann über entsprechende Parameter eingestellt werden, an welche Adresse der `Sicherheitsdienst` in einer Gefahrensituation eine E-Mail senden oder welche Musik der `Weckdienst` morgens spielen soll. Wie dieses Beispiel zeigt, können neben allgemeinen Parametern auch Parameter angegeben werden, über die das Dienstverhalten an einzelne Benutzer angepasst werden kann. Mit anderen Worten ist die *Personalisierung* von eHome-Diensten eine spezielle Art der Parametrisierung und wird nach Norbisrath durch Parameter vorgenommen, die in der Dienstspezifikation angegeben sein müssen.

Durch die Parametrisierung wird die Dienstkomposition um weitere Informationen angereichert, sodass der Begriff der Komposition zu speziell ist und diesen Sachverhalt nicht umfasst. Aus diesem Grund werden die Phasen Komponierung und Parametrisierung als *Konfigurierung* zusammengefasst. Das Ergebnis der Konfigurierung ist eine Graphdatenstruktur, die als *eHome-Konfiguration* bezeichnet wird. Im weiteren Verlauf dieser Arbeit werden die Begriffe Konfiguration und Komposition oft synonym verwendet. An den Stellen, an denen eine konkrete Unterscheidung nötig ist, wird darauf hingewiesen.

Die gesamte Phase der Konfigurierung, und damit auch die Erstellung der Konfiguration, wird von dem eHomeConfigurator unterstützt. Bis auf die Parametrisierung und die interaktive Auswahl von alternativen Unterdiensten wird die Konfigurierung automatisch durchgeführt. Um die Automatisierung gegenüber der manuellen Konfigurierung abzugrenzen, verwendet Norbisrath für die werkzeuggesteuerte Konfigurierung den Begriff der *generativen Konfigurierung* [Nor07].

3.2.1.4 Deployment

Nachdem die Konfigurierung beendet wurde, wird in der nächsten *Deployment*-Phase die Konfiguration in Ausführung gebracht. Die Konfiguration enthält alle Informationen für das Deployment (Installation und Instanziierung) der ausgewählten Dienste. Das Deployment wird von dem eHomeConfigurator auf Basis der Konfiguration automatisch durchgeführt. Genauer gesagt wird diese Aufgabe von dem Deployer durchgeführt, der eine Teilfunktionalität des eHomeConfigurator realisiert. Wichtig ist die Berücksichtigung der Dienstabhängigkeiten beim Deployment. Während die Komponierung der Dienste *Top-down* durchgeführt wurde, muss das Deployment *Bottom-up* durchgeführt werden, damit alle benötigten Dienste auf der Plattform schon verfügbar sind, bevor ein integrierender Dienst gestartet wird.

Während des Deployments wird die Konfiguration noch mal um weitere Informationen angereichert. Besonders nennenswert ist die Erstellung von Objekten zur Abbildung von Zuständen und Attributen, über die das Verhalten von Diensten beeinflusst werden kann.

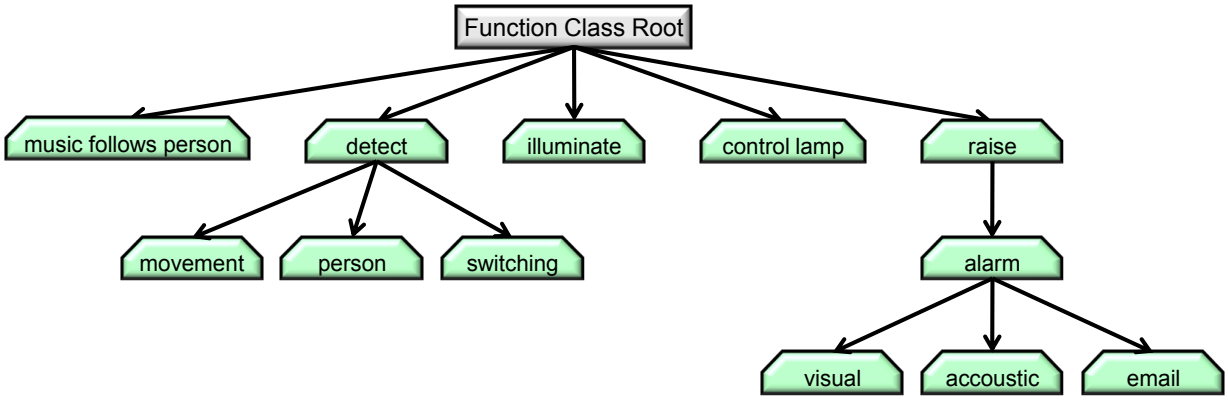


Abbildung 3.5: Funktionalitäten von eHome-Diensten (angelehnt an [Nor07]).

3.2.1.5 Ausführung und Deinstallation

Nach dem Deployment befindet sich das eHome nun in der *Ausführung*. Falls die Dienste nicht mehr benötigt werden, können sie vom System entfernt werden (*Deinstallation*). Es ist aber auch möglich, bestehende Dienste zu aktualisieren oder neue Dienste zu installieren. Die Aktualisierung oder Installation von Diensten kann zur erneuten Ausführung des SCD-Prozesses führen, wie der mit *Wartung* annotierte Pfeil aus Abbildung 3.2 andeutet. Dafür muss das System zunächst jedoch runtergefahren werden, bevor Änderungen an der Konfiguration vorgenommen werden können. In Abschnitt 3.3 wird beschrieben, welchen Ansatz Retkowitz in seiner Arbeit verfolgt hat, um zur Laufzeit des eHomes Änderungen an der Konfiguration vorzunehmen, ohne das System runterfahren zu müssen.

3.2.2 Das eHome-Modell

Für die Unterstützung des SCD-Prozesses hat Norbisrath [NARS06] das *eHome-Modell* entwickelt. Es fasst alle Informationen zusammen, die der eHomeConfigurator zur Unterstützung des eHome-Prozesses, speziell auch für den eingebetteten SCD-Prozess, benötigt. Das eHome-Modell wurde mithilfe von *Fujaba* entwickelt. Fujaba ist ein Werkzeug, das die modellbasierte Entwicklung von Softwaresystemen ermöglicht [FNTZ98]. Das eHome-Modell enthält sowohl statische als auch dynamische Festlegungen. Die dynamischen Festlegungen sind in Form von Story Diagrammen [FNTZ98] modelliert und sind Teil der Funktionalitäten des eHomeConfigurator.

Die statischen Festlegungen werden in Form von UML-Klassen-Diagrammen modelliert und beschreiben die Klassen des Modells und ihre Beziehungen untereinander. Zur Laufzeit wird dann eine Instanz des Modells erzeugt, die Objekte der modellierten Klassen enthält. Dabei wird das eHome-Modell für jedes eHome einmal instanziiert. Die Modellinstanz hat zur Laufzeit des eHomes zwei Aufgaben. Einerseits dient sie dem eHomeConfigurator als Grundlage für die Umsetzung des SCD-Prozesses, weil alle hierfür benötigten Informationen in dieser abgelegt werden. Andererseits fungiert sie auch als *Kommunikationsmedium*

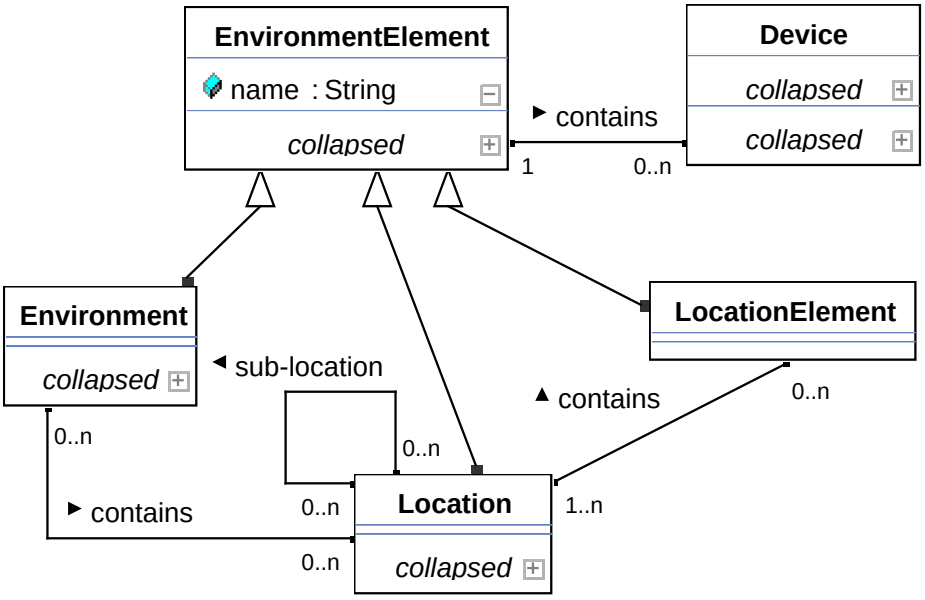


Abbildung 3.6: Modellierung von Umgebungsinformationen (Quelle: [NARS06]).

für eHome-Dienste, weil sie alle zur Laufzeit benötigten Kontextinformationen enthält. Ein ähnlicher Ansatz wird beispielsweise auch in [AGGH06] verfolgt. Im Folgenden wird beschrieben, welche Arten von Informationen das eHome-Modell umfasst.

3.2.2.1 Funktionalitäten

Einer der wichtigsten Aspekte des eHome-Modells ist die Modellierung von Funktionalitäten, die später für die Komponierung von Diensten benötigt werden. Funktionalitäten werden dabei durch die Function-Klasse modelliert, die eine reflexive Relation besitzt. Dadurch können Funktionalitäten hierarchisch verfeinert werden. Ein Ausschnitt aus der Funktionalitätshierarchie nach Norbisrath ist in Abbildung 3.5 gezeigt. Beispielsweise ist die Funktionalität `detect` dargestellt, die durch drei weitere Funktionalitäten `movement`, `person` und `switching` verfeinert wird. Sie können bei der Dienstspezifizierung verwendet werden, um anzugeben, welche Funktionalitäten Dienste benötigen oder anbieten. Ein *Personendetektor* z. B. würde die Funktionalität `person` anbieten. Auf Basis dieser Dienstspezifikationen wird dann die Komponierung der Dienste vorgenommen (s. Abschnitt 3.2.1.3). Die Verfeinerungsbeziehung wird hier nur zur Strukturierung/Klassifizierung von Funktionalitäten verwendet und hat keine weitere semantische Bedeutung.

3.2.2.2 Geräte

Bevor in der Spezifizierungsphase einem eHome Geräte hinzugefügt werden können, müssen diese Geräte vorher spezifiziert werden. Für Gerätetypen sieht das eHome-Modell die Klasse DeviceDefinition vor, über die der Name und weitere Informationen wie der Hersteller des Gerätetyps angegeben werden können. Ein oder mehrere konkrete Geräte dieses Typs können dann als Instanzen der Klasse Device der eHome-Spezifikation hinzugefügt werden.

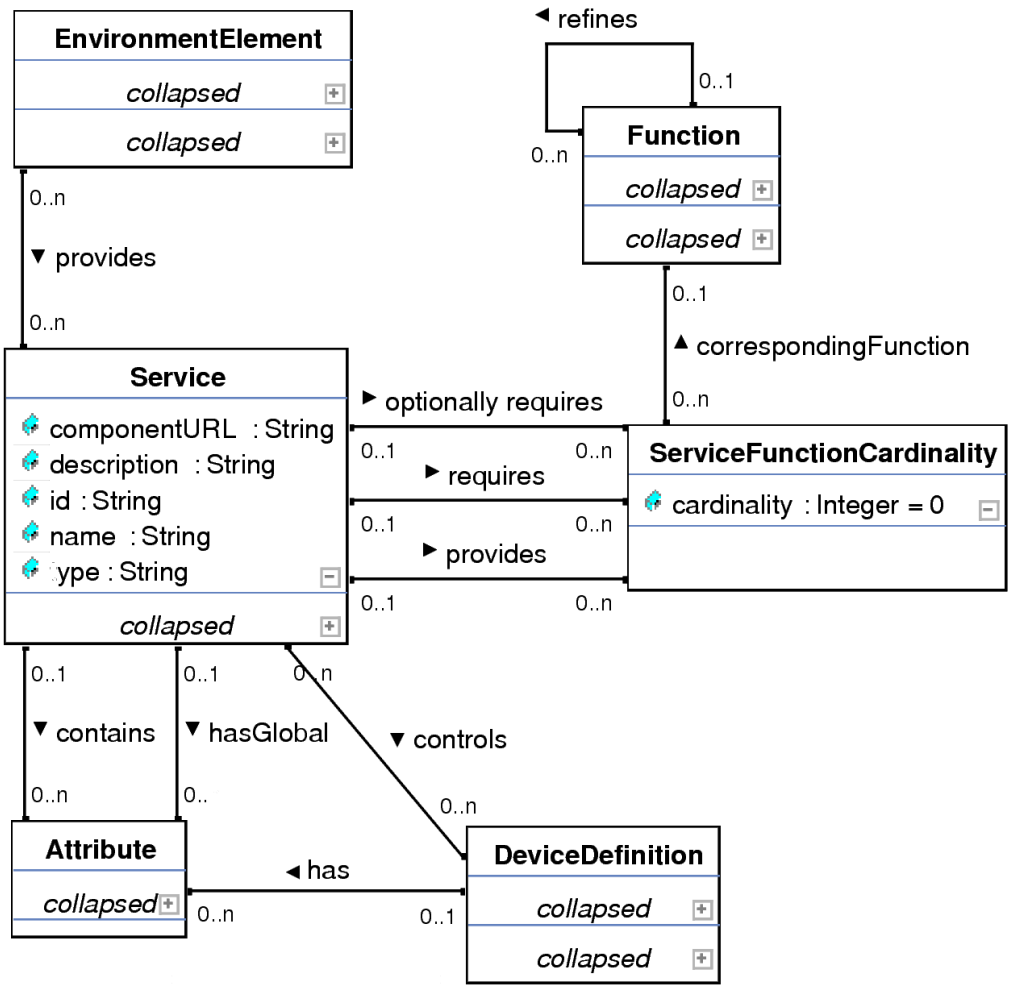


Abbildung 3.7: Modellierung von Diensttypen (Quelle: [NARS06]).

Geräten werden keine Funktionalitäten zugeordnet, weil der SCD-Prozess nur eHome-Dienste berücksichtigt. Um Gerätefunktionalitäten dem SCD-Prozess zugänglich zu machen, werden sie den Basisdiensten zugeordnet, die die Geräte steuern.

3.2.2.3 Umgebungen

Für die Modellierung der Gebäudearchitektur und der in einem eHome vorhandenen Geräte sieht das eHome-Modell weitere Klassen vor, die in Abbildung 3.6 abgebildet sind. Umgebungsinformationen werden von der Oberklasse EnvironmentElement abgeleitet. Ein eHome wird als Instanz der Klasse Environment modelliert. Räume hingegen werden als Instanzen von Location modelliert, Fenster und Türen können auf Instanzen von LocationElement abgebildet werden. Durch die geerbte `contains`-Beziehung zur Klasse Device können alle Bereiche einer Umgebung mit Geräten in Verbindung gesetzt werden. Beispielsweise können Fernseher und Lichtschalter in einem Raum vorhanden sein, während Fenster dagegen mit Glasbruchsensoren ausgestattet sein können.

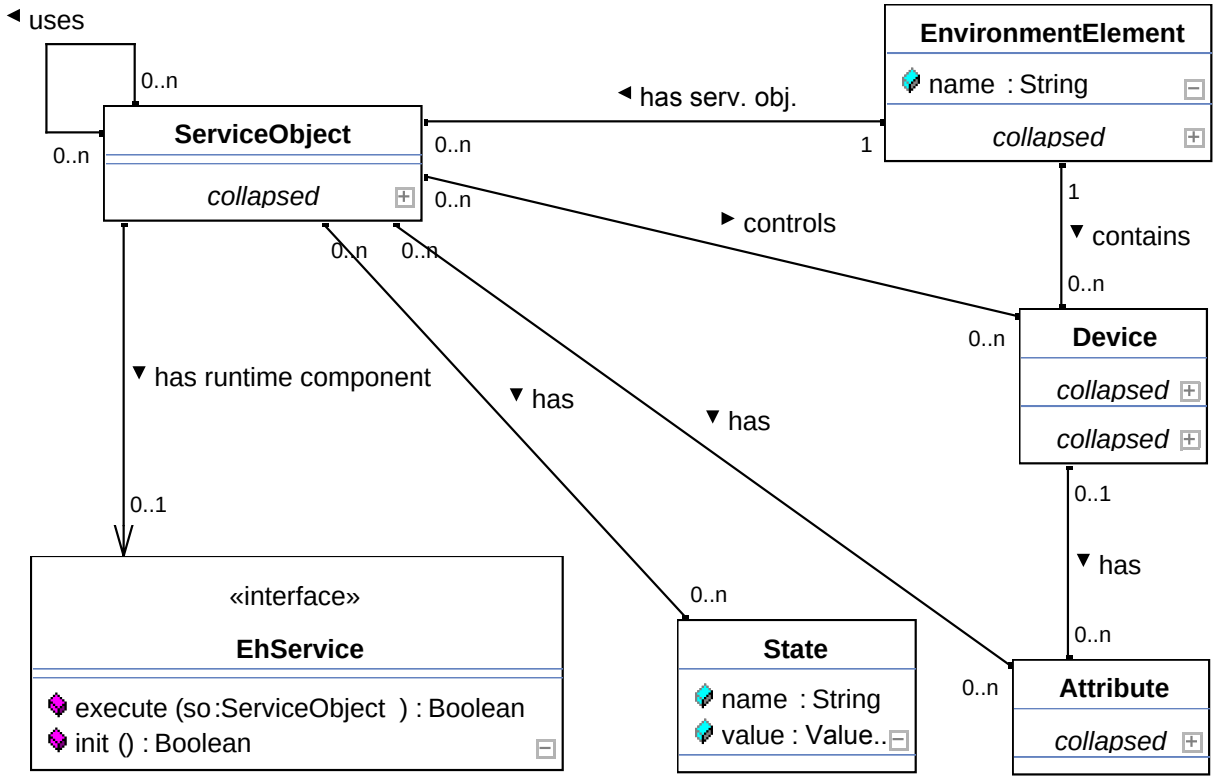


Abbildung 3.8: Modellierung von Konfigurationen (Quelle: [NARS06]).

3.2.2.4 Diensttypen

Für die Spezifikation von eHome-Diensten sieht das eHome-Modell die Klasse Service vor (s. Abbildung 3.7). Neben allgemeinen Informationen wie Name, Typ und Beschreibung enthält diese Klasse indirekte Beziehungen zur Klasse Function. Ein Dienst kann Funktionen anbieten, benötigen und optional benötigen. Die Zwischenklasse ServiceFunctionCardinality dient der Angabe, in welcher Quantität die angegebenen Funktionalitäten benötigt oder angeboten werden. Zum Beispiel könnte ein Dienst, der ein Schaltfeld mit sechs Schaltern steuert, die Funktionalität `switching` sechsfach anbieten. Ferner hat die Klasse Service zwei Beziehungskanten zur Klasse Attribute, die für die Parametrisierung von Diensten eingesetzt werden (s. Abschnitt 3.2.1.3). Dabei sind globale Attribute für alle Instanzen eines Dienstes gültig, während lokale Attribute für jede Instanz unterschiedliche Ausprägungen haben können. Die Beziehung zur Klasse EnvironmentElement deutet daraufhin, dass ein eHome oder ein bestimmter Raum im eHome bestimmte Dienste anbieten. Durch diese Beziehung kann ein Benutzer z. B. vorgeben, dass der `Sicherheitsdienst` im Badezimmer keine Bilder aufnehmen soll, in allen übrigen Räumen aber schon. Durch die Beziehung zur Klasse DeviceDefinition kann für Basisdienste angegeben werden, welche Typen von Geräten durch diese Basisdienste gesteuert werden können.

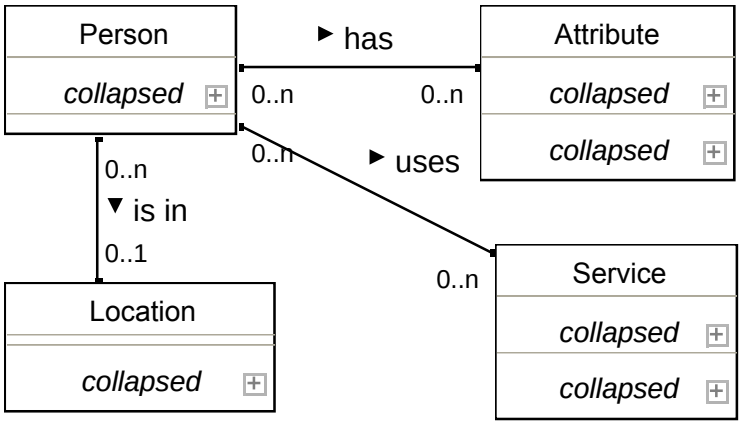


Abbildung 3.9: Modellierung des Personenkontexts (Quelle: [NARS06]).

3.2.2.5 Dienstinstanzen

Während die Klasse Service für die Spezifizierung von *Diensttypen* verwendet wird, wird für die Modellierung der Konfiguration die Klasse ServiceObject eingesetzt (s. Abbildung 3.8). Dabei können von einem Dienst (Service) mehrere Instanzen (ServiceObject) erzeugt werden. Die Grundidee dabei ist die Zuordnung genau einer Instanz zu jedem Raum, in dem der entsprechende Dienst verfügbar sein soll. Aufgrund dieser festen Zuordnung von Diensten zu Räumen auf Modellebene werden solche Dienste in dieser Arbeit auch als *raumgebundene* Dienste bezeichnet. Diese unterscheiden sich von *personengebundenen* Diensten, wie später in Abschnitt 4.3.2 erläutert wird. Die für die Dienstkomposition wichtigste Beziehung der Klasse ServiceObject ist die reflexive Relation `uses`. Über diese Relation wird zur Laufzeit festgehalten, welche Dienste miteinander verbunden sind. Der durch diese Relation aufgebaute Graph ergibt die eHome-spezifische Dienstkomposition. Sie wird im Laufe des SCD-Prozesses mit weiteren Informationen angereichert.

Für Instanzen von Diensten können konkretere Angaben gemacht werden, als es in der Dienstspezifikation möglich ist (s. Abbildung 3.8). Beispielsweise kann nun angegeben werden, welches konkrete Gerät (Device) im aktuellen eHome von einer Dienstinstanz gesteuert wird. Ferner werden konkrete Objekte der Klassen Attribute und State erstellt. Letztere sind wichtig für die *Kommunikation* von Diensten untereinander. Norbisrath setzt voraus, dass miteinander verbundene Dienste über Zustände kommunizieren. Dabei wird implizit angenommen, dass für jede Funktionalität ein oder mehrere Zustände vorhanden sind, die von Dienstentwicklern mit entsprechenden State-Objekten umgesetzt werden. Nur dann, wenn sich die verbundenen Dienste an diese Konvention halten, können sie miteinander fehlerfrei kommunizieren. Die Beziehung zur Schnittstelle EhService stellt eine Verbindung zwischen einem ServiceObjekt auf Modellebene und dem OSGi-Bundle her, das die Dienstfunktionalitäten realisiert (s. Abschnitt 2.1.4). Über die in der Schnittstelle EhService vorgegebenen Methoden `execute` und `init` werden die für das Deployment nötigen Methoden bereitgestellt. Damit kann ein Dienst nach der Konfigurierung durch die Laufzeitumgebung gestartet und initiiert werden.

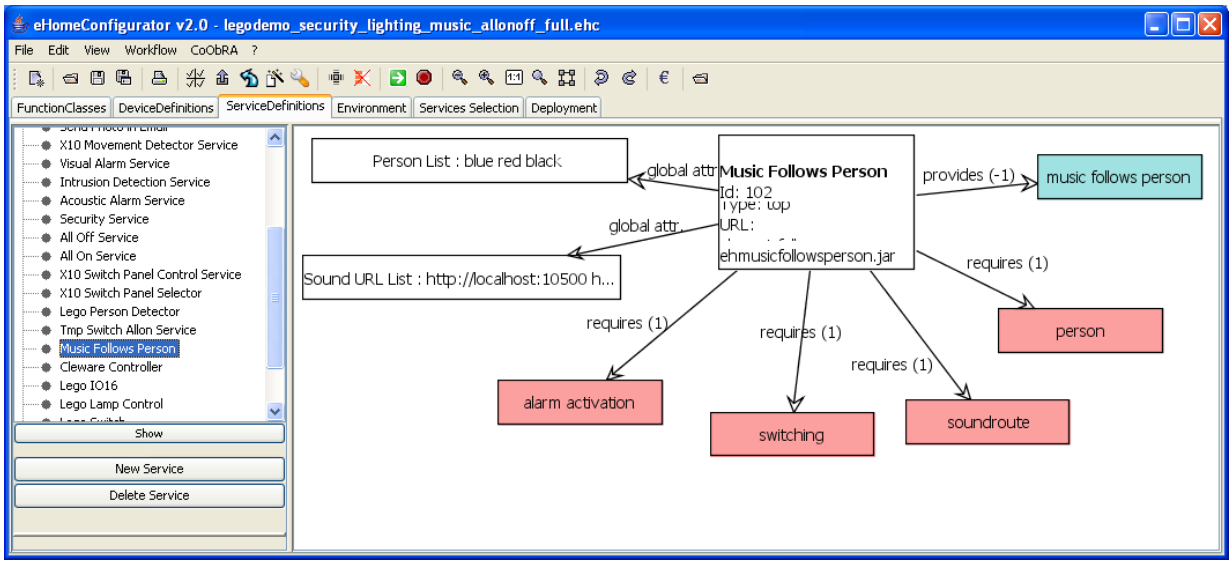


Abbildung 3.10: Spezifikation des Musikdienstes mit dem eHomeConfigurator (Quelle: [Nor07]).

Ein OSGi-Bundle kann als Softwarekomponente mehrere Dienste realisieren. Im eHome-Projekt und damit auch in dieser Arbeit wird jedoch die Konvention eingehalten, dass jedes Bundle nur einen Dienst realisiert, der aber in einem eHome mehrfach instanziiert werden kann. Aus diesem Grund werden die Begriffe *Komponenten* und *Dienst* oft synonym verwendet.

3.2.2.6 Personen

Ein letzter wichtiger Aspekt des eHome-Modells betrifft Personen, oder genauer gesagt, den *Personenkontext* (s. Abschnitt 2.2.4). Abbildung 3.9 zeigt die für die Modellierung des Personenkontexts verwendeten Klassen. Die Klasse Person dient zur Modellierung von Personen. Für jede Person im eHome wird vor Beginn der Konfigurierungsphase eine Instanz dieser Klasse angelegt, sodass Benutzerdaten mit dieser Instanz verknüpft werden können.

Über die Beziehung `is in` zur Klasse Location wird für jede Person festgehalten, in welchem Raum sich eine Person aktuell befindet. Diese Information ist besonders für personalisierbare Dienste wichtig, deren Funktionalität vom Aufenthaltsort einer Person abhängt. Über die `uses`-Beziehung zur Klasse Service wird dargestellt, welche Dienste eine Person verwendet. Schließlich werden die Benutzerdaten einer Person mithilfe der über die `has`-Beziehung assoziierten Klasse Attribute modelliert. Beispielsweise können in einem Attribut der Musikgeschmack und in einem anderen die Temperaturpräferenz einer Person abgelegt werden.

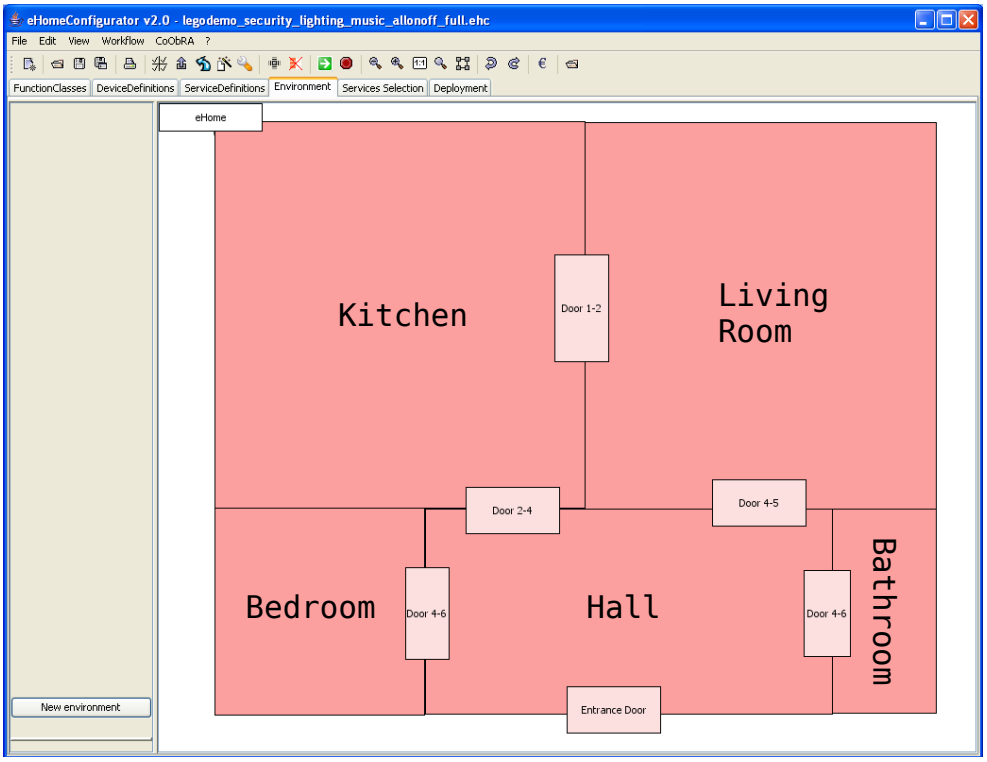


Abbildung 3.11: Spezifikation einer Umgebung mit dem eHomeConfigurator (Quelle: [Nor07]).

3.2.3 Werkzeugunterstützung: der eHomeConfigurator

Norbisrath hat zur Unterstützung des SCD-Prozesses ein Werkzeug namens *eHomeConfigurator* entwickelt. Mit diesem Werkzeug kann neben allen Phasen des SCD-Prozesses auch die Dienstspezifizierung durchgeführt werden. Ein Beispiel hierfür ist in Abbildung 3.10 gegeben. Im Hauptfenster ist die Spezifikation des `Musikdienstes` abgebildet. Neben benötigten und angebotenen Funktionalitäten sind auch zwei Attribute spezifiziert. Das Attribut `Person List` dient zur Angabe der möglichen Personen, die diesen Dienst verwenden werden. Die Namen `blue, red, black` stehen dabei für die Namen der Lego-Männchen, die in einem Lego-Demonstrator eingesetzt wurden. Das Attribut `Sound URL List` dient der Angabe von Musikpräferenzen der genannten Personen.

Auf der linken Seite sind weitere Dienste aufgelistet, die mit dem eHomeConfigurator spezifiziert wurden. Zudem gibt es die Möglichkeit, neue Dienste zu spezifizieren. Neben dem Reiter für Dienstspezifikation sind auch weitere Reiter zur Spezifikation von Funktionalitäten, Geräten und Umgebungen möglich. Abbildung 3.11 zeigt die Spezifikation einer Umgebung Küche, Wohnzimmer, Schlafzimmer, Badezimmer und Flur sowie den zugehörigen Türen. Hier können den Räumen noch Geräte zugeordnet werden. Der Reiter `Deployment` zeigt die aktuelle Konfiguration für ein eHome in Form einer *Graphstruktur* an. Dort sind alle Informationen über die im eHome vorhandenen Geräte und installierten Dienste sowie ihre Komposition und Parametrisierung enthalten. Diese Graphstruktur kann manuell oder mit einem Wizard bearbeitet werden, wie als Nächstes beschrieben wird.

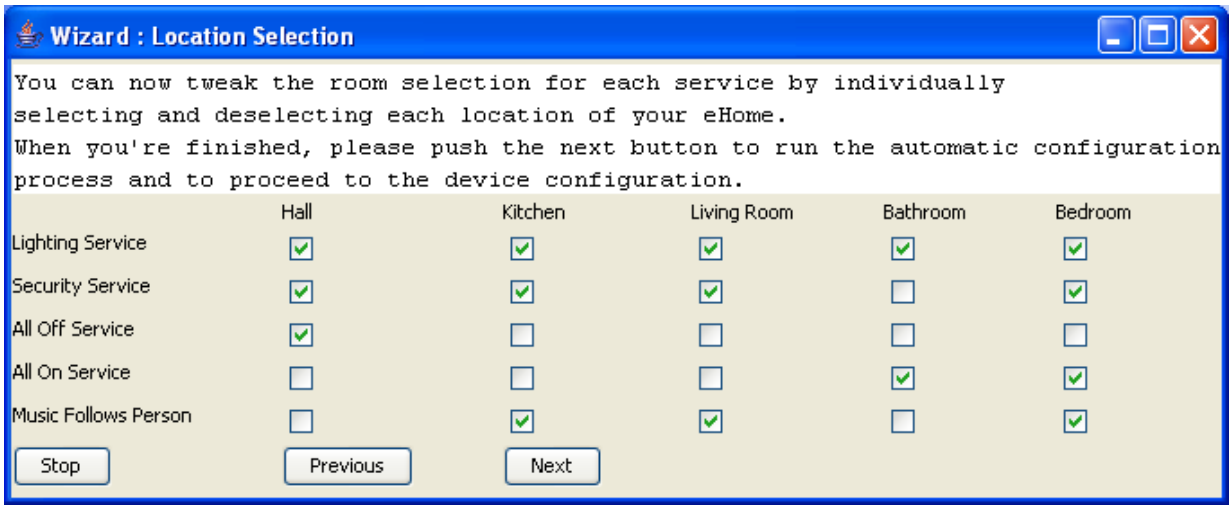


Abbildung 3.12: Auswahl von Diensten mit dem eHomeConfigurator (Quelle: [Nor07]).

Für die Auswahl von Diensten zur Konfigurierung bietet der eHomeConfigurator einen *Wizard* an, der den Benutzer in mehreren Schritten bei dieser Aufgabe begleitet. Abbildung 3.12 zeigt den Wizard zur Auswahl von Diensten und deren Zuordnung zu Räumen, in denen sie verfügbar sein sollen. Beispielsweise soll der `Lighting Service` in jedem Raum verfügbar sein, während der `Security Service` im Badezimmer nicht verfügbar sein soll, weil dort keine Bilder aufgenommen werden sollen.

Auch die Parametrisierung von Diensten findet mithilfe des Wizards statt. Abbildung 3.13 zeigt die Bearbeitung von Parametern für den Dienst `Send Photo in Email`. Die Attribute `mail destination url`, `mail text`, `attachement url` und `mail subject` werden im Falle einer Gefahrensituation für die Versendung eines Fotos an eine voreingestellte E-Mail-Adresse benötigt.

Weitere Details über die Funktionalitäten des eHomeConfigurator können in [Nor07] nachgelesen werden. Im nächsten Abschnitt wird die Arbeit von Retkowitz beschrieben, die auf dem Ansatz von Norbisrath aufsetzt.

3.3 Retkowitz: kontinuierlicher SCD-Prozess

Zwei wichtige Aspekte, die von Norbisrath nicht bearbeitet wurden, sind das Thema der Arbeit von Retkowitz [Ret10]. Diese Arbeit wurde parallel zur vorliegenden Arbeit erstellt. Das Hauptziel seiner Arbeit war die Adaption von eHomes. Dabei unterscheidet Retkowitz zwischen der strukturellen und der semantischen Adaption. Teilweise basieren die hier entwickelten Konzepte auch auf den Ansätzen von Retkowitz, die im Folgenden zusammengefasst wiedergegeben werden.

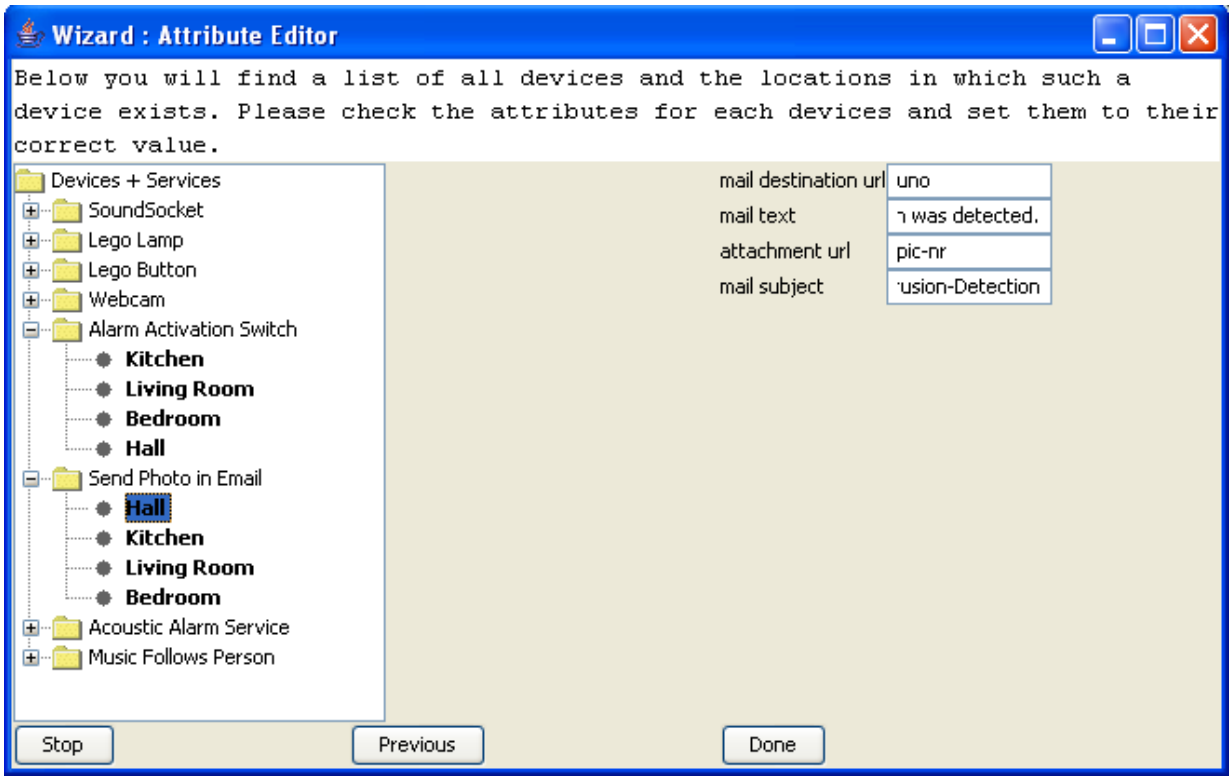


Abbildung 3.13: Parametrisierung von Diensten mit dem eHomeConfigurator (Quelle: [Nor07]).

3.3.1 Strukturelle Adaption

Die *strukturelle Adaption* dient der Anpassung der eHome-Konfiguration an unterschiedliche Ereignisse zur Laufzeit. Diese Ereignisse sind das Ergebnis von Mobilität und Dynamik. Dazu gehören wechselnde Benutzeranforderungen (neue Dienste kommen dazu, oder werden abbestellt) genauso wie die Mobilität von Personen und Geräten innerhalb des eHomes.

Der von Norbisrath entwickelte Ansatz unterstützt nur die einmalige Durchführung des SCD-Prozesses zur Einrichtung von eHomes. Nach dem Deployment können der Konfiguration beispielsweise keine weiteren Dienste hinzugefügt oder aus ihr entfernt werden, ohne das System anzuhalten und den gesamten SCD-Prozess erneut auszuführen. Retkowitz hat den SCD-Prozess zu einem *kontinuierlichen* Prozess erweitert, sodass zur Laufzeit unterschiedliche Anpassungen durchgeführt werden können.

Abbildung 3.14 zeigt diesen kontinuierlichen SCD-Prozess. Es fallen zwei Unterschiede zum klassischen SCD-Prozess auf. Zunächst ist die Umgebungsspezifizierung jetzt ausgelagert. Dies hat den Hintergrund, dass statischer Umgebungskontext wie Grundrissinformationen eines Gebäudes üblicherweise nur einmal aufgenommen werden müssen, weil sie keinen ständigen Veränderungen unterliegen. Dafür ist im kontinuierlichen SCD-Prozess der Fokus auf Dynamikaspekte gerichtet. Hierzu gehören variable Benutzeranforderungen, wie das Hinzufügen oder Entfernen von Diensten, und eine variable Geräteumgebung, die sich

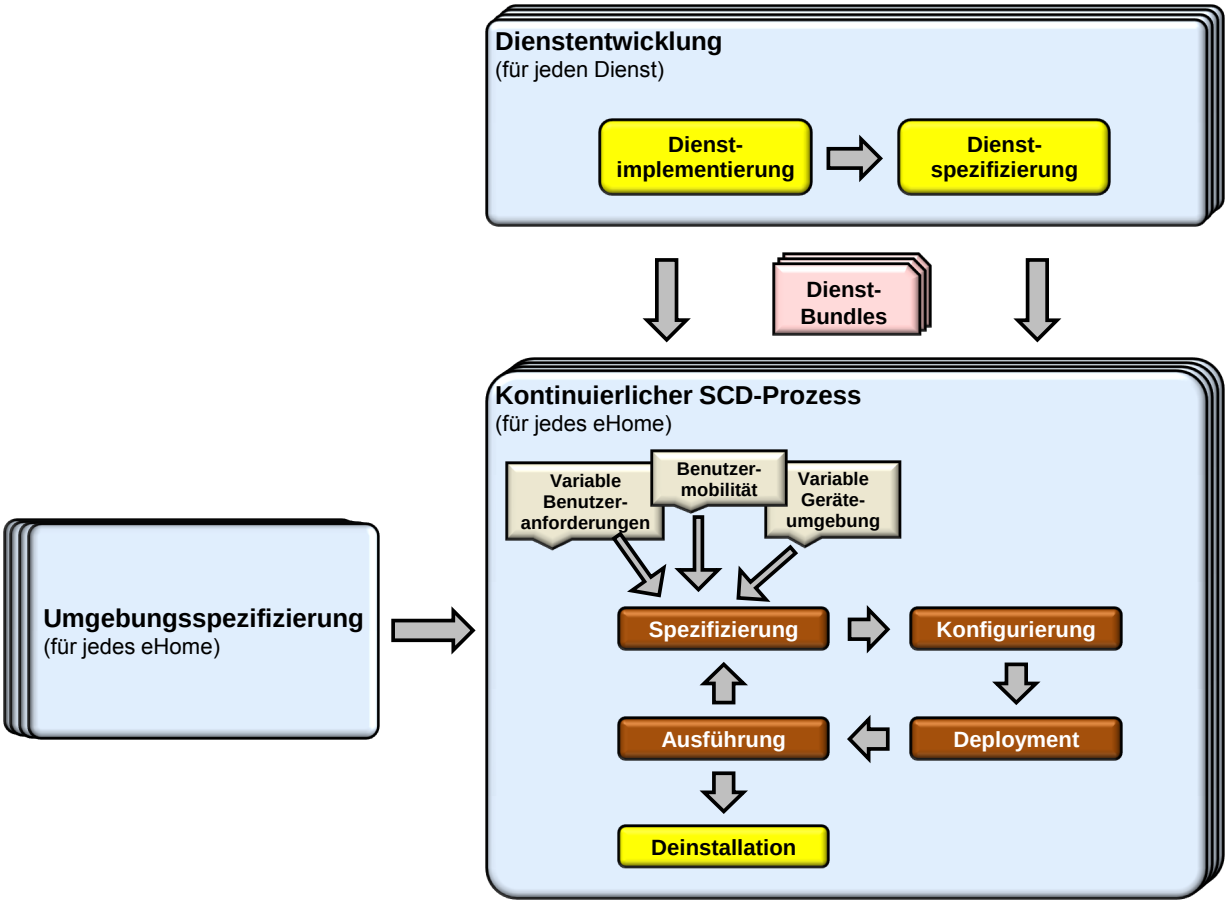


Abbildung 3.14: Kontinuierlicher SCD-Prozess (angelehnt an [Ret10]).

dadurch auszeichnet, dass jederzeit Geräte an- und abgeschafft werden oder im eHome von Raum zu Raum bewegt werden (Gerätemobilität). Außerdem ist die Benutzermobilität ein wichtiger Dynamikaspekt, der berücksichtigt werden muss. Wie in Abbildung 3.14 gezeigt, führen diese Dynamikaspekte zur Änderung der eHome-Spezifikation.

Der klassische SCD-Prozess sieht für die Abbildung solcher Änderungen auf die Konfiguration die Wartungsphase vor. Hierfür muss das System zunächst runtergefahren werden und anschließend der ganze SCD-Prozess, auch für die schon konfigurierten Dienste, erneut ausgeführt werden. Im Gegensatz dazu kann der kontinuierliche SCD-Prozess diese Änderungen dynamisch zur Laufzeit handhaben und die Konfiguration an Kontextänderungen anpassen [RS08, Ret10]. Dieser Sachverhalt ist durch die dunkelfarbige Hervorhebung der Ausführungsphase angedeutet. Dadurch wird verdeutlicht, dass der kontinuierliche SCD-Prozess auch die Ausführungsphase aktiv unterstützt und auf Kontextänderungen so reagieren kann, dass die Konfiguration, einschließlich der Dienstkomposition, an veränderte Umstände zur Laufzeit angepasst wird. Beispielsweise kann es vorkommen, dass zur Laufzeit die Bindungen des *Musikdienstes* zu den Lautsprechern in einem Raum gelöst und in einem anderen Raum neu aufgebaut werden, wenn sich eine Person von einem Raum in einen anderen bewegt.

Um die Konfiguration zur Laufzeit verändern zu können, hat Retkowitz die Dienstspezifikation um die Angabe von *Bindungsstrategien* und *Bindungsbeschränkungen* erweitert, sodass auf Basis dieser Informationen zur Laufzeit bestimmt werden kann, ob eine Umkonfigurierung von Diensten nötig ist. Eine Voraussetzung für die Anpassung der Dienstkomposition zur Laufzeit ist die Möglichkeit, einen Dienst in seiner Ausführung zu unterbrechen, um entweder bestehende Bindungen zu anderen Diensten zu löschen oder neue aufzubauen. Anschließend kann der Dienst fortgesetzt werden. Hierfür wurde die Schnittstelle EhService (s. Abbildung 3.8) um entsprechende Methoden erweitert, die von den Diensten geeignet implementiert werden müssen. Eine weitere wichtige Änderung an der Dienstimplementierung ist die Anwendung des Entwurfsmusters *Dependency Injection* [Fow04]. Nun müssen sich die Dienste ihre Unterdienste nicht selbst mithilfe der Instanz des eHome-Modells finden. Stattdessen übergibt der Deployer jedem Dienst die Referenzen seiner Unterdienste über dafür vorgesehene Methoden.

Weitere Aspekte der strukturellen Adaption betreffen die Erkennung und Lösung von *Ressourcenkonflikten* zur Laufzeit, wenn zwei oder mehr Dienste auf die gleiche Ressource zugreifen. Hierfür wurde ein *aspektorientierter* Ansatz gewählt [RK09]. Mit diesem Ansatz können Zugriffe auf Ressourcen abgefangen und überprüft werden. Falls mehrere Dienste zur gleichen Zeit auf eine Ressource zugreifen, werden *Prioritäten* der zugreifenden Dienste herangezogen, um zu entscheiden, welcher Dienst die Ressource nutzen darf.

3.3.2 Semantische Adaption

Darüber hinaus hat Retkowitz auch das Problem der Heterogenität von Diensten betrachtet. Weil eHome-Dienste von verschiedenen Herstellern stammen und nicht immer syntaktisch kompatibel sind, auch wenn sie semantisch die gleichen Funktionalitäten anbieten, wurde die *semantische Adaption* eingeführt [RP08, Ret10].

Während bei Norbisrath die Dienstkommunikation über States funktionierte, hat Retkowitz dies so geändert, dass Dienste nun über Java-Schnittstellen kommunizieren. Diese Vorgehensweise bietet mehr Flexibilität durch die Anwendung von Information Hiding. Es kann aber trotzdem sein, dass Dienste inkompatible Schnittstellen aufweisen. Daher werden für die semantische Adaption keine Schnittstellen, sondern semantische Beschreibungen von Dienstfunktionalitäten als Grundlage genommen. Hierzu wurde eine *eHome-Ontologie* eingeführt, auf deren Basis eHome-Dienste semantisch beschrieben werden können. Später kann auf Basis der semantischen Dienstspezifikationen ein semantisches *Matching* durchgeführt werden, um passende Dienste zu komponieren. Um bei Bedarf syntaktisch inkompatible aber semantisch kompatible Dienste komponieren zu können, kann weiterhin auf die *automatische Adaptergenerierung* zurückgegriffen werden. Dadurch entsteht eine größere Flexibilität, weil auf diese Weise mehr Dienste miteinander, wenn auch über Adapter, komponiert werden können, als nur über die direkte syntaktische Kompatibilität möglich wäre.

3.3.3 Werkzeugunterstützung

Für die Umsetzung der entwickelten Konzepte hat Retkowitz entsprechende Werkzeuge realisiert, die im Kern die Funktionalitäten des eHomeConfigurator erweitern. Teilweise wurden diese Funktionalitäten voneinander getrennt in unterschiedliche Werkzeuge ausgelagert. Für die Ontologie-Entwicklung wurde kein eigenes Werkzeug entwickelt, weil dafür das bekannte *Protégé* eingesetzt werden kann. Der *Service-Editor* dient der Spezifikation von Diensten. Der *Umgebungseditor* wird für die Spezifizierung des statischen Umgebungskontexts eingesetzt. Die wichtigsten Funktionalitäten zur strukturellen und semantischen Adaption werden vom *Laufzeit-Manager* umgesetzt, der den eHomeConfigurator abgelöst hat. Weiterführende Informationen sind in [Ret10] enthalten. Für den weiteren Verlauf der Arbeit sei jedoch darauf hingewiesen, dass der Begriff eHomeConfigurator der Einfachheit halber sowohl für den ursprünglichen eHomeConfigurator als auch für den Laufzeit-Manager verwendet wird. Dabei sollte beachtet werden, dass ab jetzt die Verwendung dieses Begriffes auch die erweiterten Funktionalitäten einbezieht, die Retkowitz entwickelt hat.

3.4 Bewertung

Die Vorgängerarbeiten im eHome-Projekt verfolgten das Ziel der Entwicklung und Konfigurierung wiederverwendbarer Softwarekomponenten zur Realisierung von eHome-Diensten. Während Kirchhof mit dem eHome-Prozess einen Gesamtprozess eingeführt hat, der sowohl den Dienstlebenszyklus als auch die Kunde-Anbieter-Beziehungen formalisiert, haben Norbisrath und Retkowitz konkrete Ansätze für den SCD-Prozess entwickelt und in Form von Werkzeugen umgesetzt. Diese Ansätze tragen zur Überwindung der Heterogenität und Dynamik in eHomes bei. Durch die erreichte Wiederverwendbarkeit von eHome-Diensten können eHomes kostengünstig eingerichtet werden, wodurch der Weg für einen zukünftigen Massenmarkt geebnet wird.

Es gibt jedoch Aspekte, die durch obige Ansätze nicht abgedeckt werden. Hierzu gehört insbesondere die Unterstützung der *Inter-eHome-Mobilität* und weiterer, daraus resultierender Fragestellungen. Die Inter-eHome-Mobilität bezeichnet die Tatsache, dass sich Benutzer im Alltag nicht nur in einem eHome aufhalten, sondern auch weitere Umgebungen besuchen, wie etwa den Arbeitsplatz, Hotels oder andere öffentliche Orte. In dieser Arbeit wird der eHome-Begriff so verwendet, dass auch diese Umgebungen von diesem Begriff umfasst werden. Im Folgenden werden die Ergebnisse der Vorgängerarbeiten bezüglich dieser Aspekte analysiert und bewertet.

3.4.1 Dienstauswahl

Die erste Fragestellung betrifft die Möglichkeit, dass mobile Benutzer auch in fremden eHomes, in denen sie sich vorübergehend aufhalten, Dienste nutzen können. Dazu muss der Benutzer in irgendeiner Form dem eHome mitteilen, ob und welche Dienste er nutzen möchte. Umgekehrt muss das eHome ihm auch mitteilen können, welche Dienste es überhaupt anbietet.

Die bisherigen Ansätze der eHome-Gruppe sehen den Einsatz des eHomeConfigurator für die Auswahl von eHome-Diensten zur Installation vor. Erst nach ihrer Installation mit dem eHomeConfigurator sind sie von Benutzern verwendbar. Bislang kann die Liste verfügbarer Dienste nur über dieses Werkzeug eingesehen werden. Aber auch die Zuordnung von Diensten zu Benutzern kann nur mit diesem Werkzeug durchgeführt werden. Dieses Werkzeug ist jedoch für die Bedienung von einem Administrator vorgesehen und nicht von dem normalen Benutzer, der nicht mit administrativen Aufgaben belastet werden möchte. Daher ist diese Vorgehensweise nicht für die Inter-eHome-Mobilität geeignet. Im Rahmen dieser Arbeit wird mobilen Benutzern die Möglichkeit geboten, in besuchten Umgebungen komfortabel *auswählen* zu können, welche Dienste sie dort verwenden möchten. Falls gewünschte Dienste in dem besuchten eHome nicht vorhanden sind, haben sie die Möglichkeit, mit wenig Aufwand eigens mitgebrachte Dienste zu verwenden.

3.4.2 Dienstinteraktion

Ein weiterer Aspekt, der auch unabhängig von der Inter-eHome-Mobilität betrachtet werden kann, ist die *Interaktion* mit Diensten. Ergebnisse des eHome-Projekts ermöglichen momentan drei Arten, wie Benutzer mit Diensten in eHomes interagieren können. Erstens können sie vorhandene Geräte wie Schalter oder Regler benutzen, deren Treiber die Benutzeraktionen an die entsprechenden Top-Level-Dienste weiter leiten. Zweitens können sie indirekt mit den Diensten interagieren, indem sie Kontextänderungen verursachen, die von Sensoren erfasst und an die Dienste weitergeleitet werden. Ein Beispiel einer solchen Kontextänderung ist der Raumwechsel, der eine Anpassung der Dienstfunktionalität nach sich ziehen kann. Drittens können sie die Parameter der Dienste im eHome-Modell bearbeiten. Dies kann mit dem eHomeConfigurator durch direkte Bearbeitung der Konfiguration oder durch Verwendung des Wizards geschehen (s. Abschnitt 3.2.3).

Ähnlich wie bei der Dienstauswahl ist auch die Interaktion mit Diensten über den eHomeConfigurator für mobile Benutzer nicht angemessen. Im Rahmen dieser Arbeit wird die Möglichkeit geschaffen, mit Diensten bequem über geeignete *Benutzeroberflächen* zu interagieren, die auf die spezifischen Funktionalitäten der Dienste zugeschnitten sind.

3.4.3 Personalisierung

Eine der wichtigsten Eigenschaften und zugleich auch Herausforderungen von eHomes ist die *Personalisierung* (s. Abschnitt 2.2.4). Bisherige Ansätze im eHome-Projekt bieten nur eine rudimentäre Unterstützung der Personalisierung von eHomes, wie im Folgenden erläutert wird.

Die Personalisierung von Diensten kann nach Norbisrath auf zwei Arten vorgenommen werden. Die erste und in [Nor07] umgesetzte Variante ist die Abbildung von Benutzerdaten auf Instanzen der Klasse Attribute im eHome-Modell, die mit den Klassen Service oder ServiceObject verbunden sind (s. Abbildung 3.7 und Abbildung 3.8). Diese Attribute müssen in der Dienstspezifikation enthalten sein und können mit dem eHomeConfigurator im Rahmen der Parametrisierung bearbeitet werden. Abbildung 3.10 zeigt die Attribute `Person`

`List` und `Sound URL List` des `Musikdienstes`, die angeben, welche Personen im eHome existieren und welchen Musikgeschmack sie haben. Der Dienst kann diese Attribute zur Laufzeit auslesen und seine Funktionalität entsprechend anpassen. Diese Vorgehensweise hat jedoch mehrere Nachteile.

Zunächst wird vorausgesetzt, dass im Voraus bekannt ist, welche Benutzer sich in einem eHome aufhalten und personalisierbare Dienste nutzen werden. Dabei sind Benutzer hochgradig mobil und können jederzeit ein eHome auch nach der Inbetriebnahme betreten oder verlassen, wie etwa in einem Hotel. Diese Form der Benutzerdynamik als Resultat der Inter-eHome-Mobilität wird in den bisherigen Ergebnissen der eHome-Gruppe jedoch nicht berücksichtigt. Der im Rahmen der vorliegenden Arbeit verfolgte Ansatz unterstützt hingegen die Handhabung der Benutzerfluktuation zur Laufzeit.

Ferner unterscheidet der bisherige Ansatz nicht, ob sich ein Benutzer aus der Personenliste aktuell im eHome befindet oder nicht. Jeder Dienst muss für sich selbst ermitteln, welche Benutzer in einen Raum aktuell präsent sind. Hierfür ist vorgesehen, dass jedem Raum ein Basisdienst zur Personenerkennung zugeordnet wird, der ein State-Objekt `lastPersonDetected` besitzt. In diesem Objekt wird der Name der Person eingetragen, die den Raum zuletzt betreten hat. Jeder Dienst, der wissen möchte, ob und welche Personen sich in einem Raum befinden, muss in seiner Spezifikation `person` als benötigte Funktion enthalten (s. Abbildung 3.10), damit er mit dem `Personendetektor` verbunden wird. Zur Laufzeit kann der Dienst auf `lastPersonDetected` zugreifen und intern eine Liste der sich in einem Raum befindenden Personen verwalten. Diese Funktionalität muss für jeden Dienst separat implementiert werden. Daraus entsteht eine *Redundanz*, die den Entwicklungsaufwand für Dienste erhöht. Daher wird im Rahmen dieser Arbeit die Benutzerverwaltung aus den Diensten ausgelagert und ihnen über die Laufzeitumgebung einheitlich zur Verfügung gestellt.

Zudem tritt hier das gleiche Problem bei der Dienstauswahl und -interaktion auf. Denn auch für die Pflege von Benutzerdaten muss der eHomeConfigurator verwendet werden, der aber für administrative Aufgaben vorgesehen ist und nicht für die Verwendung durch den Endbenutzer (s. auch Abschnitt 3.4.1 und Abschnitt 3.4.2). Vorzugsweise sollten Endbenutzer auf eine *einfachere* Art und Weise Dienste personalisieren können.

Diese Probleme werden durch die Inter-eHome-Mobilität zusätzlich verstärkt, weil für jeden Benutzer und jedes eHome der Personalisierungsprozess erneut ausgeführt werden muss. Dies führt dazu, dass Benutzerdaten in jedem besuchten eHome separat gespeichert werden und daher verteilt und redundant vorliegen (*Redundanzproblem*). Somit müssen Änderungen oder Ergänzungen von Benutzerdaten in jedem betroffenen eHome manuell nachgezogen werden (Synchronisierung). Dadurch wird der Personalisierungsaufwand multipliziert. Eine Möglichkeit zur zentralen Speicherung und Verwaltung von Benutzerdaten sowie zur automatischen Synchronisierung der Daten zwischen mehreren eHomes fehlt bisher.

Das *Redundanzproblem* besteht nicht nur auf eHome-Ebene, sondern tritt auch schon dann auf, wenn ein eHome für sich allein betrachtet wird. Dies ist dadurch bedingt, dass für jeden Dienst eigene Attribute für Benutzerdaten vorgesehen sind. Sind nun zwei oder mehr Dienste in einem eHome vorhanden, die dieselben Präferenzen eines Benutzers benötigen,

müssen diese Präferenzen für jeden Dienst in separaten Attributen abgelegt werden. Beispielsweise kann es vorkommen, dass sowohl der `Temperaturdienst` als auch der Weckdienst die Temperaturpräferenz desselben Benutzers benötigen. Offensichtlich führt die Redundanz der Dienstattribute auch innerhalb eines eHomes zu *erhöhtem Personalisierungsaufwand*, der vermieden werden sollte.

Um diesen Aufwand zu reduzieren, hat Norbisrath das Datenmodell um den Personenkontext erweitert (s. Abschnitt 3.2.2.6). Im Personenkontext wird jeder Benutzer durch eine Instanz der Klasse Person repräsentiert, an die alle Benutzerdaten in Form von Attributen angehängt werden können. Personalisierbare Dienste können dann über diese Attribute iterieren, um die erforderlichen Informationen zu ermitteln. Die Redundanz von Dienstattributen in einem eHome könnte vermieden werden, wenn der Personenkontext konsequent verwendet würde. Der Personenkontext kann jedoch nicht die Redundanz auf eHome-Ebene vermeiden, die aufgrund redundanter Speicherung und Verwaltung von Benutzerdaten in mehreren eHomes entsteht.

Den Personenkontext hat Norbisrath nur als Konzept vorgeschlagen, die Umsetzung ist jedoch ausgeblieben. Insbesondere bietet der eHomeConfigurator keine Möglichkeit zum Erstellen und Bearbeiten von Person-Objekten. Daher wurde z. B. der `Musikdienst` in Abbildung 3.10 weiterhin über Dienstattribute personalisiert. Retkowitz hat dieses Konzept in seiner Arbeit aufgegriffen und ermöglicht das Anlegen und Bearbeiten von Person- und Attribute-Instanzen direkt auf der Konfiguration. Aber auch hier wurde der eHomeConfigurator zur Bearbeitung des Personenkontexts ausgewählt. Aus den schon oben genannten Gründen ist dies für die Inter-eHome-Mobilität jedoch nicht praktikabel.

Ein weiteres Problem, das unter anderem auch die Personalisierung betrifft, ist eine *fehlende Abstraktionsschicht* auf Modellebene. Sowohl die Struktur von Dienstattributen als auch die des Personenkontexts ist nicht gekapselt. Dadurch werden Benutzerdaten über das ganze eHome-Modell zerstreut und mit anderen Daten vermischt. Zudem erlaubt der eHomeConfigurator keine getrennte Speicherung von Benutzerdaten. Wie auch von Kobsa in [KW89] gefordert wird, sollten Benutzerdaten jedoch vom Rest des Systems getrennt werden. Diese Forderung wird durch das eHome-Modell nicht erfüllt. Im Rahmen dieser Arbeit werden die Benutzerdaten daher vom eHome-Modell entkoppelt und gekapselt. Dadurch wird auch die Mitnahme von Benutzerdaten in andere eHomes vereinfacht.

Die fehlende Abstraktionsschicht führt aber auch dazu, dass Dienstentwickler unnötige Details des Personenkontexts kennen. Durch die Verwendung dieser resultieren wiederum Dienste, die zu stark von internen Details des eHome-Modells abhängen. Diese Abhängigkeit beeinträchtigt unnötigerweise die *Wartbarkeit* und *Flexibilität* von eHome-Diensten, weil strukturelle Änderungen des eHome-Modells zwangsläufig die Anpassung von Diensten nach sich ziehen. Durch Anwendung von *Datenabstraktion* [Nag90] sollten eHome-Dienste daher vom Datenmodell entkoppelt werden. In [Ret10] hat Retkowitz Dienste vom eHome-Modell insofern entkoppelt, dass sie ihre Unterdienste nicht mehr selbst über das Modell ermitteln, sondern sie durch Dependency Injection mitgeteilt bekommen. Durch die Entkopplung der Benutzerdaten im Rahmen dieser Arbeit können die Dienste zudem nur noch über definierte Schnittstellen auf Benutzerdaten zugreifen, wodurch die Wartbarkeit und Flexibilität von eHomes verbessert wird.

3.4.4 Schutz der Privatsphäre

Eine wichtige Voraussetzung für die Akzeptanz von eHomes ist ihre Vertrauenswürdigkeit. Sie werden sich nur dann nachhaltig durchsetzen, wenn die Benutzer darauf vertrauen können, dass ihre *Privatsphäre* durch die neue Technik nicht stärker gefährdet wird, als es ohne diese Technik möglich war. Gleichzeitig eröffnen die Miniaturisierung und Verbreitung von Mikroprozessoren neue Möglichkeiten zur Aufzeichnung und Verarbeitung digitaler Daten. Speziell die Verwendung personalisierbarer eHome-Dienste führt dazu, dass Benutzerdaten im Rahmen der Inter-eHome-Mobilität an verschiedenen Orten offengelegt werden. Daher besteht eine besondere Herausforderung darin, den Schutz der Privatsphäre mit der Personalisierung von eHomes miteinander in Einklang zu bringen.

Im eHome-Projekt wurde das Problemfeld der Privatsphäre bisher nicht betrachtet. Daher existieren auch keine Konzepte, die Benutzer beim Schutz ihrer Privatsphäre unterstützen. Dazu zählt auch der Schutz von Benutzerdaten (*Datenschutz*). Beispielsweise wird bei näherer Betrachtung des eHome-Modells deutlich, dass keinerlei *Zugriffskontrolle* für Benutzerdaten realisiert wurde. Diese liegen in Form von Dienst- oder Personenattributen vor und können von jedem Dienst bzw. Werkzeug gelesen und überschrieben werden. Daher können bösartige Dienste Benutzerdaten aller Personen eines eHomes sammeln und *zweckentfremdet* verarbeiten. Im schlimmsten Fall könnten im Rahmen der Inter-eHome-Mobilität Daten eines Benutzers aus verschiedenen eHomes kombiniert werden (*Verkettung*), um beispielsweise ein Bewegungsprofil zu erstellen oder Informationen über soziale Kontakte zu ermitteln (s. auch Abschnitt 1.2.2). Es besteht daher der Bedarf an Konzepten, die mobile Benutzer beim Schutz ihrer Privatsphäre unterstützen (s. auch Abschnitt 1.3.4). Die Entwicklung solcher Konzepte ist ein zentrales Thema dieser Arbeit.

3.4.5 Schutz von eHomes

Eng mit dem Schutz der Privatsphäre ist der Schutz von eHome-Diensten (und damit auch von eHomes) vor dem Zugriff unbefugter *Dienste* und *Benutzer* verbunden. Einerseits sollten Dienste nur auf diejenigen Unterdienste zugreifen können, die sie zur Erfüllung ihrer Funktionalität benötigen. Beispielsweise sollte ein *Musikdienst* nicht auf einen *Türdienst* zugreifen können, um die Haustür zu öffnen, wenn niemand zu Hause ist. Andererseits sollten Benutzer nur diejenigen Dienste verwenden können, für die sie autorisiert wurden. Beispielsweise sollten Kinder oder Gäste nicht den *Sicherheitsdienst* ausschalten können oder Hotelgäste nicht die gebührenpflichtige Sauna benutzen, ohne dafür bezahlt zu haben.

Auch diese Problemstellung wurde im eHome-Projekt noch nicht betrachtet. Daher existieren keinerlei Konzepte für eine *Zugriffskontrolle* zum Schutz von eHome-Diensten. Im Rahmen dieser Arbeit werden entsprechende Konzepte entwickelt.

3.5 Zusammenfassung

In diesem Kapitel wurden Vorarbeiten der eHome-Gruppe vorgestellt und bezüglich der Zielsetzung der vorliegenden Arbeit bewertet. Diese Arbeiten beschäftigen sich hauptsächlich mit

der Anwendung softwaretechnischer Konzepte auf eHome-Systeme, um Herausforderungen wie Heterogenität und Dynamik in eHomes zu bewältigen. Das Ergebnis ist der Übergang vom klassischen Entwicklungsprozess für eHomes zu einem konfigurierungsbasierten Prozess. Der neue Prozess entkoppelt die Dienstentwicklung von der spezifischen Systementwicklung für einzelne eHomes. Letzteres reduziert sich nun lediglich auf die Konfigurierung von Diensten in Form von Standardkomponenten [Nor07]. Erweiterungen des Prozesses ermöglichen die dynamische Adaption der Konfiguration zur Laufzeit an veränderte Benutzeranforderungen oder die Mobilität von Benutzern und Geräten in eHomes [Ret10]. Unterstützt wird der Prozess von verschiedenen Werkzeugen, die auf dem eHome-Modell aufsetzen.

Durch die Analyse der Vorgängerarbeiten bezüglich ihrer Eignung zur Unterstützung der Inter-eHome-Mobilität sowie der Wahrung der Privatsphäre und den Schutz von eHomes wurden verschiedene Aspekte identifiziert, die nicht angemessen oder überhaupt nicht behandelt wurden und somit Potenzial für weitere Forschung bieten. Daraus wurden die Ziele für diese Arbeit abgeleitet. Zum einen sollen Konzepte zu Unterstützung mobiler Benutzer für die Auswahl von, die Interaktion mit und die Personalisierung von eHome-Diensten entwickelt werden. Für die Personalisierung wurde festgestellt, dass sowohl die Benutzerdaten aus dem eHome-Modell als auch die personalisierbaren Dienste von den Benutzerdaten entkoppelt werden müssen. Als ein weiteres Ziel wurde die Entwicklung geeigneter Konzepte zur Wahrung der Privatsphäre mobiler Benutzer sowie zum Schutz von eHomes vor Zugriff durch unbefugte Benutzer oder Dienste identifiziert.

In den folgenden Kapiteln werden die im Rahmen dieser Arbeit entwickelten Konzepte vorgestellt. Dabei wird an geeigneten Stellen auf die Ergebnisse der Vorarbeiten zurückgegriffen. An diesen Stellen wird hierzu jeweils ein Hinweis gegeben.

Kapitel 4

Benutzermodell

Im vorausgegangenen Kapitel wurde dargelegt, dass die Architektur des eHome-Modells einige Schwächen in Bezug auf die Verwaltung von Benutzerdaten aufweist. In diesem Kapitel wird nun ein neuer Baustein eingeführt, der für die Verwaltung von Benutzerdaten zuständig ist. Diese Komponente wird als *Benutzermodell* bezeichnet und behebt folgende Probleme:

- Eine der Probleme besteht darin, dass die Benutzerdaten im eHome-Modell mit anderen Daten *vermischt* sind und nicht getrennt gespeichert und verarbeitet werden können. Nun werden die Benutzerdaten aus dem eHome-Modell extrahiert und ausschließlich im Benutzermodell verwaltet. Dadurch wird die in Abschnitt 2.1.1 geforderte *Entkopplung* von Benutzerdaten vom Rest des Systems erzielt (s. [KW89]).

- Ein weiteres Problem besteht in der *fehlenden Kapselung* von Benutzerdaten, weswegen eine zu starke Kopplung zwischen Produzenten und Konsumenten von Benutzerdaten und dem eHome-Modell entsteht. Das Benutzermodell hingegen kapselt seine internen Datenstrukturen durch Anwendung von Information Hiding. Dadurch wird eine lose Kopplung des Benutzermodells mit den übrigen Komponenten im eHome erzielt.

- Im eHome-Modell können ferner *Redundanzen* von Benutzerdaten entstehen, weil konkrete Vorgaben zu ihrer genauen Repräsentation fehlen. Im neuen Benutzermodell kann dieses Problem hingegen nicht mehr auftreten.

- Schließlich macht das eHome-Modell keine konkreten Vorgaben über die Terminologie und die Struktur von Benutzerdaten. Daraus können Inkonsistenzen zwischen den Diensten unterschiedlicher Hersteller resultieren. Das neue Benutzermodell bezieht sich auf eine aus der Literatur bekannte Ontologie. Dadurch wird sichergestellt, dass alle eHomes die gleiche Terminologie und Semantik für Benutzerdaten verwenden. Aus diesem Grund können mobile Benutzer ihre Daten auch in mehreren eHomes wiederverwenden.

Um diese Probleme zu beheben, reicht lediglich eine *Umgestaltung* (engl. *Refactoring*) der Architektur des existierenden eHome-Prototyps aus. Als Ergebnis dieser Umgestaltung werden Benutzerdaten vom eHome-Modell in das neu entstandene Benutzermodell

verschoben. Doch das Benutzermodell soll im Rahmen dieser Arbeit nicht nur der reinen Modellierung von Benutzern, also der passiven Datenhaltung, dienen. Sie wird auch aktiv eigene Funktionalitäten realisieren, die eine wichtige Rolle bei der Erreichung der Ziele dieser Arbeit spielen und bisher vom eHome-Prototyp nicht realisiert wurden:

- Im Gegensatz zum eHome-Modell bietet das Benutzermodell eine aktive Benutzerverwaltung an. Es hat stets eine Übersicht über alle in einem eHome bzw. Raum anwesenden Benutzer. Diese Informationen werden den eHome-Diensten (und weiteren Komponenten) im eHomezur Verfügung gestellt, damit sich diese nicht mehr selbst um die Benutzerverwaltung kümmern müssen.

- Das Benutzermodell kann ferner die dynamische Benutzerfluktuation handhaben. Es kann für Benutzer, die das eHome im Rahmen der Inter-eHome-Mobilität betreten oder verlassen, neue Sitzungen aufbauen oder bestehende löschen.

- Das Benutzermodell realisiert auch eine Zugriffskontrolle, indem es sicherstellt, dass nur berechtigte Dienste auf Benutzerdaten zugreifen können.

Damit ist das Benutzermodell ein *aktiver* Verwaltungsbaustein. Dieser aktive Charakter könnte durch Begriffe wie *Benutzermodell-System* oder *Benutzermodell-Manager* betont werden. Der Einfachheit halber wird im weiteren Verlauf dieser Arbeit jedoch ausschließlich der Begriff *Benutzermodell* ohne Zusätze verwendet. Dieser Begriff umfasst daher sowohl den passiven Teil zur Modellierung von Benutzern als auch den aktiven Teil mit den oben genannten Funktionalitäten.

Im Folgenden wird zunächst erläutert, wie Benutzerdaten aus dem eHome-Modell entkoppelt werden, mit welchen anderen Bausteinen das Benutzermodell in einem eHome interagiert, welche Rolle es bei der Handhabung der Intra-eHome-Mobilität von Benutzern spielt und wie die Architektur des Benutzermodells aussieht. Dabei wird auch speziell darauf eingegangen, wie sich die verwendete Ontologie auf die Architektur des Benutzermodells ausgewirkt hat. Die Handhabung der Benutzerfluktuation und die Realisierung der Zugriffskontrolle durch das Benutzermodell werden in Kapitel 6 behandelt.

4.1 Entkopplung von Benutzerdaten

Durch die Verflechtung mit anderen Daten im eHome-Modell können Benutzerdaten in eHomes nur mit großem Aufwand gepflegt und nicht unabhängig vom Rest des eHome-Modells gespeichert und verarbeitet werden. Ferner erschwert dies die Dienstentwicklung und die Wartbarkeit von eHomes.

Abbildung 4.1 zeigt die Situation vor und nach der Einführung des neuen Benutzermodells, der die aufgeführten Probleme behebt. Auf der linken Seite der Abbildung wird die Situation vor der Einführung des Benutzermodells gezeigt. Hier existiert in *jedem* eHome eine Instanz des eHome-Modells, in der alle Informationen enthalten sind, die für den SCD-Prozess und den Betrieb von eHomes benötigt werden. Das eHome-Modell wird sowohl vom eHomeConfigurator als auch von den eHome-Diensten verwendet. Der eHomeConfigurator

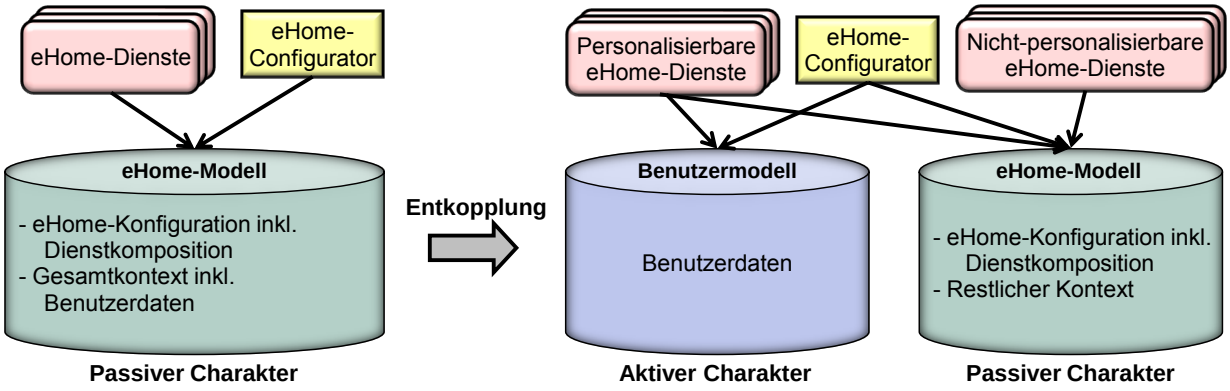


Abbildung 4.1: Entkopplung von Benutzerdaten aus dem eHome-Modell.

ermittelt aus dem eHome-Modell die Informationen, die für die Durchführung des SCD-Prozesses nötig sind. Beispielsweise gehören hierzu Umgebungsinformationen und die Liste der im eHome ausgeführten Dienste. Die eHome-Dienste hingegen verwenden das eHome-Modell, um einerseits über die Dienstkomposition ihre Unterdienste zu ermitteln und andererseits Kontextinformationen zu erlangen, zu denen auch Benutzerdaten zählen. Auch die Interaktion mit anderen Diensten findet über das eHome-Modell statt, eine direkte Dienstinteraktion über gegenseitige Methodenaufrufe erfolgt nicht (s. Abschnitt 3.2.2.5). In der Abbildung ist außerdem angedeutet, dass das eHome-Modell einen *passiven* Charakter hat. Das bedeutet, dass es ausschließlich der Datenhaltung dient. Die Daten werden von anderen Komponenten erstellt oder weiterverarbeitet. Es erfüllt selbst keine aktive Funktionalität. Für eine genauere Diskussion der unterschiedlichen Bedeutungen von Aktiv und Passiv sei hier auf [Nag90] verwiesen.

Auf der rechten Seite von Abbildung 4.1 ist die Situation nach der Einführung des Benutzermodells dargestellt. Dabei ersetzt das Benutzermodell das eHome-Modell nicht, sondern ist nun für die Verwaltung der vom eHome-Modell entkoppelten Benutzerdaten zuständig. Der Ansatz sieht zunächst vor, dass in jedem eHome jeweils genau eine Instanz des Benutzermodells und eine Instanz des eHome-Modells vorhanden sind. Das Benutzermodell ist somit für die Verwaltung der Daten *aller* Benutzer in einem eHome verantwortlich (s. Abbildung 4.2). Im nächsten Kapitel wird eine mobile Version des Benutzermodells für Handhelds vorgestellt, deren Instanzen ausschließlich die Daten *eines* Benutzers verwalten. Wie oben erwähnt, unterscheidet sich das Benutzermodell von dem eHome-Modell insbesondere dadurch, dass es einen *aktiven* Charakter hat, weil es neben der Datenhaltung auch aktive Funktionalitäten realisiert.

Das Problem der Redundanz taucht nun nicht mehr auf, da das Benutzermodell allein für die Verwaltung der Benutzerdaten zuständig ist. sorgt dafür, dass eine Information nicht durch mehrere Attribute redundant repräsentiert wird.

Durch die Einführung des Benutzermodells wird nun zwischen personalisierbaren und nicht-personalisierbaren eHome-Diensten unterschieden. Während personalisierbare Dienste jetzt zusätzlich mit dem Benutzermodell interagieren, um Benutzerdaten zu erlangen, wird

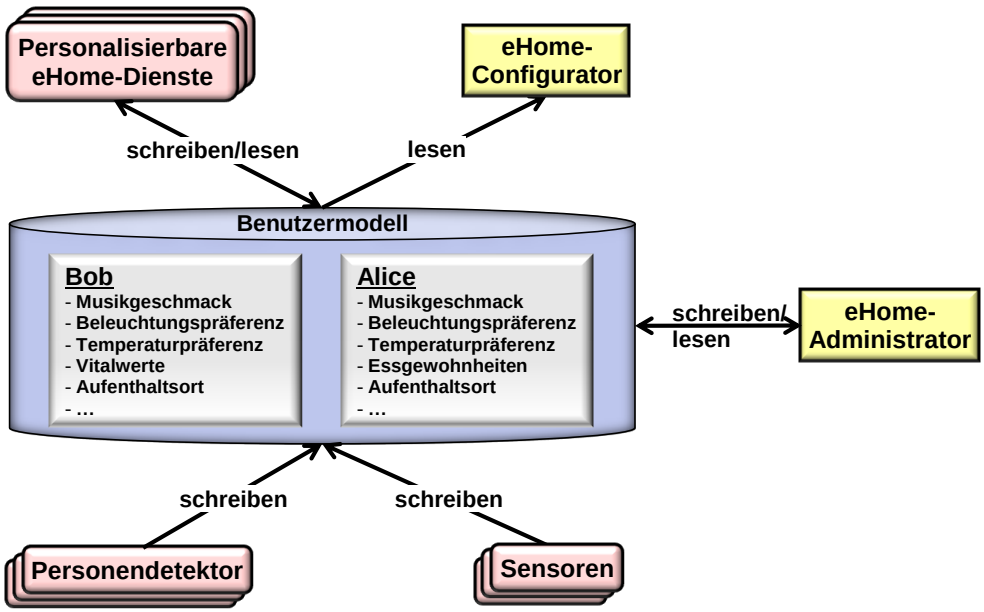


Abbildung 4.2: Produzenten und Konsumenten von Benutzerdaten im eHome.

das eHome-Modell weiterhin von beiden Diensttypen verwendet. Dies liegt insbesondere daran, dass das eHome-Modell noch Kontextinformationen enthält, die nicht-personenbezogen sind und von beiden Diensttypen benötigt werden können.

Zusätzlich ist in Abbildung 4.1 dargestellt, dass auch der eHomeConfigurator mit dem Benutzermodell interagiert. Dadurch kann er auf die Informationen über die Aufenthaltsorte von Benutzern zugreifen, die zuvor noch im eHome-Modell abgelegt wurden. Der eHome-Configurator muss nämlich personalisierbare Dienste zur Laufzeit umkonfigurieren, wenn ein Benutzer seinen Aufenthaltsraum ändert. Dieser Aspekt wird in Abschnitt 4.3 näher diskutiert. Der Zugriff des eHomeConfigurator auf das eHome-Modell bleibt bestehen, weil das eHome-Modell weiterhin die Grundlage für den SCD-Prozess bildet.

In Abschnitt 3.4.3 wurde erwähnt, dass Retkowitz in seiner Arbeit eHome-Dienste vom eHome-Modell so entkoppelt hat, dass sie ihre Unterdienste nicht mehr selbst über das Modell ermitteln müssen. Stattdessen teilt ihnen der eHomeConfigurator die Referenzen ihrer Unterdienste per *Dependency Injection* mit. Auch wird das eHome-Modell inzwischen nicht mehr als Kommunikationsmedium für Dienste verwendet. In Zusammenarbeit mit Retkowitz wurde die Konvention eingeführt, dass eHome-Dienste durch direkte Methodenaufrufe auf Instanzebene kommunizieren. Somit ist der einzige verbleibende Grund für die Verwendung des eHome-Modells durch eHome-Dienste die Erlangung nicht-personenbezogener Kontextinformationen, wie etwa die aktuelle Raumtemperatur oder die Zustände von Elektrogeräten.

4.2 Produzenten und Konsumenten

Abbildung 4.2 zeigt beispielhaft ein Benutzermodell, das die Benutzerdaten von zwei Benutzern, namentlich Alice und Bob, verwaltet. Der Begriff *Benutzerdaten* ist dabei weit gefasst und kann sehr unterschiedliche Informationen umfassen, wie in Abschnitt 2.1.1.2 allgemein erläutert wurde. In eHomes verwaltet das Benutzermodell speziell solche Daten, die für die Personalisierung von eHome-Diensten benötigt werden. Beispielsweise sind Bobs `Musikgeschmack`, seine `Licht-` und `Temperaturpräferenzen` sowie seine `Vitalwerte` im Benutzermodell festgehalten. Die `Vitalwerte` von Alice sind dagegen nicht abgelegt, dafür aber ihre `Essgewohnheiten`. Zusätzlich sind für beide Benutzer ihre aktuellen `Aufenthaltsorte` gespeichert. Sie entsprechen jeweils dem Raum, in dem sich die Benutzer befinden. Diese Liste ist keinesfalls vollständig und dient nur der Illustration, dass das Benutzermodell für jeden Benutzer unterschiedliche Daten enthalten kann.

Weiterhin zeigt Abbildung 4.2 die möglichen Interaktionspartner des Benutzermodells. Sie werden unterteilt in Produzenten und Konsumenten von Benutzerdaten. *Konsumenten* greifen *lesend* auf das Benutzermodell zu, um die für ihre Funktionalitäten relevanten Daten zu erlangen. Wie in Abbildung 4.2 dargestellt, gibt es zwei Arten von Konsumenten. Erstens sind dies *personalisierbare Dienste*, weil jeder personalisierbare Dienst benutzeradaptiv ist und daher seine Funktionalitäten an Benutzerdaten anpassen kann. Beispielsweise passt der `Musikdienst` die abgespielte Musik an den `Musikgeschmack` seiner Benutzer an. Dagegen überwacht der `Medizindienst` die `Vitalwerte` seiner Benutzer und ergreift in Gefahrensituationen geeignete Maßnahmen, wie etwa die Benachrichtigung medizinischen Personals. Weitere Beispiele personalisierbarer Dienste sind in Abschnitt 2.2.3 gegeben. Neben den personalisierbaren Diensten zählt auch der eHomeConfigurator zu den Konsumenten von Benutzerdaten. Wie im nächsten Abschnitt genauer beschrieben wird, benötigt er speziell Informationen über den Aufenthaltsort der Benutzer im eHome, um bei Bedarf personalisierbare Dienste zur Laufzeit umkonfigurieren zu können.

Produzenten hingegen greifen *schreibend* auf das Benutzermodell zu, um es mit Benutzerdaten zu beliefern. Zunächst zählen *personalisierbare Dienste* zu den Produzenten, weil sie Benutzerdaten nicht nur konsumieren, sondern auch generieren können. Beispielsweise könnte ein intelligenter `Heizungsdienst` unbemerkt die `Raumtemperatur` ein Grad niedriger einstellen, als vom Benutzer angegeben. Falls der Benutzer mit diesem Wert zufrieden ist, kann der Dienst seine `Temperaturpräferenz` im Benutzermodell überschreiben und zum Energiesparen beitragen. Ferner gehört der *eHomeAdministrator* zu den Produzenten von Benutzerdaten. Dies ist ein Werkzeug, das Benutzern die Bearbeitung ihrer Daten über eine grafische Benutzeroberfläche ermöglicht. Dieses Werkzeug wurde im Rahmen dieser Arbeit entwickelt und wird in Abschnitt 8.1 beschrieben. Schließlich können Basisdienste in Form von *Sensor*treibern Benutzerdaten generieren und in das Benutzermodell schreiben. Beispiele für solche Sensoren sind der `Pulsmesser`, der `Blutdruckmesser` oder `Beschleunigungssensoren` zur Messung der Schrittgeschwindigkeit von Personen.

Ein besonderer und in Abbildung 4.2 hervorgehobener Basisdienst, der Benutzerdaten erzeugt, ist der `Personendetektor`. In dieser Arbeit wird angenommen, dass in jedem Raum ein `Personendetektor` vorhanden ist, der erkennen kann, welche Benutzer einen

Raum betreten. Immer wenn in einem Raum ein Benutzer erkannt wird, benachrichtigt der `Personendetektor` das Benutzermodell, das wiederum den Aufenthaltsort des betroffenen Benutzers aktualisiert. Es gibt unterschiedliche Möglichkeiten, Personen in einem Gebäude zu erkennen und zu lokalisieren [HVJS07]. An dieser Stelle wird jedoch von den Details der Erkennungstechnik abstrahiert. Es wird angenommen, dass in zukünftigen eHomes eine ausgereifte Technik zur Verfügung steht und diese an das Benutzermodell angebunden werden kann. In Abschnitt 8.3 werden einige Demonstratoren beschrieben, die im Rahmen dieser Arbeit zu Testzwecken eingesetzt wurden und mit solchen Techniken ausgestattet sind.

Die nähere Analyse des `Personendetektor`-Beispiels veranschaulicht einen wichtigen Unterschied zu den Vorgängerarbeiten im eHome-Projekt (s. Abschnitt 3.4.3). Bislang mussten alle personalisierbaren Dienste in den Räumen, in denen ihre Funktionalitäten verfügbar sein sollten, mit den entsprechenden Instanzen des `Personendetektor`-Dienstes direkt verbunden werden. Jeder Dienst musste zur Laufzeit dann für jeden Raum das Attribut `lastPersonDetected` beobachten, in das der `Personendetektor` den Namen der letzten Person einträgt, der den Raum betreten hat. Ein ähnlicher Ansatz wurde auch für alle anderen Sensoren verfolgt, die Benutzerdaten erzeugen. Im hier vorgestellten Ansatz dient das Benutzermodell hingegen als *Vermittler* zwischen Sensoren und personalisierbaren Diensten. Dadurch werden Top-Level-Dienste von Sensortreibern entkoppelt, wodurch der Entwicklungs- und Wartungsaufwand personalisierbarer Dienste reduziert und ein flexiblerer Einsatz ermöglicht wird. Am Beispiel der Personenerkennung brauchen sich die Dienste nur noch beim Benutzermodell zu registrieren. Sie werden dann automatisch benachrichtigt, wenn eine Person ihren Aufenthaltsort ändert. Sie müssen weder intern eine Benutzerverwaltung implementieren, noch müssen sie spezielle Kenntnisse über den `Personendetektor` besitzen.

Zusammenfassend kann festgehalten werden, dass in einem eHome unterschiedliche Produzenten und Konsumenten von Benutzerdaten existieren. Dabei können beide Rollen gleichzeitig angenommen werden, wie das am Beispiel der personalisierbaren Dienste gezeigt wurde.

4.3 Unterstützung der Intra-eHome-Mobilität

In Abschnitt 2.2.3 wurden eHome-Dienste bezüglich ihrer Zuordnung zu den Architekturschichten Basisdienste, integrierende Dienste und Top-Level-Dienste klassifiziert. Im darauffolgenden Abschnitt wurde eine Einteilung von Top-Level-Diensten in nicht-personalisierbare und personalisierbare Dienste vorgenommen. In diesem Abschnitt wird gezeigt, dass personalisierbare Dienste weiter klassifiziert werden können. Diese Klassifikation wird in diesem Abschnitt erläutert, weil sie die Art der Interaktion der Dienste mit dem Benutzermodell beeinflusst.

Gegenstand der neuen Klassifizierung ist die Frage, wie personalisierbare Dienste die Intra-eHome-Mobilität ihrer Benutzern handhaben. Als Intra-eHome-Mobilität ist hier der Sachverhalt gemeint, dass sich Benutzer innerhalb eines eHomes von Raum zu Raum

bewegen können. Dabei muss das eHome angemessen auf diese Mobilität reagieren, damit Funktionalitäten stets im richtigen Raum bereitgestellt werden. Doch bevor diese Einteilung erläutert wird, werden im Folgenden zunächst zwei für diesen Sachverhalt wichtige Begriffe eingeführt.

4.3.1 Ausführungsort vs. Auswirkungsort

Zur Laufzeit können eHome-Dienste durch ihren Ausführungsort und den Auswirkungsort ihrer angebotenen Funktionalitäten charakterisiert werden. Wie bereist im Grundlagenkapitel erläutert, werden eHome-Dienste auf einem Gateway ausgeführt. Daher ist dieses *Gateway* der *Ausführungsort* aller Dienste eines eHomes und ändert sich zur Laufzeit nicht.

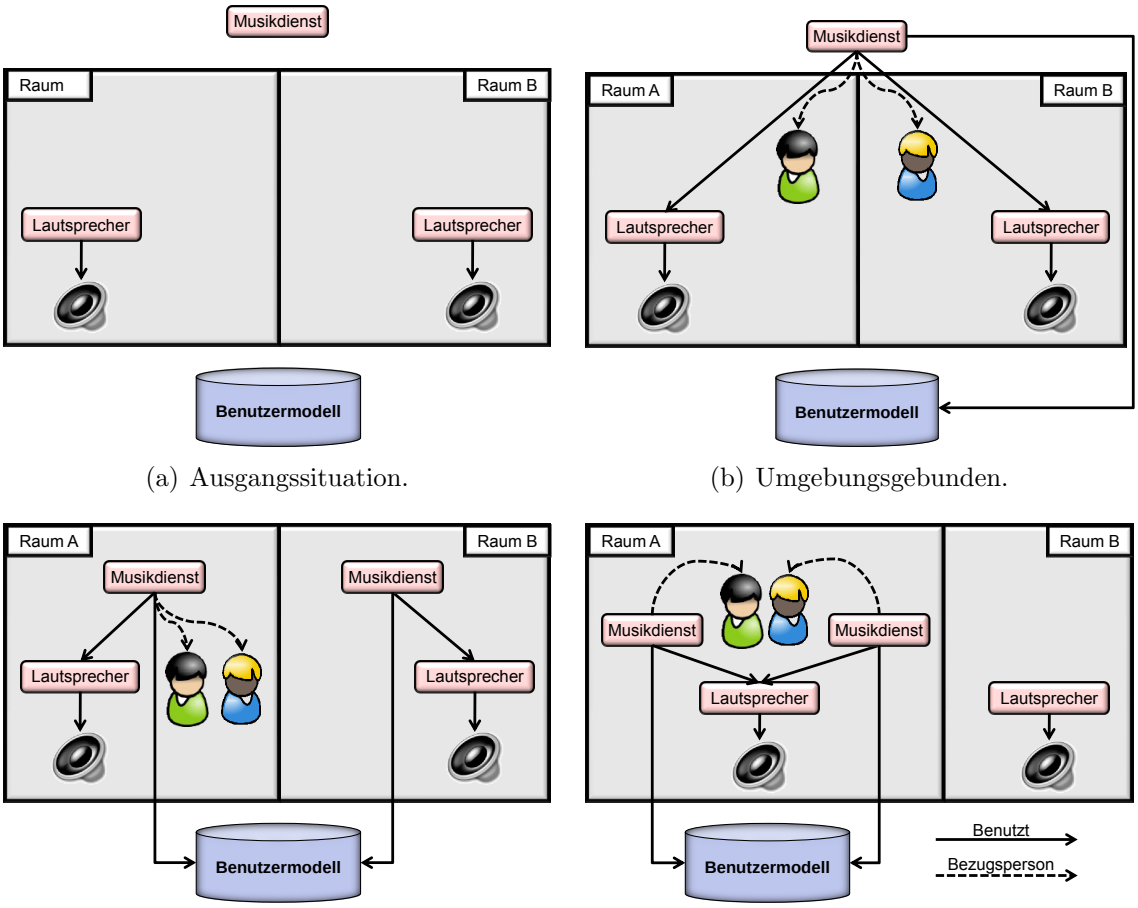
Als *Auswirkungsort* einer Dienstfunktionalität wird dagegen der *Raum* bezeichnet, in dem die Funktionalität bereitgestellt wird. Im Gegensatz zum Ausführungsort eines Dienstes ist der Auswirkungsort einer Funktionalität flexibel und kann sich zur Laufzeit ändern. Beispielsweise spielt der `Musikdienst` die Musik immer in dem Raum, in dem sich der Benutzer befindet. Wenn sich der Benutzer nun in einen anderen Raum begibt, „folgt“ ihm die Musik ebenfalls in den neuen Raum. Also kann sich der Auswirkungsort der Funktionalität `music follows person` (s. Abbildung 3.10) zur Laufzeit ändern.

Weil ein Dienst mehrere Funktionalitäten anbieten kann, können sich die Auswirkungsorte einzelner Funktionalitäten desselben Dienstes unterscheiden. Beispielsweise kann der `Weckdienst` zunächst das Badezimmer vorheizen und dann den Benutzer im Schlafzimmer mit Musik aufwecken. Daher ist es wichtig, das Konzept des Auswirkungsortes feingranular auf der Ebene einzelner Funktionalitäten zu betrachten und nicht auf der Ebene der Dienste.

4.3.2 Typen personalisierbarer Dienste

An dieser Stelle werden nun unterschiedliche Typen personalisierbarer Dienste zur Handhabung der Intra-eHome-Mobilität analysiert. Abhängig davon, wie die Handhabung umgesetzt ist, können sie unterschiedlich konfiguriert werden und benötigen unterschiedliche Informationen aus dem Benutzermodell. Die Analyse wird am Beispiel des `Musikdienstes` durchgeführt. Für die folgende Diskussion wird an einigen Stellen die Kenntnis des eHome-Modells aus Abschnitt 3.2.2 benötigt.

Abbildung 4.3 zeigt verschiedene Möglichkeiten der Konfigurierung des `Musikdienstes`. Die Ausgangssituation, in der der Dienst noch nicht konfiguriert ist, ist in Abbildung 4.3(a) dargestellt. Es zeigt ein eHome mit den Räumen `A` und `B`. In jedem Raum ist ein Lautsprecher mit zugehörigem Basisdienst abgebildet, der für die Audioausgabe verwendet werden kann. Ferner ist das Benutzermodell dargestellt, das sowohl den `Musikgeschmack` als auch den `Aufenthaltsort` der Benutzer enthält. Aus Übersichtsgründen sind der eHomeAdministrator und die `Personendetektoren` nicht dargestellt, die die genannten Daten an das Benutzermodell liefern.



(a) Ausgangssituation.

(b) Umgebungsgebunden.

(c) Raumgebunden.

(d) Personengebunden.

Abbildung 4.3: Unterschiedliche Typen personalisierbarer Dienste zur Handhabung der Intra-eHome-Mobilität am Beispiel des `Musikdienstes`.

4.3.2.1 Umgebungsgebunden

Die erste Möglichkeit zur Handhabung der Intra-eHome-Mobilität besteht darin, personalisierbare Dienste raumübergreifend *genau einmal für das gesamte eHome* zu instanziieren. Solche raumübergreifend konfigurierten Dienste werden als *umgebungsgebunden* bezeichnet. Sie werden im eHome-Modell durch ein ServiceObject realisiert, das an das Environment gebunden wird, das das eHome repräsentiert. Gleichzeitig werden alle für seine Funktionalitäten benötigten Unterdienste fest an dieses ServiceObject gebunden. Am Beispiel des `Musikdienstes` bedeutet das, dass dieser Dienst *alle* im eHome vorhandenen `Lautsprecher` für die Audioausgabe verwenden kann, weil er mit allen verbunden ist. Dieser Sachverhalt ist in Abbildung 4.3(b) durch die durchgestrichenen Pfeile dargestellt.

Ferner ist diese eine Instanz des `Musikdienstes` für *alle* Benutzer im eHome zuständig, wie durch die gestrichelten Pfeile dargestellt ist. Wie Abbildung 4.3(b) außerdem zeigt, benötigt dieser Dienst Informationen aus dem Benutzermodell, weil er bestimmen muss, in welchem Raum welche Musik gespielt werden soll. Das Benutzermodell benachrichtigt den Dienst, wenn ein neuer Benutzer das eHome betritt und diesen Dienst verwenden möchte. Sein `Musikgeschmack` und sein `Aufenthaltsort` werden dem Dienst mitgeteilt,

sodass die Musik im aktuellen Aufenthaltsraum gestartet wird. Wechselt der Benutzer seinen Aufenthaltsort, informiert das Benutzermodell den Dienst erneut, damit die Musik im alten Raum gestoppt und im neuen fortgesetzt werden kann. Somit können für Funktionalitäten umgebungsgebundener Dienste mehrere Auswirkungsorte existieren, wenn mehrere Benutzer den Dienst gleichzeitig aus unterschiedlichen Räumen verwenden. Diese können sich aufgrund der Intra-eHome-Mobilität zudem zur Laufzeit ändern.

Umgebungsgebundene Dienste haben jedoch zwei Nachteile. Erstens werden alle für eine benötigte Funktionalität geeigneten und verfügbaren Ressourcen des eHomes im Voraus belegt, auch wenn sich in einigen Räumen keine Benutzer aufhalten sollten. Dadurch können weitere Dienste unnötig blockiert werden, sodass *Ressourcenkonflikte* entstehen. Beispielsweise könnte ein *TV-Dienst* die belegten Lautsprecher nicht verwenden, obwohl in dem Raum momentan keine Musik gespielt wird. Zweitens muss jeder umgebungsgebundene Dienst seine *eigene Benutzerverwaltung* realisieren. Er muss selbst eine Liste seiner Benutzer führen und selbst dafür sorgen, dass entsprechend der Aufenthaltsorte seiner Benutzer die passende Musik im jeweiligen Raum abgespielt wird. Sinnvoller ist es jedoch, diese Aufgabe in die Laufzeitumgebung zu verlagern, um den Entwicklungsaufwand von Diensten zu reduzieren.

Auf den ersten Blick könnte die eigene Benutzerverwaltung auch als Vorteil aufgefasst werden, weil jeder Dienst dadurch *Benutzerkonflikte* selbst erkennen und lösen kann, etwa durch Priorisierung von Benutzern nach Alter, Verantwortung oder Funktion. Beispielsweise kann der `Musikdienst` in diesem Fall selbst entscheiden, welche Musik gespielt wird, wenn sich mehrere Personen im gleichen Raum aufhalten. Eine einfache Strategie, die auch im Rahmen dieser Arbeit umgesetzt wurde, ist das Abspielen der Musik des Benutzers, der einen Raum zuletzt betreten hat. Die diensteigene Konflikterkennung und -lösung führt jedoch dazu, dass einerseits der Dienstentwicklungsaufwand steigt und andererseits die Vorgabe der Strategien für jeden betroffenen Dienst einzeln vorgenommen werden muss. Eine Auslagerung z. B. in das Benutzermodell würde sowohl den Entwicklungs- als auch den Konfigurierungsaufwand reduzieren.

4.3.2.2 Raumgebunden

Die zweite Möglichkeit zur Handhabung der Intra-eHome-Mobilität, die hauptsächlich von Norbisrath in [Nor07] umgesetzt wurde, besteht darin, personalisierbare Dienste *genau einmal pro Raum* zu instanziieren. Genauer gesagt wird für jeden Raum ein ServiceObject instanziiert, das wiederum an eine Location gebunden ist. Dabei repräsentiert jede Location einen Raum im eHome. Gleichzeitig werden alle benötigten und verfügbaren Unterdienste im jeweiligen Raum permanent an das entsprechende ServiceObject des personalisierbaren Dienstes gebunden. Der Aktionsradius jeder Instanz beschränkt sich daher auf einen einzigen Raum. Solche fest an Räume gebundenen Dienste werden als *raumgebunden* bezeichnet. Das Beispiel in Abbildung 4.3(c) zeigt jeweils eine Instanz des `Musikdienstes` für `Raum A` und `Raum B`, von denen jede mit dem `Lautsprecher` im selben Raum verbunden ist. Der Auswirkungsort der Funktionalitäten raumgebundener Dienste entspricht also dem gleichen Raum, an den die Dienstinstanz gebunden ist, und verändert sich zur Laufzeit nicht.

Instanzen raumgebundener Dienste sind im Gegensatz zu Instanzen umgebungsgebundener Dienste nur für die Benutzer zuständig, die sich in *ihrem* Raum aufhalten. Daher muss jede Instanz benachrichtigt werden, wenn ein Benutzer den Raum betritt oder verlässt. Wie oben fällt auch diese Aufgabe in den Verantwortungsbereich des Benutzermodells. In Abbildung 4.3(c) halten sich beide Benutzer beispielsweise im `Raum A` auf. Was in anderen Räumen passiert, muss die Instanz im `Raum A` nicht wissen.

Raumgebundene Dienste weisen die gleichen Probleme auf wie umgebungsgebundene Dienste. Ressourcenkonflikte treten auch hier auf, weil wieder in allen Räumen Ressourcen belegt werden, auch wenn in einigen Räumen keine Benutzer anwesend sind (s. `Raum B` in Abbildung 4.3(c)). Ferner muss wieder jeder Dienst eine eigene Benutzerverwaltung für den *eigenen* Raum realisieren. Schließlich kann die obige Diskussion bzgl. der Benutzerkonflikte analog auf raumgebundene Dienste angewendet werden, sodass diese hier nicht noch mal aufgegriffen werden muss.

4.3.2.3 Personengebunden

Die dritte Möglichkeit der Handhabung der Intra-eHome-Mobilität besteht darin, personalisierbare Dienste durch jeweils ein ServiceObject zu realisieren, das genau an ein Person-Objekt gebunden wird. Dadurch sind sie nicht mehr an eine Umgebung oder einen Raum gebunden, sondern an *genau einen Benutzer* (*Bezugsperson*) und werden daher als *personengebunden* bezeichnet. Aus diesem Grund muss der Auswirkungsort entsprechender Funktionalitäten auch immer dem Raum entsprechen, in dem sich die Bezugsperson aufhält.

Das heißt für den `Musikdienst`, dass jede Instanz genau für eine Person zuständig ist und immer den `Lautsprecher` ansprechen muss, der sich im selben Raum befindet wie die Bezugsperson. In Abbildung 4.3(d) halten sich beispielsweise beide Personen im `Raum A` auf. Daher sind beide Instanzen des `Musikdienstes` mit dem Lautsprecher im `Raum A` verbunden.

Um sicherzustellen, dass der Auswirkungsort einer Funktionalität einer personengebundenen Dienstinstanz immer dem Raum entspricht, in dem sich die Bezugsperson aufhält, muss das entsprechende ServiceObject zur Laufzeit umkonfiguriert werden. Und zwar immer dann, wenn die Bezugsperson ihren Raum wechselt. Dadurch kann die Funktionalität dem Benutzer folgen und die Ressourcenbelegung wird auf das Nötigste reduziert. Dieser Ansatz wurde von Retkowitz in [Ret10] umgesetzt. Um solche Dienste zur Laufzeit umkonfigurieren zu können, hat er wie in Abschnitt 3.3.1 beschrieben, die Dienstspezifikation erweitert. Mit der erweiterten Spezifikation kann für entsprechende Dienstfunktionalitäten angegeben werden, dass der Auswirkungsort dieser Funktionalität stets mit dem Aufenthaltsraum der Bezugsperson synchronisiert werden muss.

Einen besonderen Vorteil hat dieser Ansatz darin, dass sich ein personengebundener Dienst nicht mehr um die Intra-eHome-Mobilität seiner Bezugsperson kümmern muss. Stattdessen kann er davon ausgehen, dass er von der Laufzeitumgebung (eHomeConfigurator) fortlaufend umkonfiguriert wird, sodass er immer mit den richtigen Unterdiensten verbunden ist und seine Funktionalitäten stets im gleichen Raum verfügbar sind, in dem sich seine Bezugsperson aufhält. Dadurch wird die Dienstentwicklung erheblich vereinfacht.

Umgekehrt kann ein solcher Dienst selbst keine Benutzerkonflikte mehr auflösen, weil er keine Informationen über andere Benutzer im eHome hat. Daher müssen entsprechende Mechanismen durch die Laufzeitumgebung umgesetzt werden. Hier kann das Benutzermodell mit einbezogen werden, weil dort alle benötigten Informationen verfügbar sind. Zum einen kennt das Benutzermodell alle Benutzer und ihre Benutzerdaten. Zum anderen kennt das Benutzermodell die von den Benutzern verwendeten Dienste und die von jedem Dienst benötigten Benutzerdaten. Durch das Hinzunehmen von geeigneten Strategien, die auch im Benutzermodell abgelegt werden können, können Konflikte bezüglich Benutzerpräferenzen einfach im Benutzermodell erkannt und aufgelöst werden. Dadurch müssen die Dienste mögliche Konfliktsituationen nicht berücksichtigen. Stattdessen erhalten sie immer die Daten, die möglicherweise aus einer Konfliktlösung resultiert sind.

4.3.3 Rolle des Benutzermodells

Abhängig davon, ob und wie personalisierbare Dienste die Intra-eHome-Mobilität ihrer Benutzer handhaben, können sie unterschiedlich konfiguriert werden und benötigen unterschiedliche Informationen aus dem Benutzermodell. Dazu gehören einerseits die Präferenzen und andererseits Informationen über die Aufenthaltsorte ihrer Benutzer. Umgebungsgebundene Dienste müssen die Aufenthaltsorte aller Benutzer kennen und überwachen, damit sie selbst entscheiden können, in welchem Raum welche Funktionalität bereitgestellt wird. Raumgebundene Dienste dagegen müssen nur wissen, welche Benutzer *ihren* Raum betreten oder verlassen. Personengebundene Dienste hingegen müssen auf diesen Aspekt gar nicht achten, weil die Laufzeitumgebung für ihre Umkonfigurierung sorgt.

Obwohl Norbisrath personalisierbare Dienste nur raumgebunden und Retkowitz nur personengebunden umgesetzt haben, ist das in dieser Arbeit entwickelte Benutzermodell generisch gehalten. Es kann prinzipiell *alle drei Typen* von Diensten unterstützen. Im Folgenden wird die Rolle des Benutzermodells für die Intra-eHome-Mobilität anhand der Umkonfigurierung eines personengebundenen Dienstes zur Laufzeit veranschaulicht. Der Vorgang ist in Abbildung 4.4 am Beispiel des `Musikdienstes` dargestellt. Auf der linken Seite der Abbildung befindet sich ein Benutzer im `Raum A`, der im Benutzermodell unter dem Namen `Bob` repräsentiert ist. Im Benutzermodell sind weiterhin Bobs `Musikgeschmack` (`Jazz`) und sein `Aufenthaltsort` (`Raum A`) gespeichert. Für diesen Benutzer ist ferner eine Instanz des `Musikdienstes` erstellt und mit dem `Lautsprecher` im `Raum A` verbunden worden. Damit entspricht der Auswirkungsort seiner Funktionalität dem `Raum A`.

Nun wird angenommen, dass sich Bob von `Raum A` nach `Raum B` bewegt (1). Diese Bewegung wird von dem nicht dargestellten `Personendetektor` erkannt und dem Benutzermodell mitgeteilt. Dieser aktualisiert daraufhin intern Bobs Aufenthaltsort (2) und benachrichtigt den eHomeConfigurator, der sich zuvor beim Benutzermodell registriert hat, um genau über solche Ereignisse informiert zu werden (3). Der eHomeConfigurator durchsucht das eHome-Modell und ermittelt alle Instanzen personengebundener Dienste, die mit Bob verbunden sind. Im gegebenen Beispiel handelt es sich nur um die Instanz des `Musikdienstes`. Nun muss dieser Dienst so umkonfiguriert werden, dass der Auswirkungsort seiner Funktionalität `Raum B` entspricht. Dies geschieht in mehreren Schritten:

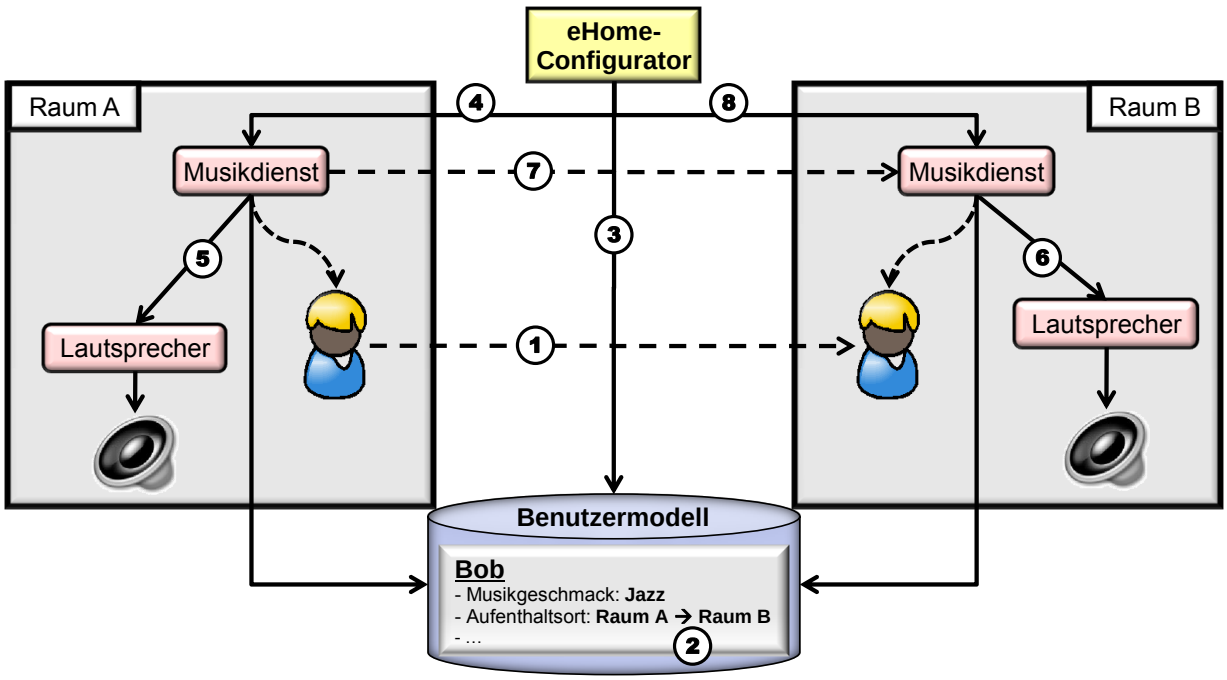


Abbildung 4.4: Umkonfigurierung des personengebundenen `Musikdienstes`.

1. **Pausieren des personengebundenen Dienstes**: In einem ersten Schritt muss die Ausführung des `Weckdienstes` angehalten werden (4). Dies ist nötig, damit dieser nicht versucht, auf seine Unterdienste zuzugreifen, während seine Bindungen zu ihnen geändert werden. In diesem Fall würde dann eine Ausnahme auftreten, da die Funktionalitäten nicht verfügbar wären.

2. **Lösen der Bindungen zu den bisherigen Unterdiensten**: Nachdem die Ausführung angehalten worden ist, kann die eigentliche Umkonfigurierung des `Weckdienstes` im eHome-Modell beginnen. Es werden zunächst die Bindungen zu seinen bisherigen Unterdiensten im `Raum A`, hier `Lautsprecher`, gelöst (5).

3. **Suchen neuer Unterdienste und Erzeugen der entsprechenden Bindungen**: Anschließend werden im eHome-Modell für den neuen `Raum B` Unterdienste ermittelt, die die benötigten Funktionalitäten anbieten. Können auf diese Weise Unterdienste zu allen benötigten Funktionalitäten gefunden werden, baut der Configurator die Bindungen zu diesen Unterdiensten entsprechend auf. Im Beispiel wird eine neue Bindung zum `Lautsprecher` im `Raum B` aufgebaut (6), wodurch sich der Auswirkungsort von `Raum A` nach `Raum B` verschiebt (7). Ist mindestens ein benötigter Unterdienst nicht verfügbar, schlägt die Umkonfigurierung fehl und der Dienst ist im neuen Raum nicht ausführbar.

4. **Fortsetzen des personengebundenen Dienstes**: Schließlich wird die Ausführung des umkonfigurierten Dienstes fortgesetzt (8). Damit ist die Umkonfigurierung abgeschlossen.

Weil diese Umkonfigurierung Teil der strukturellen Adaption der Arbeit von Retkowitz ist, wird für weitere Details auf [Ret10] verwiesen. Die vorliegende Arbeit leistet für die Unterstützung der Intra-eHome-Mobilität insofern einen Beitrag, in dem das hier eingeführte Benutzermodell dem eHomeConfigurator die notwendigen Informationen für die Umkonfigurierung aufbereitet und bereitstellt. Daher ist das Zusammenspiel zwischen dem hier entwickelten Benutzermodell und dem von Retkowitz erweiterten eHomeConfigurator für die Intra-eHome-Mobilität von besonderer Bedeutung. In Kapitel 6 wird die Interaktion des eHomeConfigurator mit dem Benutzermodell noch mal aufgegriffen und es wird gezeigt, wie das Benutzermodell zum Schutz der Privatsphäre beiträgt.

4.4 Verwendete Ontologie

Wie auch in Abschnitt 3.2.2 erläutert wurde, ist in den Vorgängerarbeiten nur ein rudimentäres Konzept zur Modellierung von Benutzern entwickelt worden. Benutzerdaten werden im eHome-Modell lediglich mit einfachen Objekten der Klasse Attribute repräsentiert [Nor07, NARS06]. Beispielsweise kann die Aussage, dass ein Benutzer gern Jazz hört, durch ein Attribut mit dem Namen `favoriteMusic` und dem Wert `Jazz` dargestellt werden.

Durch diesen Ansatz ist es möglich, Benutzerdaten syntaktisch zu strukturieren. Es gab jedoch keinerlei Vorgaben darüber, welche Arten von Benutzerdaten vorhanden sind, welche Begriffe und Konzepte zu ihrer Formalisierung verwendet werden sollen und welche Beziehungen zwischen ihnen existieren. Stattdessen wurde implizit davon ausgegangen, dass Entwickler personalisierbarer Dienste benötigte Attributnamen und -wertebereiche selbst definieren und verwenden.

Dieser Ansatz funktioniert jedoch nur dann, wenn alle personalisierbaren Dienste vom gleichen Hersteller stammen. Er funktioniert nicht, wenn verschiedene Hersteller personalisierbarer Dienste existieren und auch das Benutzermodell von einem anderen Hersteller stammt. Beispielsweise könnten dann zwei Musikdienste existieren, die jeweils den Musikgeschmack ihrer Benutzer benötigen, von denen der eine dafür das Attribut `favoriteMusic` und der andere das Attribut `musicPreference` vorsieht.

Um dem zu begegnen, wird im Rahmen dieser Arbeit eine sogenannte *Ontologie* zur Benutzermodellierung eingesetzt. Durch den Einsatz der Ontologie wird einerseits das Schema des Benutzermodells standardisiert. Mit anderen Worten wird der Aufbau der verwendeten Datenstrukturen festgelegt. Andererseits gibt die Ontologie vor, welche Arten von Benutzerdaten existieren, welche Ausprägungen sie haben können und wie sie mit anderen zusammenhängen. Im Folgenden wird der Ontologie-Begriff näher erläutert, bevor die in dieser Arbeit verwendete Ontologie und ihre Abbildung in den eHome-Prototyp beschrieben wird.

4.4.1 Der Ontologie-Begriff

Der *Ontologie*-Begriff hat seinen Ursprung aus dem Griechischen und bezeichnet die Lehre vom *Sein* bzw. vom *Seienden*, also „von dem, was ist und was nicht ist“ [SN99]. In der

Informatik wird eine Ontologie als *eine konzeptuelle Formalisierung* von Wissensbereichen und Begriffssystemen verstanden. Sie beschreibt mithilfe einer standardisierten Terminologie sowie von Beziehungen zwischen den Begriffen einen Ausschnitt der realen Welt und ermöglicht eine abstrakte, vereinfachte Sicht darauf (s. [Gua98]). Mit anderen Worten legt eine Ontologie fest, welche Dinge existieren und wie sie miteinander zusammenhängen.

Anfang der 90er Jahre wurden Ontologien hauptsächlich im Bereich der künstlichen Intelligenz eingesetzt. Etwa zehn Jahre später wurden sie im Zusammenhang mit dem *Semantic Web* verwendet [BLHL01] und sollten die Kommunikationsgrundlage für Suchmaschinen bilden. Inzwischen werden Ontologien in vielen Anwendungsfeldern wie *Kommunikation, automatisches Schließen* und *Wissensrepräsentation* verwendet. Dabei können jeweils für das Anwendungsfeld zugeschnittene Ontologien zum Einsatz kommen.

Der erste Gedanke, im Rahmen dieser Arbeit eine neue, für das eHome-Projekt zugeschnittene Ontologie für die Benutzermodellierung zu entwickeln, wurde früh verworfen. Erstens ist die Entwicklung einer Ontologie nicht das Hauptziel dieser Arbeit gewesen. Zweitens existieren schon Ontologien, die für den Einsatz im eHome-Projekt geeignet sind (s. für einen Überblick [Kob07]).

4.4.2 General User Model Ontology (GUMO)

Ein Beispiel für solch eine spezielle (engl. *Domain*) Ontologie für das Anwendungsgebiet der Benutzermodellierung ist die von Heckmann et al. entwickelte *General User Model Ontology (GUMO)* [HSB+05]. Die Entwickler hoffen, dass sich GUMO als Standardontologie zur Benutzermodellierung etabliert und von allen benutzeradaptiven Anwendungen gemeinsam verwendet wird. Dadurch würde GUMO die semantische Grundlage für benutzeradaptive Anwendungen bilden und den Austausch von Benutzerdaten unter ihnen vereinfachen. Außerdem würden strukturelle und syntaktische Inkompatibilitäten vermieden werden. GUMO ist ferner Teil der bereichsübergreifenden (engl. *Top Level*) Ontologie *UbisOntology*, die versucht, alle Aspekte abzudecken, die für das Ubiquitous Computing relevant sind. UbisOntology und damit auch GUMO sind im Internet verfügbar und können daher von Anwendungsentwicklern eingesehen und bei Bedarf sogar erweitert werden. Für weitere Informationen sei auf http://www.ubisworld.org verwiesen.

In der aktuellen Version enthält GUMO unterschiedliche Kategorien von Terminologien zur Modellierung von Benutzern, auf die sich benutzeradaptive Anwendungen, zu denen auch personalisierbare eHome-Dienste zählen, beziehen können. Zu diesen Kategorien gehören etwa *demografische Daten*, *Benutzerwissen*, *Benutzerfähigkeiten*, *Benutzerinteressen* und *Benutzerziele*. Beispielsweise können das *Alter*, der aktuelle *Aufenthaltsort*, der *Blutdruck*, der *Musikgeschmack* oder die *Temperaturpräferenz* eines Benutzers für benutzeradaptive Anwendungen relevant sein. Für eine vollständige Auflistung aller in GUMO formalisierten Konzepte sei auf die oben angegebene Webseite verwiesen.

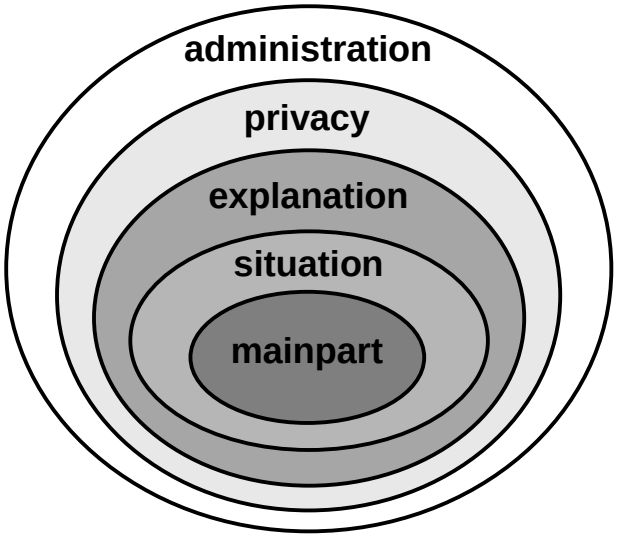


Abbildung 4.5: Hierarchische Struktur eines SituationalStatement (Quelle: [Hec06]).

4.4.3 Die Datenstruktur SituationalStatement

Der Entwurf von GUMO selbst basiert wiederum auf einem Modell, dass Heckmann als *SituationalStatement* bezeichnet. Ein SituationalStatement dient der Beschreibung einer Situation oder Aussage wie sie beispielsweise in einem Benutzermodell vorkommen können [Hec03b, HSB+05]. Sie bildet die Hauptdatenstruktur zur Situationsmodellierung, die als eine Art *Zwiebelmodell* aufgefasst werden kann, wie Abbildung 4.5 zeigt. Durch dieses Modell können die Informationen eines SituationalStatement hierarchisch strukturiert werden.

Jede Schicht der Hierarchie enthält dabei fünf Attribute, die unterschiedliche Bedeutungen haben. Die eigentliche *Aussage* eines SituationalStatement wird in den Attributen der Schicht mainpart gespeichert. Dabei sind die Attribute `subject`, `predicate` und `object` zwingend erforderlich und genügen prinzipiell, um die wichtigsten Aussagen über einen Benutzer formulieren zu können. Beispielsweise kann die Aussage, dass *Bob gern Jazz hört*, durch folgende Attributbelegung formuliert werden:

`subject=bob` & `predicate=favoriteMusic` & `object=jazz`

Ähnlich kann auch der aktuelle Aufenthaltsort eines Benutzers formuliert werden. Beispielsweise wird die Aussage, dass sich *Bob momentan im Schlafzimmer befindet*, folgendermaßen formuliert:

`subject=bob` & `predicate=location` & `object=bedroom`

Optional enthält mainpart noch die Attribute `auxiliary` und `range`. Sie erweitern das Attribut `predicate` um weitere Angaben, sodass der Ontologiedefinition von GUMO mehr Ausdrucksstärke verliehen wird [Hec06]. Der Musikgeschmack eines Benutzers könnte durch Verwendung dieser zusätzlichen Attribute auch folgendermaßen formuliert werden:

`subject=bob` & `auxiliary=hasPreference` & `predicate=music` &
`range=pop-jazz-rock` & `object=jazz`

Die übrigen Schichten aus Abbildung 4.5 können zusätzlich zu mainpart verwendet werden, um die eigentliche Aussage eines SituationalStatement um optionale *Meta-Informationen* zu erweitern. Die Schicht situation dient der zeitlichen und räumlichen Einbettung der entsprechenden Aussage in die reelle Welt[1]. Das Attribut `start` dieser Schicht enthält die Zeit, ab der die Aussage gültig wird (beispielsweise der Beginn einer Benutzer-Sitzung). Entsprechend enthält `end` die Zeit, in der die Aussage ihre Gültigkeit verliert (beispielsweise das Ende einer Benutzer-Sitzung). Passend dazu enthält `durability` die Gültigkeitsdauer der Aussage. Durch diese Informationen können Historien für bestimme Aussagen angelegt und analysiert werden, um aus der Historieanalyse automatisch weitere Informationen zu gewinnen. Von besonderem Interesse für diese Arbeit ist das Attribut `location`. Es beschreibt den Ort, an dem die Aussage gültig ist. In Abschnitt 6.5.1.3 wird beschrieben, wie dieses Attribut verwendet wurde, um Benutzerpräferenzen eHome-spezifisch modellieren zu können. Das Attribut `Aufenthaltsort` hingegen kann zur Angabe von genauen Koordinaten des in `location` angegebenen Ortes verwendet werden.

Die nächste Schicht hat den Namen explanation und enthält die optionalen Attribute `source`, `creator`, `method`, `evidence` und `confidence`. In dieser Arbeit ist insbesondere das Attribut `source` von Interesse. Es wird im Rahmen der Inter-eHome-Mobilität verwendet und dient der Unterscheidung, woher Benutzerdaten stammen: vom Benutzer, von personalisierbaren Diensten oder von Basisdiensten wie etwa dem `Personendetektor`. Weitere Details hierzu folgen in Abschnitt 6.5.1.3. Die übrigen Attribute können verwendet werden, interessierte Benutzer darüber zu informieren, wer eine Aussage erstellt hat (`creator`), welche Methode dafür verwendet wurde (`method`), welche Nachweise für die Aussage vorhanden sind (`evidence`) oder wie zuversichtlich die Aussage ist (`confidence`).

Die Schicht privacy hat eine besondere Bedeutung für diese Arbeit. Ihre Attribute können für den Schutz der Privatsphäre von Benutzern eingesetzt werden. Die Erklärung dieser Attribute und ihrer Verwendung wird in Abschnitt 6.5.1.1 vorgenommen.

Schließlich dienen die Attribute der Schicht administration der effizienten Durchführung organisatorischer Aufgaben in sehr großen Mengen von SituationalStatements. Beispielsweise sind eindeutige Identifikatoren (`id`) für einzelne Aussagen vorgesehen. Alle Attribute dieser Schicht sind optional und werden in dieser Arbeit nicht verwendet, weil sie für die Forschungsziele dieser Arbeit nicht relevant sind.

Im nächsten Abschnitt wird erläutert, wie SituationalStatements in die Architektur des Benutzermodells übertragen wurden.

4.5 Architektur des Benutzermodells

In diesem Abschnitt wird die Architektur des Benutzermodells beschrieben. Die wichtigsten Bestandteile der Architektur und ihre Beziehungen sind in Abbildung 4.6 in Form eines UML-Klassendiagramms [Rum04] dargestellt und werden im Folgenden erläutert.

[1] Der Begriff *SituationalStatement* leitet sich übrigens von dieser Schicht ab.

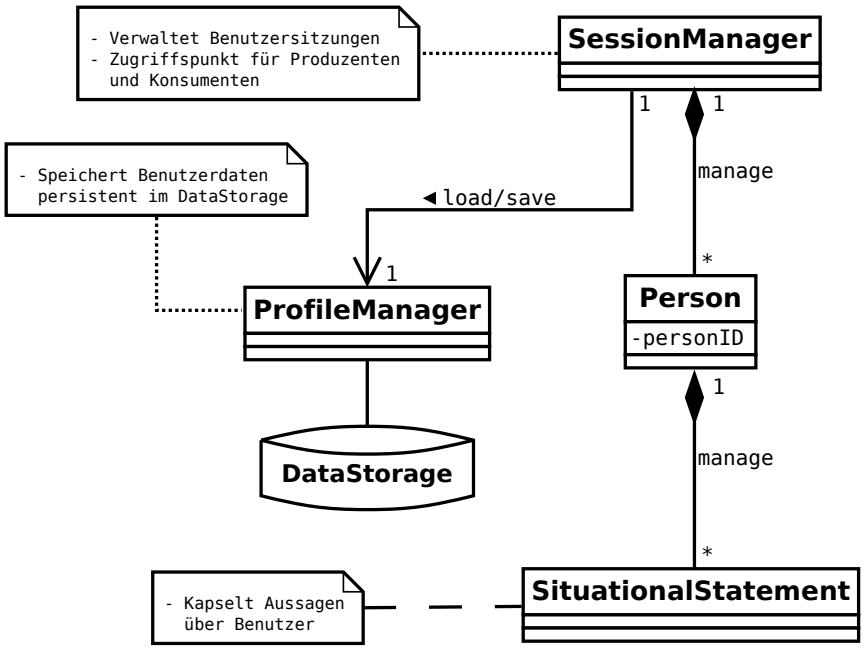


Abbildung 4.6: Ausschnitt aus der Architektur des Benutzermodells.

4.5.1 DataStorage

Ein Kritikpunkt aus Abschnitt 3.4.3 war die Vermischung von Benutzerdaten mit anderen Daten im eHome-Modell. Es war daher nicht möglich, Daten einzelner oder aller Benutzer separat zu verwalten und zu speichern. Das neu entwickelte Benutzermodell sieht nun den eigenen Datenspeicher DataStorage für die persistente Speicherung von Benutzerdaten vor. Dadurch wird die Entkopplung vom eHome-Modell erreicht, sodass die getrennte Verwaltung und Verarbeitung von Benutzerdaten vereinfacht wird.

Die konkrete Speichertechnik, auf der DataStorage aufsetzt, ist in der Abbildung nicht dargestellt. Im Rahmen dieser Arbeit wird die Java-Serialisierung zur persistenten Speicherung von Benutzerdaten verwendet. Serialisiert werden dabei Objekte der Klasse Person und die zugehörigen SituationalStatement-Objekte. Beispielsweise können dadurch die Daten einzelner Benutzer gespeichert und auf Handhelds übertragen werden. Dadurch wird es möglich, Benutzerdaten mitzunehmen und auch in anderen eHomes einzusetzen. Es könnten bei Bedarf jedoch auch andere Techniken wie relationale Datenbanken oder XML-basierte Textdateien für die persistente Speicherung eingesetzt werden.

4.5.2 ProfileManager

Während DataStorage als Datenspeicher fungiert, bietet der ProfileManager die Schnittstelle zum Speichern und Laden von Benutzerdaten an. Andere Bausteine auf dem Gateway können diese Schnittstelle verwenden, um Daten zu speichern oder zu lesen. Dabei abstrahiert der ProfileManager von der konkreten Speichertechnik. Die Bausteine, die den ProfileManager verwenden, müssen daher keine Kenntnis über die verwendete Speichertech-

nik haben. Durch Anwendung von Information Hiding an dieser Stelle wird der Aufwand für zukünftige Erweiterungen und Wartungstätigkeiten minimiert. Würde beispielsweise die Java-Serialisierung durch eine relationale Datenbank ersetzt werden, müsste nur der *Rumpf* des ProfileManagers angepasst werden.

4.5.3 SessionManager

Der wichtigste Baustein des Benutzermodells ist der SessionManager. Während der ProfileManager für die persistente Speicherung von Benutzerdaten zuständig ist, verwaltet der SessionManager die Daten aktiver Benutzer, die sich in einer Sitzung im eHome befinden. Ferner dient er als Zugriffspunkt für Produzenten und Konsumenten von Benutzerdaten.

4.5.3.1 Sitzungen aktiver Benutzer

Als *aktive* Benutzer werden Personen bezeichnet, die sich aktuell in einem eHome aufhalten. Für einen aktiven Benutzer baut der SessionManager eine *Sitzung* (engl. *Session*) auf, die solange aufrecht erhalten wird, bis er das eHome verlässt. Der Aufbau der Sitzung beginnt durch die Erstellung eines Person-Objekts für einen Benutzer, der das eHome betritt. Ist der Benutzer dem eHome schon bekannt und sind seine Daten persistent im DataStorage gespeichert, werden sie über den ProfileManager angefordert und in Form von SituationalStatement-Objekten an das entsprechende Person-Objekt gebunden. Falls gewünscht, werden in der Sitzung geänderte oder neu angelegte Daten mit dem ProfileManager synchronisiert und im DataStorage für zukünftige Sitzungen ablegt. Der SessionManager verwaltet somit die Sitzungen aller aktiven Benutzer im eHome.

4.5.3.2 Zugriffspunkt für Produzenten und Konsumenten

Ferner dient der SessionManager Produzenten und Konsumenten von Benutzerdaten als Zugriffspunkt. Er ermöglicht Produzenten das Anlegen und Verändern von Daten und Konsumenten das Lesen dieser. Insbesondere personalisierbare Dienste verwenden den SessionManager, um auf die Daten ihrer Benutzer zuzugreifen.

Für das Lesen von Benutzerdaten sieht der SessionManager zwei Möglichkeiten vor. Zunächst können Konsumenten konkret den Wert eines bestimmten Attributs anfragen. Diese Möglichkeit setzt voraus, dass der Konsument selbst entscheidet, wann er welche Daten auslesen möchte. Es kann jedoch vorkommen, dass einige Daten zwischenzeitlich aktualisiert werden und der Konsument das nicht mitbekommt. Daher können sich Konsumenten alternativ entsprechend des *Beobachter*-Musters beim SessionManager auf bestimmte Attribute registrieren und werden dann über jede Änderung der Attribute benachrichtigt. Auf diese Weise hat der Konsument stets aktuelle Daten vorliegen.

Produzenten hingegen können entweder schon existierende Attribute eines Benutzers aktualisieren oder neue Attribute anlegen. Falls personalisierbare Dienste Daten anfragen, die noch nicht im Benutzermodell abgelegt wurden, bietet der SessionManager die Möglichkeit, über eine entsprechende Benutzeroberfläche den betroffenen Benutzer nach den Daten zu fragen.

Zusammenfassend realisiert das Benutzermodell mit dem SessionManager eine Art *Tell/Ask*-Schnittstelle für den Informationsaustausch mit anderen Komponenten eines eHomes (s. Abschnitt 2.1.1). Dadurch behebt das Benutzermodell das zuvor im eHome-Modell vorhandene Problem der offenen Datenstrukturen, weil der SessionManager die Interna des Benutzermodells nach Außen kapselt. Daher können die Dienste selbst nicht mehr direkt auf die Datenstrukturen zugreifen. Stattdessen müssen sie sich stets an den SessionManager wenden, um die benötigten Informationen zu erhalten. Falls in Zukunft die interne Darstellung geändert werden sollte, werden die Auswirkungen auf die Produzenten und Konsumenten minimal ausfallen.

4.5.4 Person

Ähnlich wie bei der ursprünglichen Modellierung des Personenkontexts im eHome-Modell (s. Abschnitt 3.2.2.6) wird im Benutzermodell auch eine Klasse Person zur Modellierung von Benutzern verwendet. Jeder Benutzer, der sich in einer aktiven Sitzung befindet, wird durch ein Objekt dieser Klasse repräsentiert. Jedes Objekt besitzt ein eindeutiges Attribut `personID`, wodurch mehrere Benutzer voneinander unterschieden werden können.

Ferner besitzt die Klasse Person *Getter*- und *Setter-Methoden* für Benutzerdaten. Diese Methoden werden vom SessionManager aufgerufen und an das entsprechende SituationalStatement-Objekt weitergeleitet, falls solch eins existiert. Falls es jedoch nicht existiert und eine Getter-Methode aufgerufen wurde, kann der Benutzer zur Eingabe der Information aufgefordert werden. Umgekehrt wird ein neues SituationalStatement-Objekt erstellt, falls eine Setter-Methode für eine nicht-existierende Aussage aufgerufen wurde.

Zusätzlich erlaubt die Person-Klasse die Registrierung von *Listenern* (Beobachter) auf Benutzerdaten. Anfragen auf Registrierungen werden zunächst vom SessionManager entgegengenommen und an die Klasse Person weitergeleitet. Dieser wiederum leitet die Anfrage weiter auf das entsprechende SituationalStatement, der dann die Listener benachrichtigt, wenn sich das entsprechende Datum ändert.

4.5.5 SituationalStatement

Eine signifikante Erweiterung, die das hier eingeführte Benutzermodell im Vergleich zum Personenkontext im eHome-Modell aufweist, ist die Ersetzung der Klasse Attribute durch die Klasse SituationalStatement zur Modellierung von Benutzerdaten. Die Klasse SituationalStatment ist eine Abbildung des gleichnamigen Konzepts aus GUMO (s. Abschnitt 4.4.3).

Das Klassendiagramm der Datenstruktur für Benutzerdaten ist in Abbildung 4.7 dargestellt. In der Mitte des Diagramms ist zunächst die Klasse SituationalStatement abgebildet. Für jeden aktiven Benutzer, der durch ein Objekt der Klasse Person repräsentiert wird (s. Abbildung 4.6), können mehrere SituationalStatement-Objekte angelegt werden, die jeweils eine bestimmte Aussage über den Benutzer enthalten. Die Summe dieser Aussagen ergibt die Menge der im Benutzermodell enthaltenen Daten eines Benutzers[2].

[2]Die Ausdrücke *„Benutzerdaten“* und *„Aussagen über einen Benutzer“* werden in dieser Arbeit synonym gebraucht.

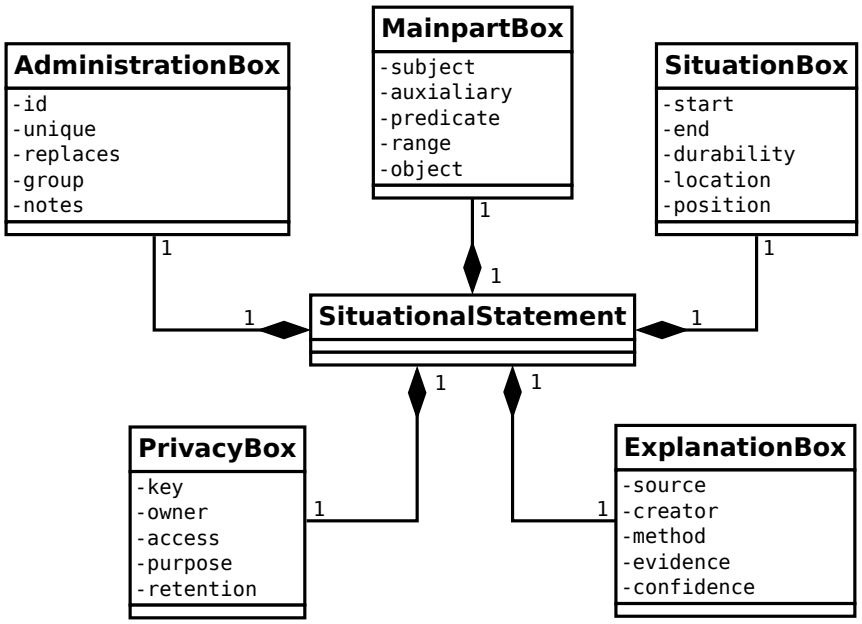


Abbildung 4.7: SituationalStatement und seine Schichten.

Die übrigen Klassen aus Abbildung 4.7 repräsentieren zusammen mit ihren Variablen jeweils eine der in Abschnitt 4.4.3 beschriebenen Schichten des Zwiebelmodells. Daher werden sie an dieser Stelle nicht erneut beschrieben. Im Verlauf dieser Arbeit wird jedoch an den benötigten Stellen Bezug auf diese Attribute genommen, um ihre Rolle bei der Umsetzung bestimmter Konzepte zu erläutern.

Für die im Benutzermodell abgelegten Daten werden die in GUMO definierten Konzepte und Begriffe verwendet, an die sich auch die Produzenten und Konsumenten halten. Dadurch wird gewährleistet, dass alle Beteiligten sowohl in einem eHome als auch eHome-übergreifend das gleiche Verständnis aufweisen und somit Inkompatibilitäten und Inkonsistenzen vermieden werden. Ferner vermeidet das Benutzermodell Redundanzen von Benutzerdaten dadurch, dass alle SituationalStatements eines Benutzers eindeutig sind. Durch die von GUMO vorgegebene Terminologie kann es auch nicht vorkommen, dass die gleichen Daten unter unterschiedlichen Namen angelegt werden können.

4.5.6 Beispiel

Abbildung 4.8 zeigt einen Ausschnitt einer möglichen Ausprägung der Daten im Benutzermodell (s. Abschnitt 4.4.3). Als Beispiel sind die Daten des Benutzers Bob angezeigt, für den ein Person-Objekt erstellt und mit dem SessionManager verbunden wurde. Der SessionManager wird pro eHome nur einmal instanziiert, sodass kein expliziter Bezeichner angegeben ist. Ferner sind zwei SituationalStatement-Objekte abgebildet, die Aussagen über Bobs Musikgeschmack und Aufenthaltsort enthalten. Jedem dieser SituationalStatements ist ferner ein MainpartBox-Objekt zugeordnet, dessen Attribute die Kernaussagen enthalten.

Prinzipiell könnte hier auf die konkrete Belegung des `subject`-Attributs verzichtet werden, weil die Zuordnung eines SituationalStatement zum zugehörigen Benutzer über

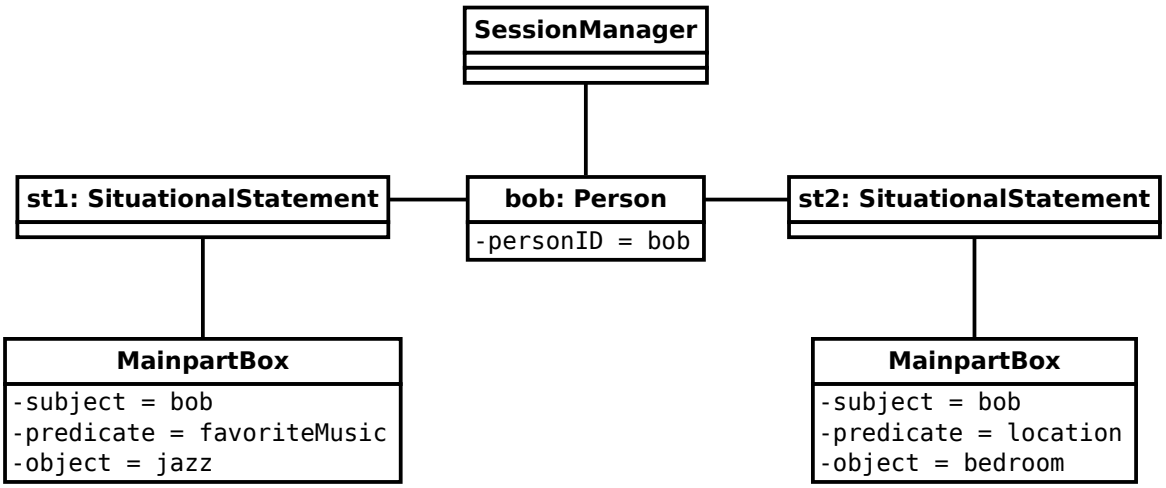


Abbildung 4.8: Ausschnitt einer möglichen Ausprägung von Benutzerdaten im Benutzermodell.

die eindeutige Assoziation zum entsprechenden Person-Objekt gegeben ist. An dieser Stelle wurde nur die MainpartBox gezeigt, weil ihre Attribute zur Repräsentation der Kernaussage ausreichen. Im weiteren Verlauf dieser Arbeit wird an geeigneten Stellen auch auf die Anwendung der übrigen Schichten von SituationalStatements eingegangen.

Abbildung 4.9 zeigt einen möglichen Ablauf einer Abfrage von Benutzerdaten durch einen Konsumenten in Form eines UML-Sequenzdiagramms [Rum04]. Als Beispiel für den Konsumenten dient der `Musikdienst`, der durch das Objekt mfp vom Typ MusicFollowsPerson repräsentiert wird. Wie alle Konsumenten muss sich auch dieser Dienst an den SessionManager wenden, um Daten anzufragen. In diesem Beispiel möchte der Musikdienst Bobs Musikgeschmack ermitteln. Dazu ruft er die Methode `getStmnt(favMusic,bob)` auf, wobei der erste Parameter für das `predicate`-, der zweite für das `subject`- und der erwartete Rückgabewert für das `object`-Attribut der MainpartBox des entsprechenden SituationalStatement steht. Der SessionManager ermittelt aus der Liste der aktiven Benutzer das zugehörige Person-Objekt und leitet die Anfrage für den Musikgeschmack an diesen weiter. Das Person-Objekt selbst ermittelt wiederum das passende SituationalStatement-Objekt, das Bobs Musikgeschmack enthält. Falls dieser existiert, wird die Anfrage an ihn weitergeleitet. Schließlich wird der gesuchte Wert aus der zugehörigen MainpartBox ermittelt. In diesem Beispiel hat Bob den Musikgeschmack `jazz`, der in der umgekehrten Reihenfolge bis zum aufrufenden Dienst weitergeleitet wird.

Falls kein SituationalStatement existiert, das die gesuchte Information enthält, benachrichtigt das Benutzermodell den Benutzer über eine entsprechende Benutzerschnittstelle und ermöglicht ihm das Anlegen der benötigten Information.

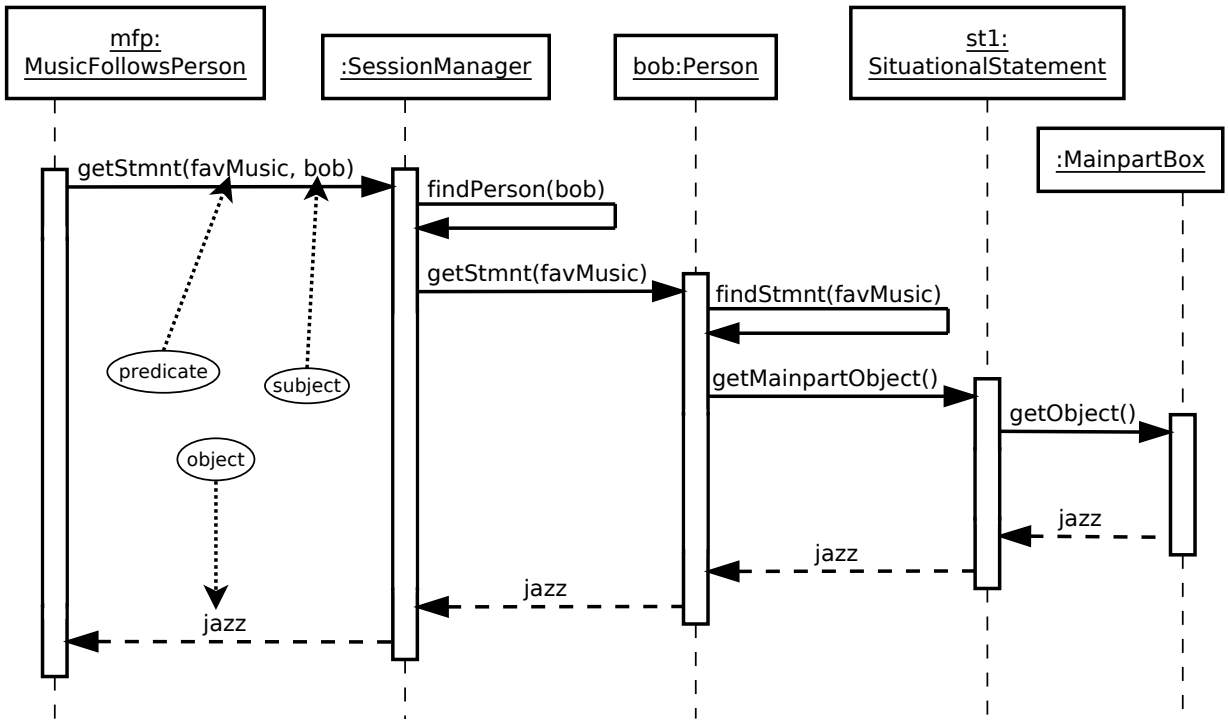


Abbildung 4.9: Möglicher Ablauf einer Abfrage von Benutzerdaten am Beispiel des `Musikdienstes`.

4.6 Zusammenfassung

In diesem Kapitel wurde ein Benutzermodell vorgestellt, das nun allein für die Verwaltung von Benutzerdaten verantwortlich ist. Dadurch wurden Benutzerdaten wie gefordert vom eHome-Modell *entkoppelt*. Das Benutzermodell stellt jetzt auch sicher, dass keine Daten mehr *redundant* vorkommen können, was im eHome-Modell noch möglich war und zu erhöhtem Aufwand bei der Datenpflege geführt hat. Durch die Anwendung von Datenabstraktion auf das Benutzermodell wurde die geforderte *lose Kopplung* des Benutzermodells mit den Produzenten und Konsumenten von Benutzerdaten erzielt. Ferner wurde mit dem Einsatz der Benutzermodellontologie GUMO die Grundlage dafür geschaffen, dass in eHomes eine konsistente Terminologie für Benutzerdaten verwendet wird. Darüber hinaus wurde das Benutzermodell so entworfen, dass in einem eHome unterschiedliche Diensttypen zur Handhabung der Intra-eHome-Mobilität ausgeführt werden können. Abschließend wurde die Architektur des Benutzermodells beschrieben.

Das in diesem Kapitel entwickelte Benutzermodell bildet die Grundlage für die Entwicklung von weiteren Konzepten zur Unterstützung der Inter-eHome-Mobilität sowie zum Schutz der Privatsphäre. In den folgenden Kapiteln wird daher auf das Benutzermodell zurückgegriffen.

Kapitel 5

Inter-eHome-Mobilität

Nachdem im letzten Kapitel das entwickelte Benutzermodell zur Verwaltung von Benutzerdaten vorgestellt wurde, widmet sich dieses Kapitel der Unterstützung von mobilen Benutzern im Rahmen der Inter-eHome-Mobilität. Wie das Beispiel in Abschnitt 1.2 gezeigt hat, ist diese Mobilität ein Resultat des Tagesablaufs eines vermeintlich „normalen“ Menschen, der zur Arbeit oder auf Geschäftsreisen geht, einkaufen muss oder Freunde besucht. Sie unterscheidet sich von der Intra-eHome-Mobilität insofern, als dass sich die Benutzer nicht *in einem* eHome von Raum zu Raum begeben, sondern von eHome zu eHome und „*unterwegs*“ eHome-Dienste nutzen (s. Abbildung 5.1).

Weil die Inter-eHome-Mobilität in den Vorgängerarbeiten jedoch nicht berücksichtigt wurde, sind die Ergebnisse dieser Arbeiten auf Benutzer zugeschnitten, die sich stets in einem eHome aufhalten. Insbesondere wurde angenommen, dass dem eHome zur Deployment-Zeit von Diensten schon alle Benutzer bekannt sind, also keine neuen Benutzer das eHome betreten oder verlassen werden. Sowohl aus der Benutzer- als auch aus der eHome-Perspektive ergeben sich daher Probleme, wenn die Inter-eHome-Mobilität unterstützt werden soll. Diese Probleme wurden in Abschnitt 3.4 detailliert diskutiert.

Dieses Kapitel widmet sich der Ausarbeitung neuer oder der Erweiterung existierender Konzepte zur Unterstützung der Dienstnutzung im Rahmen der Inter-eHome-Mobilität. Diesen Konzepten liegen dabei mehrere Annahmen zugrunde: Die Hauptannahme bildet die Tatsache, dass der Benutzer überhaupt *mobil* ist. Diese Annahme bildet gleichzeitig auch die Hauptmotivation dieser Arbeit und wird als gegeben betrachtet. Die übrigen Annahmen sind wie folgt:

- Zunächst wird angenommen, dass mobile Benutzer auch *unterwegs Funktionalitäten* von eHome-Diensten *nutzen* möchten. Dabei werden Funktionalitäten abstrakt und unabhängig von Herstellern betrachtet, sodass es irrelevant ist, welcher konkrete Dienst hinter einer bestimmten Funktionalität steckt. Eine mögliche Kategorisierung von Dienstfunktionalitäten findet sich z. B. in Abschnitt 3.2.2.1 und in [Ret10].

- In diesem Zusammenhang ist es prinzipiell unwichtig, wo sich der Ausführungsort (s. Abschnitt 4.3.1) eines Dienstes befindet. Vielmehr ist es wichtig, dass der *Auswirkungsort* einer Funktionalität dem Ort entspricht, in der die Funktionalität genutzt werden soll.

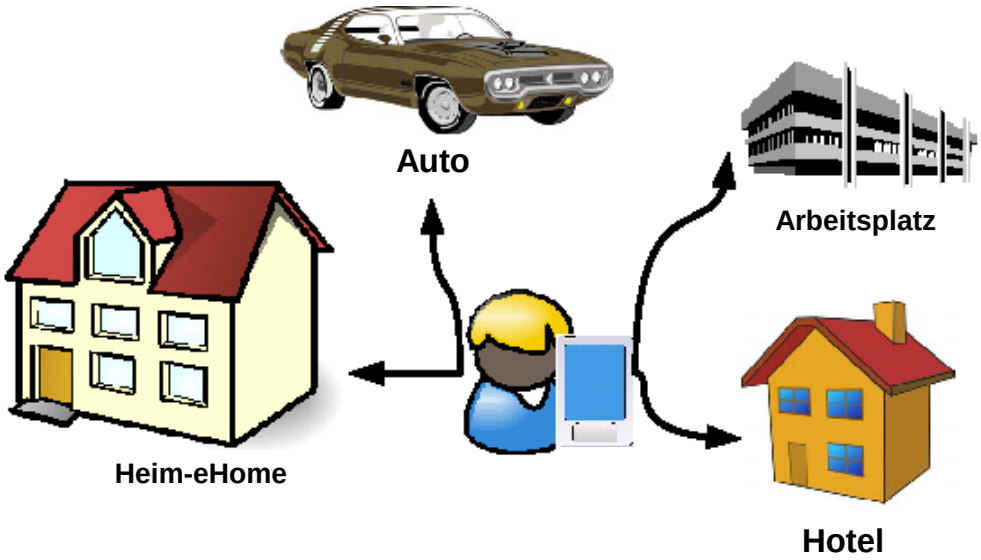


Abbildung 5.1: Inter-eHome-Mobilität.

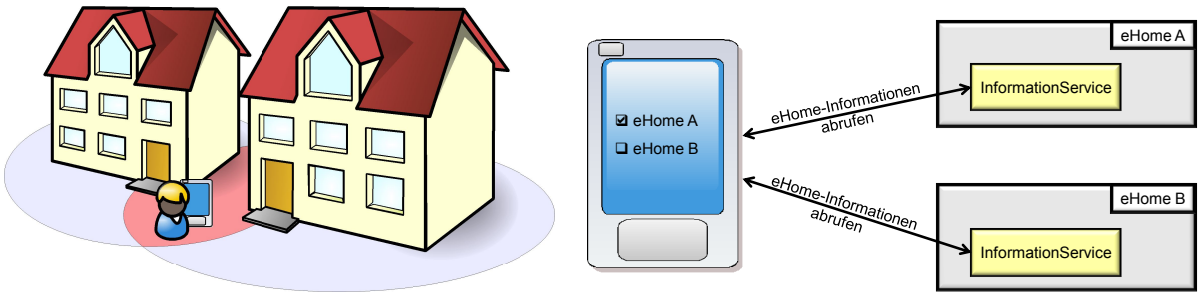
- Ferner können besuchte eHomes *unterschiedlicher Natur* sein (Wohnung, Hotel, Arbeitsplatz etc.). Es wird angenommen, dass alle besuchten eHomes über die erforderliche Infrastruktur zum Betrieb von eHome-Diensten verfügen und diese auch mobilen Benutzern zur Verfügung stellen.

- Schließlich wird hier angenommen, dass jeder Benutzer wie in Abbildung 5.1 dargestellt ein *Handheld* bei sich trägt, etwa ein PDA oder Handy. Dieses Gerät soll über drahtlose Kommunikationsmöglichkeiten wie etwa Bluetooth oder WLAN verfügen und komplexe Anwendungen ausführen können. Diese Annahme ist aus zwei Gründen realistisch. Erstens finden Handhelds immer mehr Akzeptanz und Verbreitung [oe209]. Zweitens nimmt die technische Entwicklung dieser Geräte stetig zu. Beispielsweise hat das *iPhone* von Apple inzwischen mehr Fähigkeiten als ein Computer vor zehn Jahren.

Die zur Unterstützung mobiler Benutzer entwickelten Konzepte basieren grundsätzlich auf der drahtlosen Kommunikation von Handhelds und eHomes, wie in den folgenden Abschnitten beschrieben wird. Zunächst werden in Abschnitt 5.1 die Konzepte zum Auswählen von und Interagieren mit Diensten beschrieben. Anschließend wird in Abschnitt 5.2 die Personalisierung von eHomes mit einem Handheld erläutert. Schließlich wird in Abschnitt 5.3 die Realisierung der Konzepte beschrieben. Ein Schwerpunkt wird hierbei auf die verteilte Objektkommunikation gelegt.

5.1 Dienstauswahl und -interaktion

Wie ein eHome eingerichtet und in Betrieb genommen werden kann, wurde schon in Kapitel 3 erläutert. Dafür wurden im eHome-Projekt bereits mehrere Werkzeuge entwickelt, deren Funktionalitäten der eHomeConfigurator zusammenfasst.



(a) Erkennen eines eHomes über WLAN.

(b) Der InformationService stellt Informationen über das eHome bereit.

Abbildung 5.2: Erkennen und Auswählen eines eHomes.

Ferner wurde in Abschnitt 3.4.2 erwähnt, dass bislang drei unterschiedliche Möglichkeiten existieren, mit eHome-Diensten zu interagieren: erstens *explizit* über Schalter oder Regler von Geräten, zweitens *implizit* über von Sensoren erfasste Kontextänderungen und drittens wiederum *explizit* durch Parametrisierung von Diensten über den eHomeConfigurator. In diesem Abschnitt wird ein neues Konzept vorgestellt, das diese Möglichkeiten so erweitert, dass nun auch das Handheld für die Auswahl und Interaktion verwendet werden kann. Die Interaktion soll dabei *explizit* über eine *dienstspezifische* Benutzeroberfläche erfolgen. Dabei wird angenommen, dass das eHome bereits eingerichtet ist und sich in Betrieb befindet.

5.1.1 Erkennen und Auswählen eines eHomes

Bevor ein Benutzer mit seinem Handheld Dienste eines eHomes verwenden kann, muss das Handheld mit dem eHome verbunden werden. Hierfür muss das Handheld erkennen können, ob sich überhaupt ein eHome in der Nähe des Benutzers befindet. Zu diesem Zweck wurde ein Discovery-Ansatz entwickelt, der folgenden Ablauf vorsieht: Zunächst erkennt das Handheld automatisch frei verfügbare Drahtlosnetzwerke, die es automatisch nach eHome-Gateways durchsucht. Wurde solch ein Gateway gefunden, wird der Benutzer informiert.

Jedes eHome, dass mobilen Benutzern den Zugriff auf Dienste gewähren möchte, verfügt dabei über einen sogenannten InformationService, der eine eindeutige Kennung und eine Beschreibung für das eHome bereitstellt. Über diese Informationen kann der Benutzer erkennen, um welches eHome es sich handelt, bevor er sich dort anmeldet. Dies ist beispielsweise in Situationen wie in Abbildung 5.2 hilfreich, falls sich in der Nähe des Benutzers mehrere eHomes befinden, die von dem Handheld gleichzeitig erkannt werden. Wie in Abbildung 5.2(b) dargestellt ist, hat sich der Benutzer in diesem Fall für eHome A entschieden.

Üblicherweise wird ein eHome verlangen, dass sich der Benutzer zunächst anmeldet, bevor er die Dienste des eHomes in Anspruch nehmen kann. Wie solch eine Anmeldung aussehen kann, wird im nächsten Kapitel näher beschrieben. Im folgenden Unterabschnitt wird beschrieben, wie sich ein bereits angemeldeter Benutzer einen Dienst zum Bedienen auswählen kann.

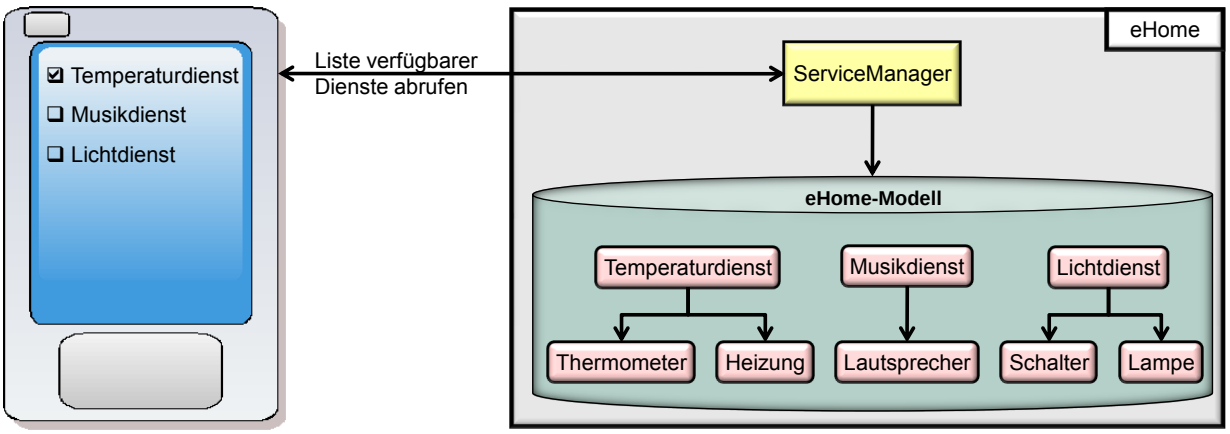


Abbildung 5.3: Auswahl eines Dienstes über das Handheld.

5.1.2 Auswahl eines Dienstes

Abbildung 5.3 stellt eine Situation dar, in der sich der Benutzer die Liste der im eHome verfügbaren Dienste anzeigen lässt. Um diese Liste abrufen zu können, interagiert das Handheld mit dem ServiceManager. Der ServiceManager wurde im Rahmen dieser Arbeit entwickelt und dient dem Handheld als Anlaufstelle für alle Angelegenheiten, die eHome-Dienste betreffen. Hierzu gehört auch die Erstellung der Liste von Diensten, die per Handheld bedient werden können. Die Dienste, die in die Liste aufgenommen werden sollen, ermittelt der ServiceManager durch Analyse der aktuellen Dienstkomposition im eHome-Modell.

Beispielsweise sind in Abbildung 5.3 die drei Top-Level-Dienste `Musikdienst`, `Temperaturdienst` und `Lichtdienst` zusammen mit ihren Unterdiensten dargestellt. In die Liste, die auf das Handheld übertragen wird, werden nur Top-Level-Dienste aufgenommen, die für die Nutzung durch Handhelds vorgesehen sind. Die Unterdienste brauchen dem Benutzer nicht angezeigt zu werden. Falls sich Benutzeraktionen auf die Unterdienste auswirken sollen, werden sie von den entsprechenden Top-Level-Diensten weitergeleitet.

Auf der linken Seite der Abbildung 5.3 ist dargestellt, dass der Benutzer auf seinem Handheld den `Temperaturdienst` ausgewählt hat. Als Nächstes wird erläutert, wie er mit diesem Dienst interagieren kann.

5.1.3 Interaktion mit einem Dienst per Handheld

Abbildung 5.4 zeigt schematisch, wie ein Benutzer mit dem zuvor ausgewählten Dienst interagieren kann. Auf dem Display des Handhelds ist die für den `Temperaturdienst` *spezifische Benutzeroberfläche* dargestellt. Zu sehen ist dabei sowohl die aktuelle Raumtemperatur (für das Schlafzimmer) als auch ein grafischer Schieberegler, mit dem er die gewünschte Temperatur einstellen kann. Falls der Benutzer eine neue Temperatur einstellen möchte, kann er dies mit dem Handheld tun. Seine Auswahl wird anschließend dem `Temperaturdienst` auf dem eHome-Gateway mitgeteilt. Dieser kann die Heizung dann so regeln, dass die gewünschte Temperatur erreicht wird.

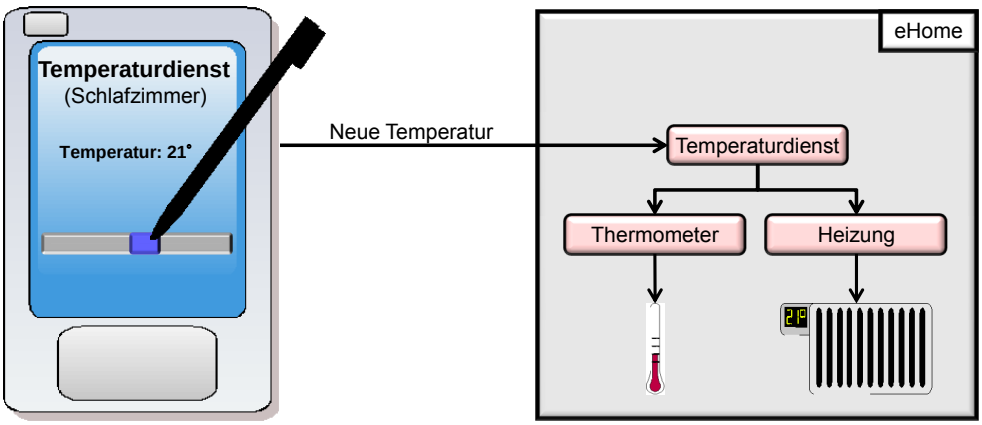


Abbildung 5.4: Interaktion mit einem Dienst über das Handheld.

Diese neue Möglichkeit der Interaktion mit eHome-Diensten bietet zwei Vorteile: Erstens müssen die Benutzer nicht mehr zwangsläufig physische Schalter oder Regler bedienen. Stattdessen können sie ihr Handheld als eine Art universelle *Fernbedienung* oder *Schaltzentrale* einsetzen, um alle wichtigen Funktionen des eHomes von einem beliebigen Raum aus aufrufen. Beispielsweise können die Beleuchtung und die Raumtemperatur über virtuelle Schieberegler eingestellt, vom Wohnzimmer aus das Bad eingelassen oder die Haustür geöffnet, die Jalousien bedient oder die sich im Keller befindende Waschmaschine angeschaltet werden. Diese Liste lässt sich beliebig fortsetzen. Einzige Voraussetzung hierfür ist die Vernetzung der Geräte über das Gateway und die Existenz entsprechender Treiberdienste. Ferner können sich die Benutzer über ihr Handheld anzeigen lassen, was im aktuellen eHome passiert. Beispiele hierfür sind die Anzeige der aktuellen Raumtemperatur, des Wasserstands in der Badewanne oder auch des aktuellen Energieverbrauchs der elektronischen Geräte. Auch die Industrie hat inzwischen die Vorteile einer solchen Schaltzentrale erkannt. Beispielsweise integriert die Firma Metz solch eine Funktionalität in den Fernseher [Met10].

Zweitens können auf diese Weise *einheitliche* Benutzerschnittstellen zum eHome-übergreifenden Einsatz realisiert werden. Beispielsweise kann festgelegt werden, dass für die Einstellung der gewünschten Raumtemperatur stets ein Schieberegler verwendet wird. Für `Musikdienste` könnte z. B. festgelegt werden, wie die Bedienelemente aussehen und wie sie angeordnet sein sollen. Die Vereinheitlichung von Benutzerschnittstellen (einheitliches *Look and Feel*) führt dazu, dass sich mobile Benutzer auch in fremden eHomes ohne großen Aufwand zurechtfinden. Sie müssen nämlich einmal erlernte Funktionen nicht erneut lernen und können Dienste so effizient nutzen. Dadurch wird die Akzeptanz von eHomes erhöht.

Um ein einheitliches Look and Feel zu erzielen, müssen sich die Dienstentwickler jedoch auf eine gemeinsame Gestaltung von Benutzeroberflächen einigen. Eine mögliche Vorgehensweise wär die Erstellung einer allgemein akzeptierten Ontologie, die sich z. B. von der im vorigen Kapitel erwähnten UbisOntology ableitet. Eine weitere Möglichkeit ist die Verwendung von Beschreibungssprachen für Dienstinteraktionen, aus denen zur Laufzeit Benutzeroberflächen generiert werden können. Ein ähnlicher Ansatz wurde beispielsweise in [Gab04] vorgeschlagen. Der hier verwendete Ansatz wird in Abschnitt 5.3.7 beschrieben.

5.2 Personalisierung

In diesem Abschnitt wird die Personalisierung von eHomes im Rahmen der Inter-eHome-Mobilität diskutiert. Der hierfür entwickelte Ansatz lässt sich in zwei Aspekte unterteilen. Zunächst wird erklärt, wie Benutzerdaten eHome-übergreifend bereitgestellt werden können und dabei das Redundanzproblem vermieden werden kann. Hierfür wird ein mobiles Benutzermodell eingesetzt. Anschließend wird diskutiert, wie das Handheld zum Mitnehmen und Ausführen persönlicher eHome-Dienste genutzt werden kann, falls das besuchte eHome nicht alle gewünschten Funktionalitäten anbietet.

5.2.1 Personalisierung durch Benutzerdaten

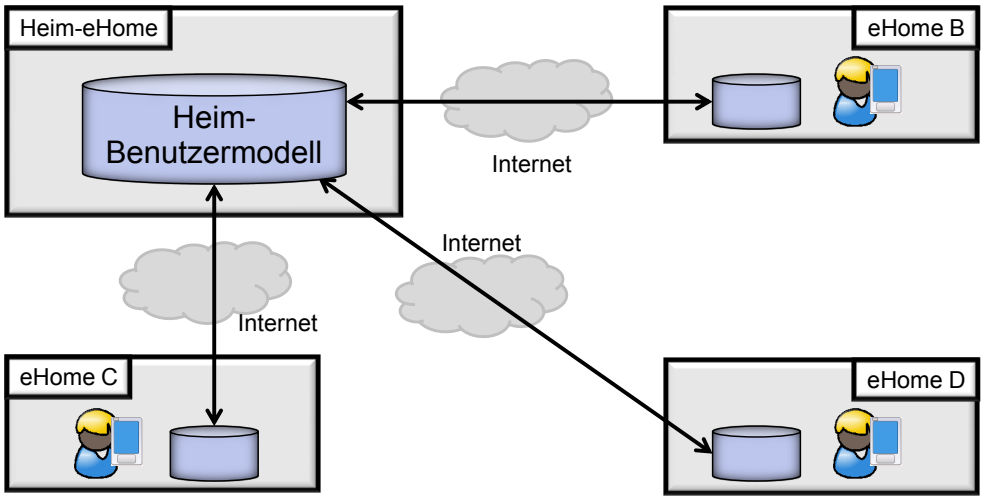
Wie in Abschnitt 3.4.3 diskutiert wurde, sind die von dem eHomeConfigurator vorgesehenen Möglichkeiten zur Personalisierung von eHome-Diensten nur eingeschränkt für die Unterstützung mobiler Benutzer geeignet. Kritisiert wurde einerseits die schlechte Benutzbarkeit, weil der eHomeConfigurator nicht für normale Benutzer konzipiert wurde, sondern für Administratoren und Spezialisten. Andererseits müssen Benutzerdaten in den eHomes redundant gespeichert werden, wenn die Personalisierung mit dem eHomeConfigurator durchgeführt wird. Im Rahmen der Inter-eHome-Mobilität führt diese Redundanz zu einem erhöhten Aufwand für die Verwaltung von Benutzerdaten.

In diesem Abschnitt wird nun ein Ansatz vorgestellt, der das Redundanzproblem löst. Prinzipiell kann das Redundanzproblem dadurch gelöst werden, dass Benutzerdaten nicht mehr in jedem eHome separat gespeichert werden. Stattdessen sollten sie an einer bestimmten Stelle vorgehalten werden und allen eHomes zur Verfügung gestellt werden, in denen der Benutzer personalisierbare Dienste verwendet. Hierfür wird auf das im letzten Kapitel eingeführte Benutzermodell zurückgegriffen.

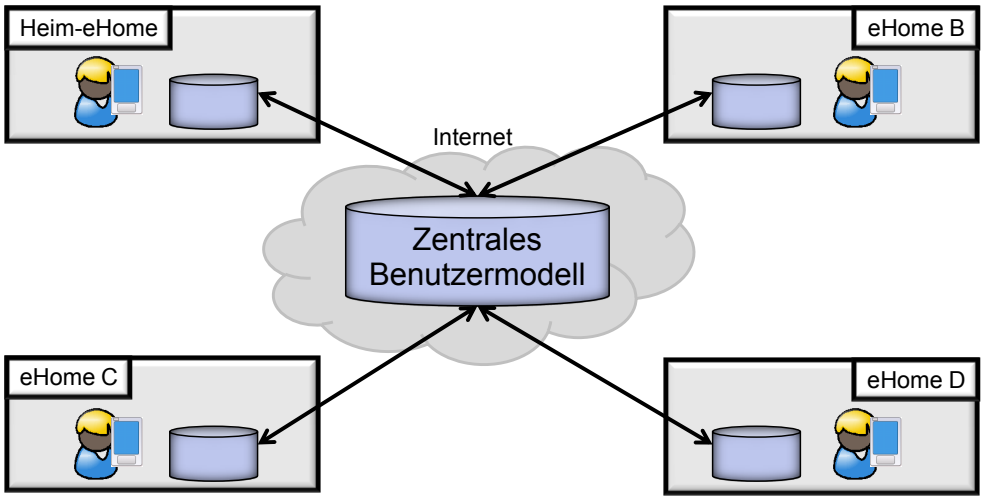
5.2.1.1 Möglichkeiten zum Lösen des Redundanzproblems

Es ist eine spannende Frage, zu klären, *wo* die Benutzerdaten am besten vorgehalten werden sollen. In Abbildung 5.5 sind drei Möglichkeiten dargestellt, die infrage kommen. In allen drei Fällen wird ein Benutzer angenommen, der mobil ist und sein Handheld dabei hat. Ferner existiert in jedem eHome eine Instanz des Benutzermodells, welche die Daten aller Benutzer verwaltet, die sich in dem eHome aufhalten. Ihre Unterschiede werden im Folgenden erläutert.

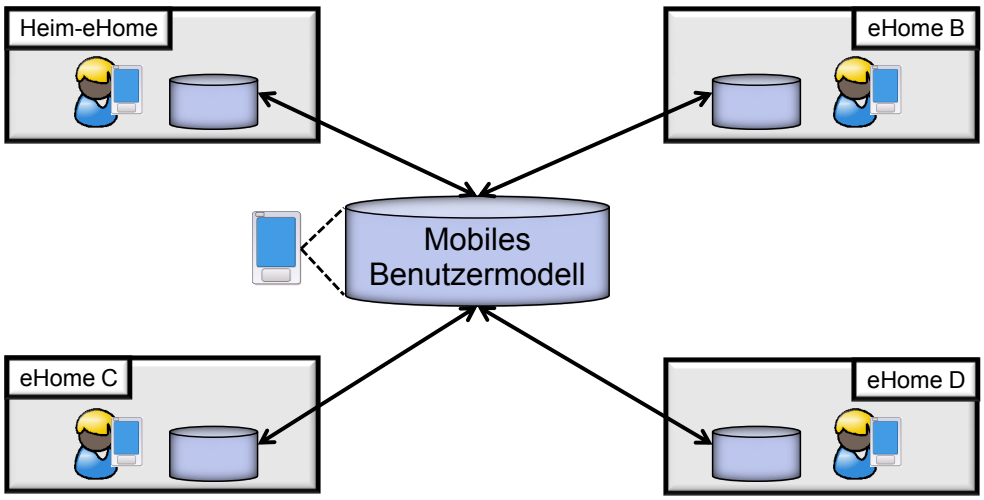
Benutzermodell im Heim-eHome Die erste Möglichkeit besteht darin, Benutzerdaten stets im *Heim*-eHome des Benutzers zu belassen (s. Abbildung 5.5(a)). Immer wenn der Benutzer ein neues eHome betritt, kann er dieses autorisieren, benötigte Daten von zu Hause abzurufen. Der Benutzer kann seine Daten entweder daheim oder unterwegs z. B. über eine Webschnittstelle bearbeiten. Auch Änderungen der Daten im besuchten eHome werden mit dem Heim-eHome synchronisiert. Weil somit die Daten nicht mehr in jedem eHome separat verwaltet werden müssen, tritt das Redundanzproblem nicht mehr auf.



(a) Benutzerdaten werden vom Benutzermodell im Heim-eHome verwaltet.



(b) Benutzerdaten werden vom zentralen Benutzermodell im Internet verwaltet.



(c) Benutzerdaten werden vom mobilen Benutzermodell auf dem Handheld verwaltet.

Abbildung 5.5: Mögliche Ansätze zur Lösung des Redundanzproblems.

Zentrales Benutzermodell im Internet Eine weitere mögliche Lösung für das Redundanzproblem sieht vor, die Daten aller eHome-Benutzer von einem *zentralen* Benutzermodell verwalten zu lassen und allen eHomes Zugriff auf diese Daten zu gewähren (s. Abbildung 5.5(b)). Betritt nun ein Benutzer ein neues eHome, kann der Benutzer dem eHome mitteilen, unter welchem Namen oder Pseudonym seine Daten beim zentralen Benutzermodell angefragt werden können. Seine Daten kann er z. B. über eine Webschnittstelle verwalten. Somit vermeidet auch dieser Ansatz das Redundanzproblem. Solch ein Ansatz wird beispielsweise in [HSBK05] vorgeschlagen. Dort stellt das sogenannte *u2m.org* benutzeradaptiven Anwendungen über das Internet Dienste eines zentralen Benutzermodells bereit. Die Anwendungen können dabei weltweit verteilt sein.

Mobiles Benutzermodell auf Handheld Die dritte Möglichkeit besteht darin, für jeden Benutzer ein eigenes, *mobiles* Benutzermodell einzusetzen (s. Abbildung 5.5(c)). Dadurch kann jeder Benutzer seine Daten auf seinem Handheld mitnehmen, damit sie stets dort verfügbar sind, wo er sich aufhält. Wenn er in einem eHome personalisierbare Dienste verwenden möchte, kann das eHome die benötigten Daten von seinem Handheld abrufen. Auch hier besteht die Möglichkeit, per Synchronisierung sicherzustellen, dass die Daten auf dem Handheld stets aktuell sind. Somit tritt das Redundanzproblem nicht mehr auf. Eine Kombination dieses Ansatzes mit der ersten Möglichkeit ist auch denkbar, sodass die Daten mit dem Heim-Benutzermodell synchronisiert werden können, weil die Bearbeitung der Daten auf einem üblichen Rechner komfortabler ist.

Alle der zuvor genannten Möglichkeiten sind technisch realisierbar und wurden im Rahmen dieser Arbeit prototypisch getestet. Die ersten beiden Möglichkeiten scheiden jedoch aus dem Grund aus, weil sie mehr Möglichkeiten zur Verletzung der Privatsphäre bieten. Im ersten Fall müsste der Benutzer jedem besuchten eHome mitteilen, woher er kommt. Dadurch könnten sich mehrere eHomes zusammenschließen und Bewegungsprofile eines Benutzers erstellen. Im zweiten Fall müssen alle Benutzerdaten dem zentralen Benutzermodell anvertraut werden. Ein bösartiger Betreiber des zentralen Benutzermodells kann sowohl die vorhandenen Daten missbrauchen als auch zusätzliche Daten über mobile Benutzer erlangen, indem er beispielsweise Bewegungsprofile erstellt, die nachweisen, wann sich welcher Benutzer in welchem eHome aufgehalten hat. Ferner müssen in beiden Fällen das Heim-Benutzermodell oder das zentrale Benutzermodell über das Internet verfügbar sein. Bei einem möglichen Ausfall der Verbindung oder der Benutzermodelle könnten keine Benutzerdaten mehr angefragt werden.

Das mobile Benutzermodell bietet hingegen die Möglichkeit, neben dem Lösen des Redundanzproblems auch die Privatsphäre der Benutzer zu wahren. Daher ist die Entscheidung zugunsten des mobilen Benutzermodells gefallen. Im Folgenden wird erläutert, wie dieser Ansatz in den eHome-Prototyp integriert wurde. Die detaillierte Beschreibung der Aspekte zum Schutz der Privatsphäre findet in Kapitel 6 statt.

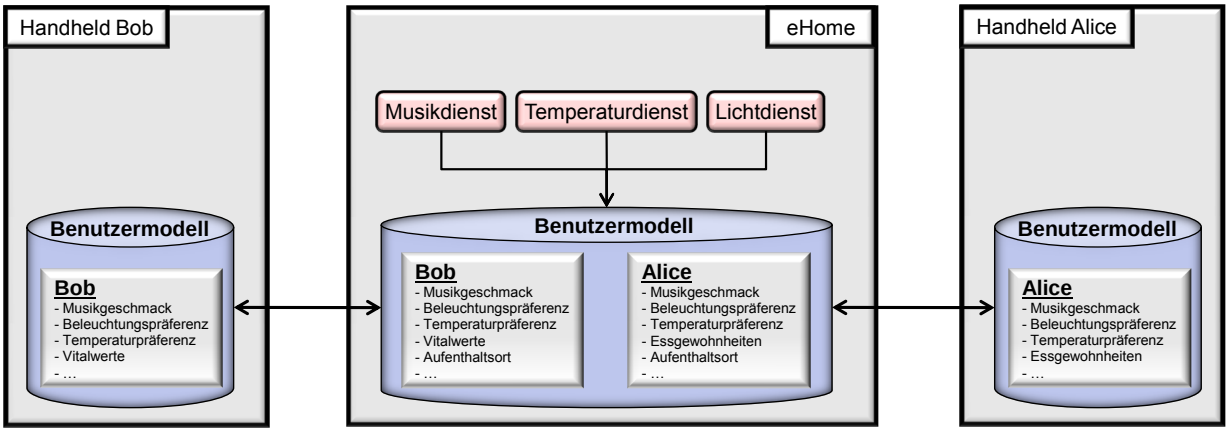


Abbildung 5.6: Zusammenspiel mobiler Benutzermodelle mit dem Benutzermodell auf dem eHome-Gateway.

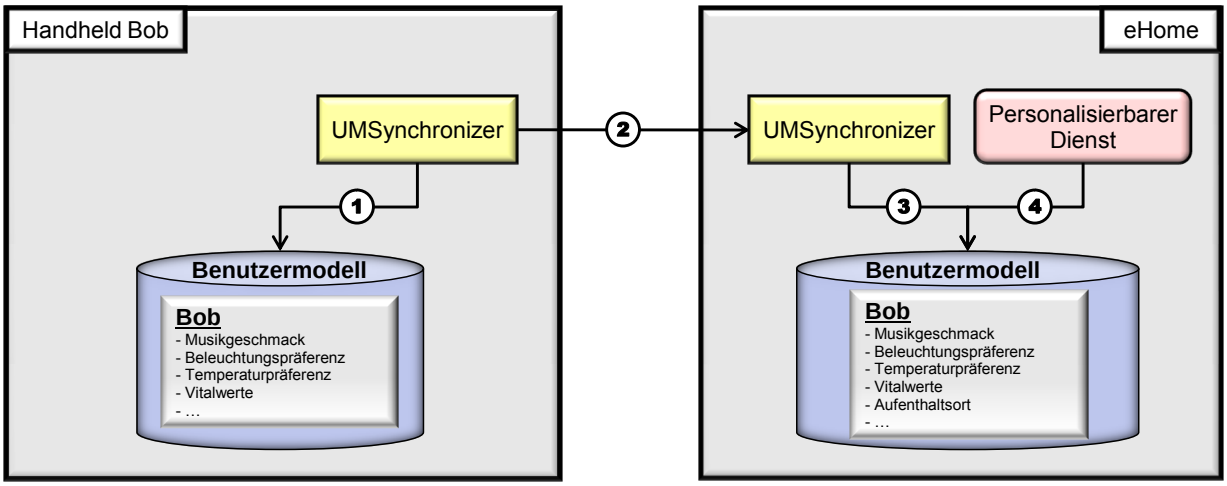
5.2.1.2 Integration des mobilen Benutzermodells in den eHome-Prototyp

Dadurch, dass nun ein mobiles Benutzermodell im Rahmen der Inter-eHome-Mobilität eingesetzt wird, um Benutzerdaten eHome-übergreifend verfügbar zu machen, entsteht die Frage, wie dieser in den eHome-Prototyp integriert werden kann. Abbildung 5.6 zeigt skizzenhaft das Zusammenspiel von mobilen Benutzermodellen und dem Benutzermodell auf dem Gateway eines eHomes. Dabei fällt auf, dass das Benutzermodell im eHome die Daten *mehrerer* aktiver Benutzer gleichzeitig verwaltet. Das *mobile* Benutzermodells verwaltet hingegen die Daten *eines* Benutzers. Beispielsweise sind auf dem linken Handheld nur Bobs Daten enthalten und auf dem rechten Handheld die Daten von Alice.

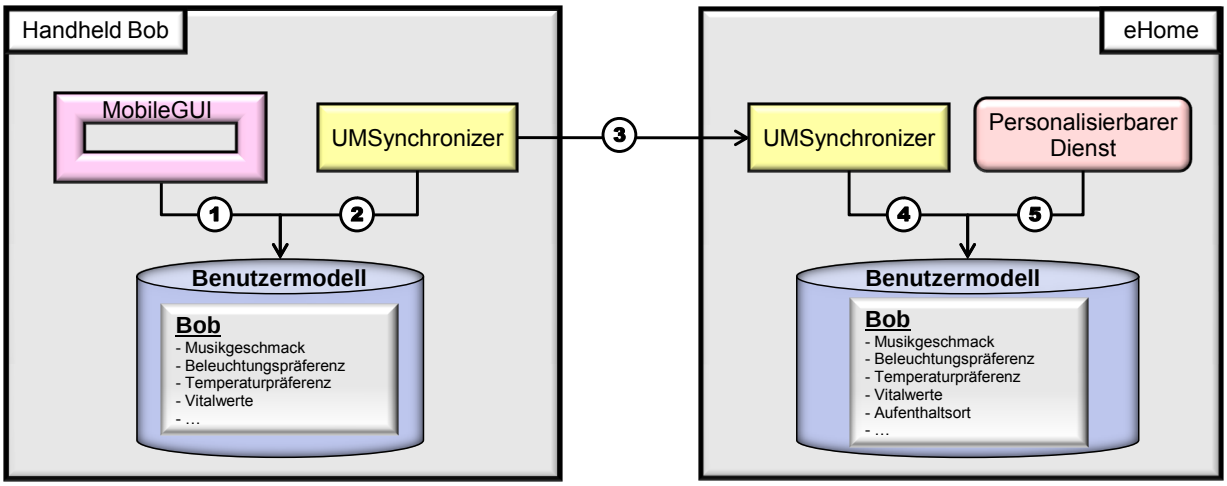
Durch den Einsatz des mobilen Benutzermodells kann nun jeder Benutzer seine Daten auf dem Handheld mitnehmen, sodass sie immer verfügbar sind. Wie weiterhin in Abbildung 5.6 zu sehen ist, erhalten personalisierbare Dienste ihre Daten stets über das Benutzermodell auf demselben Gateway. Auf diese Weise wird die Verteilung der Daten vor den Diensten verborgen. Dadurch wird die Entwicklung der Dienste vereinfacht, weil sie davon ausgehen können, dass die Daten stets lokal vorhanden sind. Wie die Daten dorthin gelangen, ob über Sensoren oder über das mobile Benutzermodell, ist für sie nicht relevant.

Gleichzeitig bedeutet dies, dass es einen Mechanismus geben muss, der einen Austausch der Daten zwischen eHomes und Handhelds ermöglicht. Da die Daten in beide Richtungen übertragen werden können, wird dieser Vorgang als *bidirektionale Synchronisierung* bezeichnet. Um die Synchronisierung durchzuführen, wurde der UMSynchronizer entwickelt, von dem sowohl auf dem Handheld als auch im eHome jeweils eine Instanz existiert. Beide können bei Bedarf eine Synchronisierung anstoßen und sind über die gesamte Dauer der Sitzung des Benutzers miteinander verbunden. Folgende Ereignisse können eine Synchronisierung auslösen (s. Abbildung 5.7):

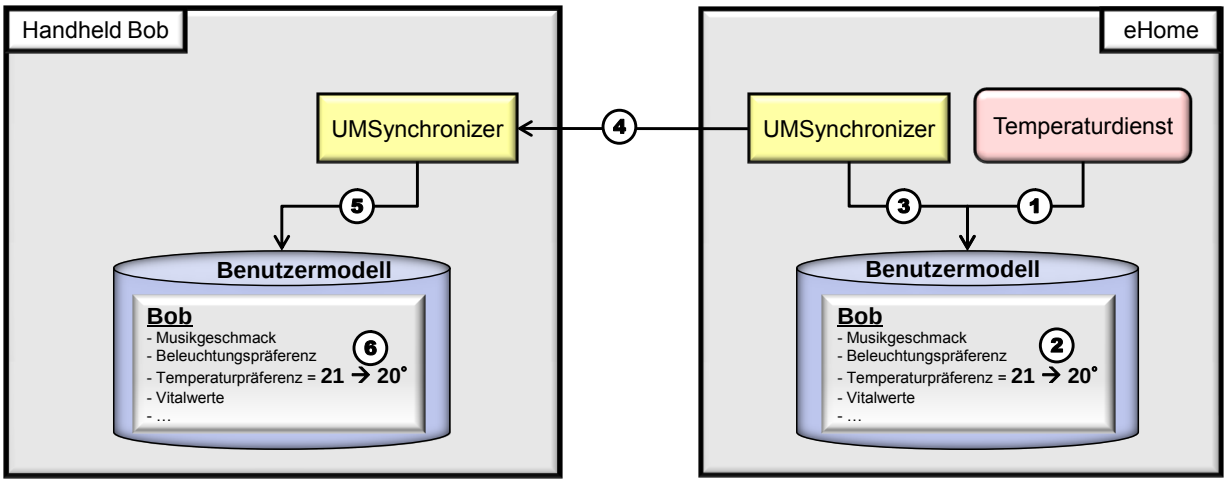
- **Anmeldung**: Der erste Austausch zwischen einem Handheld und einem eHome findet während der Anmeldung des Benutzers am eHome statt. In dieser Phase werden



(a) Benutzerdaten werden beim Anmelden synchronisiert.



(b) Benutzerdaten werden synchronisiert, wenn Attributwerte auf dem Handheld geändert werden.



(c) Benutzerdaten werden synchronisiert, wenn Attributwerte im eHome geändert werden.

Abbildung 5.7: Ereignisse, die eine Synchronisierung von Benutzerdaten nach sich ziehen.

die Daten vom Handheld in das eHome übertragen, um sie dort den Diensten zur Verfügung zu stellen. Wie in Abbildung 5.7(a) dargestellt ist, liest der UMSynchronizer auf dem Handheld nach der Anmeldung die Daten aus dem mobilen Benutzermodell aus (1) und überträgt sie dem UMSynchronizer im eHome (2). Letzterer übermittelt diese Daten dann dem Benutzermodell im eHome (3). Anschließend sind diese Daten für die personalisierbaren Dienste im eHome verfügbar (4). Im nächsten Kapitel wird beschrieben, wie die Menge der übertragenen Daten minimiert werden können, um einen besseren Schutz der Privatsphäre zu bieten.

- **Aktualisierung auf dem Handheld**: Ein weiteres Ereignis, das eine Synchronisierung auslösen kann, ist die Änderung von Attributen auf dem Handheld nach der Anmeldung, also während der aktiven Sitzung. Zusätzlich zur direkten Interaktion über dienstspezifische Benutzeroberflächen ergibt sich durch die Einführung des mobilen Benutzermodells nun die Möglichkeit, auch *indirekt* mit einem Dienst zu interagieren, und zwar durch das Bearbeiten der Präferenzen auf dem Handheld. Hierfür bietet die im Rahmen dieser Arbeit entwickelte MobileGUI eine grafische Benutzerschnittstelle, über die der Benutzer seine Daten verwalten, also Attribute anlegen, löschen oder ändern kann. Falls er beispielsweise über die MobileGUI seine Musikpräferenz ändert (1) (s. Abbildung 5.7(b)), erkennt der UMSynchronizer dies (2) und teilt die Änderungen umgehend seinem Gegenüber mit (3). Dieser benachrichtigt wiederum das Benutzermodell im eHome (4). Anschließend werden alle personalisierbaren Dienste über die Änderung informiert, deren Funktionalität von den geänderten Attributen abhängt. Dadurch wird sichergestellt, dass alle Dienste im eHome auf solche Änderungen reagieren können und stets mit den aktuellen Benutzerdaten arbeiten. Hierfür müssen sich sowohl der UMSynchronizer als auch die Dienste beim Benutzermodell als Beobachter registrieren, um über Änderungen der Daten benachrichtigt zu werden.

- **Aktualisierung im eHome**: Attributänderungen können auch im eHome beispielsweise durch Dienste vorgenommen werden. Ein Beispiel hierfür ist der `Temperaturdienst`, der aus ökologischen Gründen die Raumtemperatur um einen Grad niedriger einstellt als vom Benutzer angegeben. Hat z. B. das Attribut `Temperaturpräferenz` aus Abbildung 5.7(c) den Wert 21°C, heizt der Dienst den Raum auf 20°C (1). Ist dieser mit der Temperatur zufrieden und beschwert sich nicht, kann der Dienst den Wert des Attributs im Benutzermodell auf 20°C umstellen (2) und den Benutzer benachrichtigen. Falls er damit einverstanden ist, wird die Änderung auch auf dem Handheld nachgezogen (3–6). Dadurch wird sichergestellt, dass sich stets die aktuellen Präferenzen auf dem Handheld befinden und in Zukunft wiederverwendet werden können, sowohl in anderen eHomes als auch im aktuellen eHome. Der `Musikdienst` kann aber auch eigene Attribute anlegen, die zuvor nicht auf dem Handheld des Benutzers vorhanden waren. Beispielsweise kann er ein Attribut zum Speichern der aktuellen Position der Wiedergabeliste anlegen, sodass die Musik bei einem Raumwechsel an der gleichen Stelle fortgesetzt werden kann. Solche Attribute sind benutzerspezifisch und gehören daher in das Benutzermodell. Ob diese Attribute auch auf das Handheld

übertragen werden sollen, liegt im Ermessen des Benutzers. Einerseits gewinnt er dadurch mehr Komfort, weil so der Übergang zwischen zwei Sitzungen fließender gestaltet werden kann. Andererseits ergeben sich hierdurch zusätzliche Gefahren in Bezug auf die Privatsphäre, wie im nächsten Kapitel diskutiert wird.

Zusammenfassend kann festgehalten werden, dass das Redundanzproblem durch die Einführung des mobilen Benutzermodells und der bidirektionalen Synchronisierung gelöst wurde. Jeder Benutzer kann nun seine Daten stets mit sich führen und den eHomes automatisch zur Verfügung stellen. Verglichen mit den Ansätzen von Norbisrath [Nor07] und Retkowitz [Ret10] reduziert sich der Aufwand zum Personalisieren von eHomes dadurch erheblich.

Weiter oben wurde argumentiert, dass das mobile Benutzermodell sich besser zum Schutz der Privatsphäre eignet als die anderen beiden Alternativen. Im nächsten Kapitel wird erläutert, wie durch den Einsatz des Handhelds die Risiken diesbezüglich vermindert bzw. vermieden werden können. Dabei wird auch die bidirektionale Synchronisierung noch mal aufgegriffen und es wird analysiert, wann und im welchem Umfang sich eine Synchronisierung mit dem Schutz der Privatsphäre vereinbaren lässt.

5.2.2 Personalisierung durch persönliche Dienste

Im vorangegangenen Abschnitt wurde diskutiert, wie ein mobiles Benutzermodell zur Mitnahme von Benutzerdaten eingesetzt werden kann, um diese zum Personalisieren besuchter eHomes zu nutzen. Dabei wurde angenommen, dass die vom Benutzer gewünschten Funktionalitäten im aktuellen eHome verfügbar sind. In diesem Abschnitt wird hingegen der Fall betrachtet, dass in dem eHome ein oder mehrere der von einem mobilen Benutzer gewünschten Funktionalitäten nicht verfügbar sind.

Nun stellt sich die Frage, wie den Benutzern die gewünschten Funktionalitäten trotzdem angeboten werden können. Im Rahmen dieser Arbeit wurden zwei alternative Ansätze analysiert, die dies ermöglichen. Dabei liegen beiden Ansätzen folgende Annahmen zugrunde:

- Eine von einem mobilen Benutzer gewünschte Funktionalität ist im besuchten eHome nicht verfügbar.
- Der Benutzer ist im Besitz eines *persönlichen* Dienstes, der ähnliche Funktionalitäten realisiert, und verwendet diesen zu Hause.
- Im besuchten eHome sind Unterdienste bzw. Geräte vorhanden, die von diesem Dienst benötigt werden.
- Das besuchte eHome ermöglicht dem Besucher die Nutzung dieser Unterdienste und/oder Geräte.

Ausgehend von diesen Annahmen sehen beide Ansätze die Verwendung persönlicher Dienste für die Personalisierung besuchter eHomes vor. Der Benutzer soll in beiden Fällen die Funktionalitäten des Dienstes so verwenden, als würde er direkt in dem aktuellen eHome

ausgeführt. Es soll für ihn keinen Unterschied machen, wo der Dienst ausgeführt wird. Im Folgenden werden beide Ansätze zunächst kurz voneinander abgegrenzt. Anschließend wird der hier umgesetzte Ansatz detailliert beschrieben.

Zur Veranschaulichung der Fragestellung wird dabei folgendes Beispiel herangezogen. Angenommen, der Geschäftsreisende Bob möchte den `Weckdienst`, den er daheim verwendet, auch in seinem Hotelzimmer nutzen. Ebenfalls wird angenommen, dass dieser Dienst im Hotel nicht angeboten wird.

Das Konzept einer virtuellen eHome-Umgebung Der erste Ansatz sieht den Aufbau einer *virtuellen eHome-Umgebung* vor, in der alle eHomes, die ein mobiler Benutzer besucht, über ein Netzwerk miteinander verknüpft sind. Dadurch können Dienste verteilt komponiert werden, sodass die Auswirkungsorte von Funktionalitäten stets dem Aufenthaltsort des Benutzers entsprechen, auch wenn sich der Benutzer nicht in dem eHome befindet, in dem die Dienste ausgeführt werden. Dies ist dann der Fall, wenn ein Top-Level-Dienst und mindestens einer seiner Unterdienste nicht im selben eHome ausgeführt werden. Am Beispiel des `Weckdienstes` würde das bedeuten, dass der `Weckdienst` im Heim-eHome des Benutzers ausgeführt wird aber die Heizung, das Licht, die Jalousie usw. im Hotelzimmer ansteuert, in der sich Bob befindet. Die für den Benutzer wichtigen Funktionalitäten sind also dort verfügbar, wo er sich aufhält. Damit verlagert sich der Auswirkungsort des Dienstes vom Heim-eHome des Nutzers in das Hotel. Dieser Ansatz ermöglicht mobilen Benutzern nicht nur die Verwendung persönlicher Dienste unterwegs, sondern auch die Nutzung von Diensten anderer eHomes zu Hause.

Mitnahme persönlicher Dienste Der zweite Ansatz hingegen sieht keine Verknüpfung von eHomes vor, sondern basiert auf der Verwendung des Handhelds, das sich neben der Mitnahme von Benutzerdaten auch für die Mitnahme *persönlicher Dienste* eignet. Nimmt ein Benutzer beispielsweise seine persönlichen Dienste wie etwa den `Weckdienst` in Form eines Bundles (s. Abschnitt 2.1.4.1) auf seinem Handheld mit, kann er diesen unterwegs nutzen. Dabei hat er zwei Möglichkeiten. Entweder er überträgt den Dienst auf das Gateway des besuchten eHomes, sodass er dort konfiguriert und deployt werden kann, oder er führt ihn auf seinem Handheld aus. Im zweiten Fall agiert das Handheld als mobiles Gateway, weshalb eine Möglichkeit zur *verteilten* Komponierung von Diensten existieren muss.

Prinzipiell ist der Aufbau einer virtuellen eHome-Umgebung zum Personalisieren besuchter eHomes durch persönliche Dienste geeignet. Doch erfordert dieser Ansatz die Verknüpfung von eHomes. Wie weiter oben schon diskutiert wurde, birgt solch eine Verknüpfung zusätzliche Gefahren in Bezug auf die Privatsphäre. Denn so könnten fremde eHomes mehr Informationen über den Benutzer gewinnen, als für die Nutzung der gewünschten Funktionalitäten erforderlich ist. Dazu gehört beispielsweise die Erstellung eines Bewegungsprofils durch kooperierende eHomes. Daher wurde im Rahmen dieser Arbeit ein Ansatz realisiert, der die Mitnahme persönlicher Dienste auf dem Handheld ermöglicht. Dabei werden beide Varianten unterstützt, also sowohl das Übertragen der Dienste in das besuchte eHome als auch das Ausführen der Dienste auf dem Handheld. Beide haben dabei ihre Vor- und Nachteile und sollten daher miteinander in Einklang gebracht werden. In

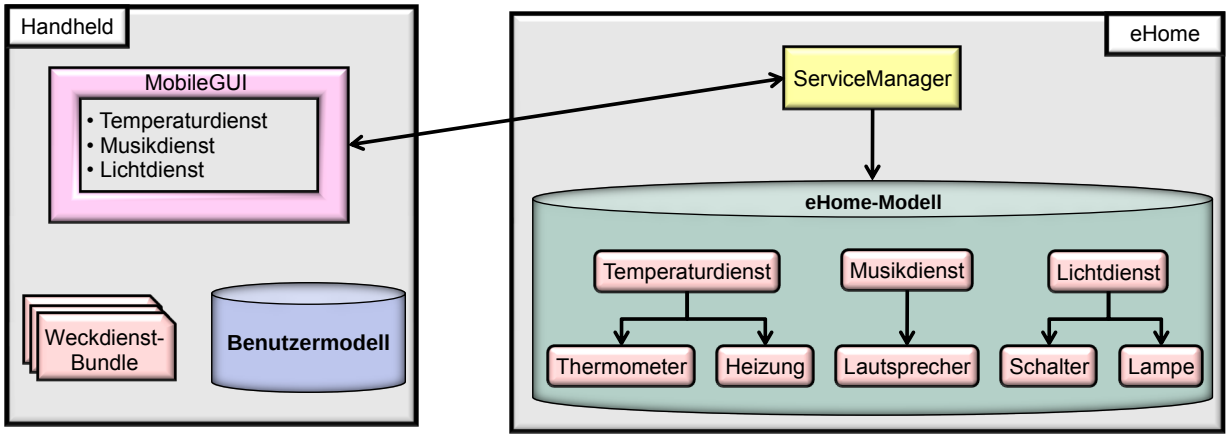


Abbildung 5.8: Anzeige der im eHome verfügbaren Dienste.

den folgenden Unterabschnitten wird detailliert beschrieben, wie sie in den bestehenden Prototyp integriert wurden.

5.2.2.1 Ausführung persönlicher Dienste im besuchten eHome

In diesem Unterabschnitt wird zunächst beschrieben, wie persönliche Dienste in das besuchte eHome übertragen und dort ausgeführt werden können. Abbildung 5.8 zeigt das erwähnte Hotelbeispiel. Auf der rechten Seite ist das eHome-Modell mit den Kompositionen der im Hotel verfügbaren Dienste `Temperaturdienst`, `Lichtdienst` und `Musikdienst` dargestellt. Diese sind schon mit passenden Unterdiensten zum Steuern von Geräten verbunden und befinden sich in Ausführung.

Auf der linken Seite ist Bobs Handheld dargestellt, auf dem die MobileGUI ihm anzeigt, welche Dienste im Hotel verfügbar sind. Die Liste wird vom ServiceManager auf dem Gateway im Hotel bereitgestellt. Sie enthält jedoch nicht den `Weckdienst`, den Bob verwenden möchte. Auf dem Handheld befinden sich jedoch Dienst-Bundles, unter denen sich auch das Bundle für den `Weckdienst` befindet. Ferner existieren im Hotelzimmer Geräte wie etwa Heizungen, Lautsprecher, Lampen und Jalousien, die von dem `Weckdienst` benötigt werden (s. Abschnitt 2.3.4).

Damit ein mitgebrachter persönlicher Dienst im besuchten eHome ausgeführt werden kann, muss der SCD-Prozess im eHome für diesen Dienst angestoßen werden. In Abbildung 5.9 ist der konkrete Ablauf dargestellt. Hat der Benutzer den `Weckdienst` über die MobileGUI zur Übertragung ins Hotel ausgewählt, greift der ServiceManager ein und sorgt für die Konfigurierung und für das Deployment des Dienstes. Hierfür benutzt er die Funktionalitäten der auf dem Hotel-Gateway vorhandenen Werkzeuge Configurator und Deployer. Dabei handelt es sich bei diesen beiden nicht um eigenständige Werkzeuge, sondern um Teilfunktionalitäten des eHomeConfigurator, der in Abschnitt 3.2.3 vorgestellt wurde. Sie werden hier jedoch bewusst voneinander getrennt erwähnt, damit die einzelnen Phasen klarer auseinander gehalten werden können.

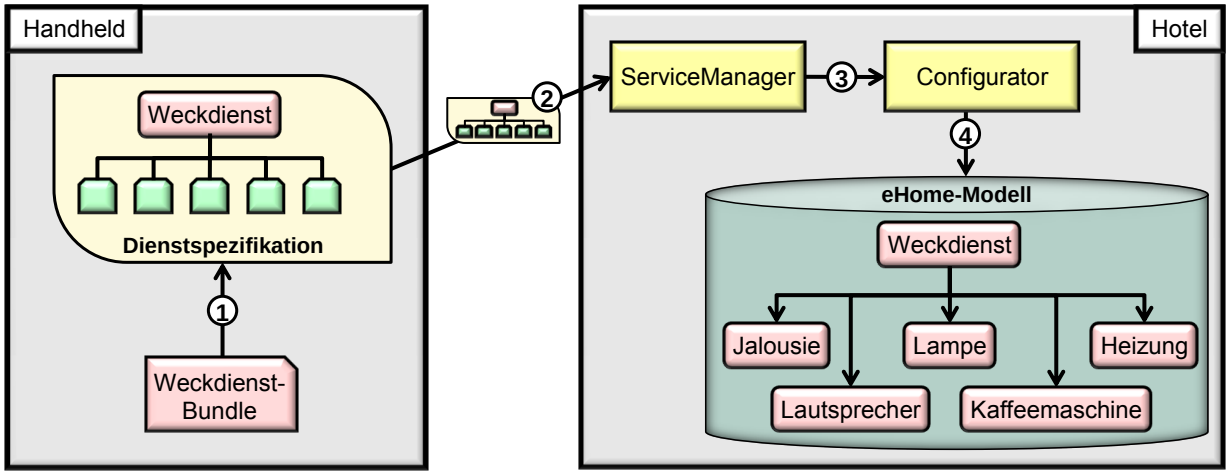


Abbildung 5.9: Konfigurierung des `Weckdienstes` für die Ausführung im eHome.

Der Ablauf sieht dabei folgendermaßen aus (s. Abbildung 5.9): Zunächst wird die Dienstspezifikation aus dem Bundle extrahiert (1) und an den ServiceManager übergeben (2). Letzterer übergibt die Spezifikation dem Configurator (3), der überprüft, ob der Dienst im aktuellen eHome, hier das Hotel, ausgeführt werden kann. Hierfür überprüft er, ob die in der Spezifikation des Weckdienstes als benötigt gekennzeichneten Funktionalitäten von passenden Unterdiensten oder Geräten im eHome realisiert werden. Konkret handelt es sich dabei um Lautsprecher, Lampen, Heizungen, Jalousien und die Kaffeemaschine. Ist diese Voraussetzung nicht erfüllt, kann der Dienst nicht konfiguriert und ausgeführt werden. Ist sie jedoch erfüllt, wird der Dienst konfiguriert (4). Das eHome-Modell aus Abbildung 5.9 zeigt das Ergebnis der Konfigurierung. Der `Weckdienst` wurde mit den Treiberdiensten der im Hotelzimmer vorhandenen Geräte verbunden und kann deren Funktionalitäten nutzen.

Das eHome-Modell dient auch als Grundlage für das anschließende Deployment, dessen Ablauf in Abbildung 5.10 dargestellt ist. Um das Deployment durchzuführen, wird der Deployer nach der Konfigurierung vom Configurator angestoßen. Daraufhin ermittelt der Deployer, welche Änderungen sich im eHome-Modell bzgl. der Dienstkomposition ergeben haben (2). Im obigen Beispiel ist der `Weckdienst` mit seinen Unterdiensten hinzugekommen. Um diese Änderungen auch auf Instanzebene nachzuziehen, benötigt der Deployer die Bundles der betroffenen Dienste. Das Besondere an dieser Situation ist, dass sich das Bundle des persönlichen `Weckdienstes` nicht wie die übrigen Bundles auf dem Hotel-Gateway befindet, sondern auf dem Handheld des Benutzers. Für das Deployment wird dieser daher zunächst vom Handheld auf das Hotel-Gateway übertragen (3). Anschließend wird aus diesem eine Instanz des `Weckdienstes` erzeugt, der mit seinen Unterdiensten über Dependency Injection verbunden (4) und anschließend gestartet wird (5). Nun ist der Benutzer in der Lage, den `Weckdienst` zusammen mit den anderen, vom Hotel angebotenen, Diensten aus Abbildung 5.8 zu verwenden.

Es muss auch berücksichtigt werden, dass der `Weckdienst` zur Kategorie der personalisierbaren Dienste gehört. Daher müssen auch die Benutzerdaten vom Handheld auf das

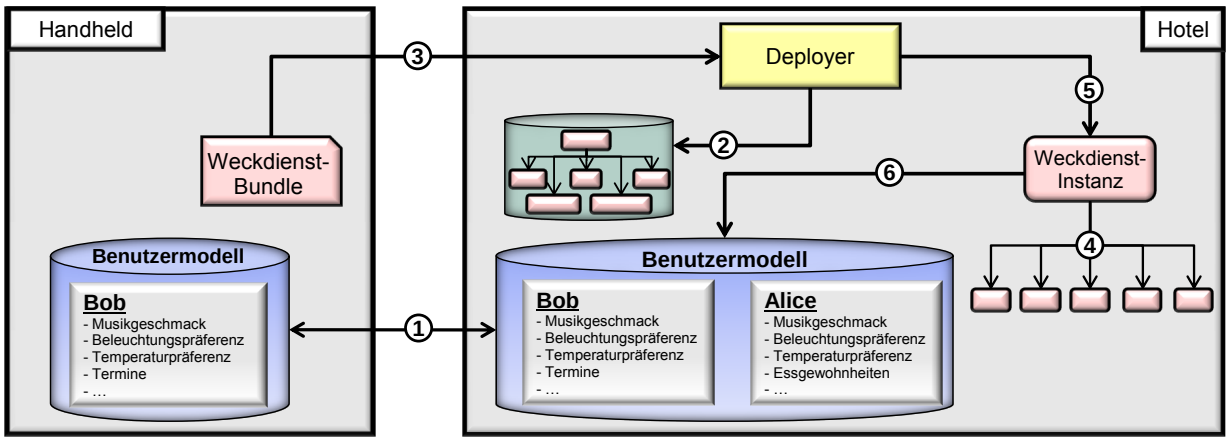


Abbildung 5.10: `Weckdienst` nach dem Deployment im eHome.

Hotel-Gateway übertragen werden. Dieser Aspekt ist auch in Abbildung 5.10 dargestellt (1). Bobs Termine und seine Präferenzen bzgl. Musik, Temperatur und Beleuchtung sind im Hotel verfügbar und können dort unter anderem auch von dem `Weckdienst` verwendet werden (6).

5.2.2.2 Ausführung persönlicher Dienste auf dem Handheld

Eine mögliche Erweiterung zur Ausführung mitgebrachter Dienste auf dem Gateway im besuchten eHome ist die *Ausführung der Dienste auf dem eigenen Handheld*. Diese Erweiterung macht Sinn, weil die Ausführung eines mitgebrachten Dienstes auf dem Gateway des besuchten eHomes nicht immer erwünscht ist. Zum einen könnte der eHome-Betreiber aus Sicherheitsgründen ablehnen, Dienste von Besuchern auf seinem Gateway auszuführen. Zum anderen trägt die Ausführung auf dem Handheld zum Schutz der Privatsphäre bei, weil so die Menge der dem eHome freigegebenen Daten reduziert werden kann, wie im nächsten Kapitel erläutert wird.

Um mobilen Benutzern eine weitere Möglichkeit zur Nutzung mitgebrachter Dienste zu bieten, wurde daher das Konzept zur Ausführung von eHome-Diensten auf Handhelds entwickelt. Der Ablauf der Konfigurierung des `Weckdienstes` ist in Abbildung 5.11 dargestellt. Er unterscheidet sich nur minimal von dem in Abbildung 5.9 dargestellten Ablauf. Auch hier wird die Spezifikation des Dienstes auf das Hotel-Gateway übertragen und dort konfiguriert. Der Unterschied liegt darin, dass der `Weckdienst` im eHome-Modell jetzt mit einer gestrichelten Linie dargestellt ist. Dadurch macht der Configurator erkenntlich, dass es sich hier nur um einen *Platzhalter* handelt und der entsprechende Dienst für die Ausführung auf dem Handheld, und nicht für das lokale Gateway, vorgesehen ist. Dabei wurde bei diesem Ansatz darauf geachtet, dass das Handheld nicht an der Konfigurierung beteiligt sein muss. Dadurch, dass die Kompositionen der mobil ausgeführten Dienste auch im eHome-Modell vorhanden sind, kann der Configurator Mehrfachbindungen und Ressourcenkonflikte mit lokalen Diensten vermeiden.

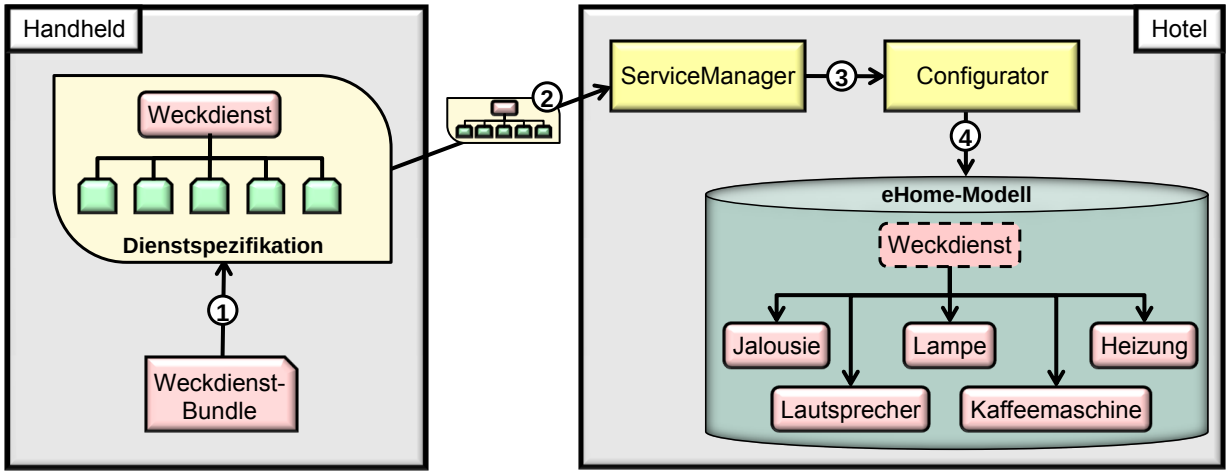


Abbildung 5.11: Konfigurierung des `Weckdienstes` für die Ausführung auf dem Handheld.

Das anschließende Deployment ist in Abbildung 5.12 dargestellt. Hier zeichnet sich eine starke Veränderung zur vorigen Variante aus. Gleich geblieben ist, dass nach der Konfigurierung der Deployer auf dem Hotel-Gateway das eHome-Modell analysiert. Falls sich dort ein Platzhalter für einen Dienst befindet, der noch nicht deployt wurde, sorgt er dafür, dass dieser Dienst auf den zugehörigen Handheld deployt wird. Hierfür wurde ein neuer Baustein auf dem Handheld, der MobileDeployer, eingeführt. Seine Aufgabe ist das Deployment von Diensten auf Handhelds. Welcher Dienst konkret deployt werden soll, ermittelt er jedoch nicht selbst, sondern bekommt die Anweisungen hierfür von dem Deployer (2), der den Zugriff auf das eHome-Modell hat. Der MobileDeployer erstellt daraufhin eine Instanz des Dienstes aus dem zugehörigen Bundle (3). Anschließend erhält der instanziierte Dienst durch Dependency Injection die Referenzen auf seine Unterdienste auf dem Hotel-Gateway (4), die zuvor von dem Deployer dort gestartet wurden. Ist der Dienst schließlich auf dem Handheld gestartet (5), kann der Benutzer dessen Funktionalitäten so in Anspruch nehmen, als würde der Dienst im Hotel laufen.

Wie weiter oben erwähnt wurde, können jene Benutzerdaten auf dem Handheld verbleiben, die von den auf dem Handheld ausgeführten Diensten benötigt werden. Um benötigte Daten zu erlangen, kann sich der Dienst an das mobile Benutzermodell auf dem Handheld wenden (6). Einerseits dient dies dem Schutz der Privatsphäre, wie in Abschnitt 6.1.1.2 näher erläutert wird. Andererseits vereinfacht dies die Dienstentwicklung, weil die Entwickler stets davon ausgehen können, dass die Benutzerdaten unabhängig vom Ausführungsort der Dienste beim lokalen Benutzermodell angefragt werden können.

Während somit die Ausführung persönlicher Dienste auf dem Handheld Vorteile bezüglich Verfügbarkeit und Privatsphäre aufweist, hat es auch einige Nachteile im Vergleich zur Ausführung der Dienste auf dem eHome-Gateway. Wenn nämlich zu viele Dienste auf dem Handheld ausgeführt werden, würde das die Ressourcen des Handhelds stark beanspruchen. Hierzu gehört die Überlastung des Prozessors genauso wie der steigende Kommunikationsaufwand zwischen Handheld und eHome-Gateway, weil die Top-Level-Dienste auf dem

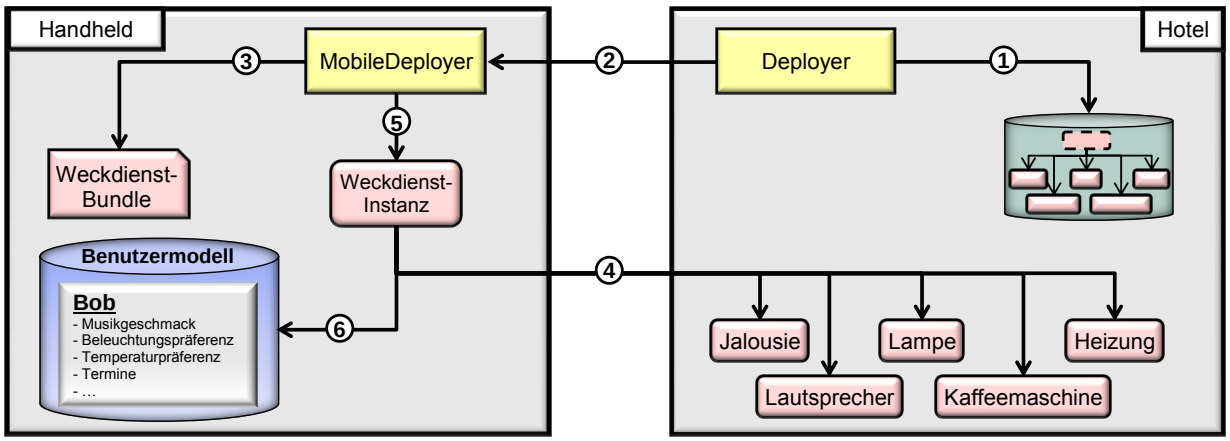


Abbildung 5.12: `Weckdienst` nach dem Deployment auf dem Handheld.

Handheld mit Unterdiensten im eHome interagieren müssen. Daher sollten möglichst viele Dienste im eHome ausgeführt werden. Nur wenn diese Möglichkeit nicht besteht oder zu sensible Daten offengelegt werden müssen, sollte die mobile Ausführung gewählt werden.

5.2.2.3 Auswirkungen auf den SCD-Prozess

In diesem Abschnitt wird erläutert, wie der SCD-Prozess angepasst werden musste, um die Personalisierung besuchter eHomes durch persönliche, vom Benutzer mitgebrachte Dienste zu ermöglichen. Dabei wird hier auf die von Retkowitz im Rahmen seiner Dissertation [Ret10] erzielten Ergebnisse, insbesondere auf die *inkrementelle* Konfigurierung, zurückgegriffen.

Zusammengefasst sieht der Ablauf dabei so aus, dass der Benutzer nach Sichtung der Liste der im eHome verfügbaren Dienste auf seinem Handheld auswählen kann, ob ein persönlicher Dienst im eHome oder auf seinem Handheld ausgeführt werden soll. Im ersten Fall kann der bestehende SCD-Prozess unverändert verwendet werden. Es muss nur dafür gesorgt werden, dass dieser Prozess vom Handheld aus angestoßen wird. Diese Aufgabe übernimmt der ServiceManager (s.Abbildung 5.9). Anders sieht es für die Ausführung von Diensten auf dem Handheld aus. Weil in diesem Fall eine Verteilungssituation entsteht, musste der SCD-Prozess so angepasst werden, dass diese Verteilung unterstützt wird. Im Folgenden werden die Anpassungen der einzelnen Phasen beschrieben.

Zur Phase der *Spezifizierung* gehört unter anderem auch die Ermittlung von Benutzeranforderungen [Ret10]. Dazu zählt auch die Entscheidung des Benutzers, einen mitgebrachten Dienst auf dem Handheld auszuführen. Damit der SCD-Prozess diese Entscheidung berücksichtigen kann, muss der Configurator wissen, um welchen Dienst es sich handelt. Hierfür wird die Dienstspezifikation verwendet, die im zugehörigen Bundle enthalten ist. Diese Spezifikation leitet der ServiceManager an den Configurator weiter, der die Vermittlerrolle einnimmt (s.Abbildung 5.11).

Anschließend kann der Configurator den Dienst für die Ausführung auf dem Handheld *konfigurieren*. Dabei wird die Konfigurierung inkrementell durchgeführt, weil sich das eHome

bereits in Betrieb befindet. Damit die auf dem Handheld ausgeführten Dienste gesondert gekennzeichnet werden können, wurde die in Abschnitt 3.2.2.4 eingeführte Klasse Service um eine boolesche Instanzvariable `remote` erweitert. In dieser wird festgehalten, ob der zugehörige Dienst auf einem Handheld oder auf dem Gateway im eHome ausgeführt werden soll. Der Ablauf der Konfigurierung für entfernt auszuführende Dienste bleibt ansonsten unverändert.

Das *Deployment* kann in fünf einzelne Schritte aufgeteilt werden, die für das Starten und Instanziieren von eHome-Diensten nötig sind. Im Einzelnen handelt es sich dabei um folgende Schritte, die der Deployer Bottom-up auf die im eHome-Modell enthaltene Dienstkomposition anwendet:

1. Installieren des zugehörigen OSGi-Bundles
2. Starten des OSGi-Bundles
3. Instanziieren des Dienstes
4. Bindung der Unterdienste an die Dienstinstanz (Dependency Injection)
5. Starten der Dienstinstanz

Durch die Bottom-up-Durchführung dieser Schritte wird sichergestellt, dass ein Dienst nicht gestartet wird, bevor alle benötigten Unterdienste gestartet worden sind. Dadurch werden die OSGi-spezifischen Anforderungen erfüllt, sodass alle Unterdienste verfügbar sind, wenn ein Dienst auf diese zugreift.

Es wird nun erläutert, inwiefern diese fünf Schritte erweitert bzw. angepasst wurden, um einen Dienst auch auf einem Handheld installieren und starten zu können. Dabei fällt zunächst auf, dass sich die genannten fünf Schritte logisch in zwei spezifische Teile gliedern lassen, die am Beispiel des `Weckdienstes` erläutert werden:

1. **Plattformspezifischer Teil**: In den Schritten 1 bis 3 werden ausschließlich Funktionalitäten der OSGi-Plattform genutzt, indem ein vorhandenes OSGi-Bundle installiert und gestartet und anschließend aus diesem die Dienstinstanz erzeugt wird. In der neuen Situation soll das Bundle des mitgebrachten Weckdienstes vom Deployer entfernt auf dem Handheld installiert und gestartet werden. Der weiter oben eingeführte MobileDeployer unterstützt den Deployer bei der Ausführung dieser drei Schritte, wie in Abbildung 5.12 dargestellt ist.

2. **Dienstspezifischer Teil**: In den Schritten 4 und 5 werden ausschließlich Funktionalitäten der Dienstinstanz genutzt. Zunächst werden der aktuell betrachteten Dienstinstanz per Dependency Injection die Referenzen auf seine Unterdienste übergeben. Anschließend wird sie durch Aufrufen der Methode `start()` gestartet. Im obigen Beispiel soll der Deployer der Instanz des `Weckdienstes` auf dem Handheld die Referenzen auf Instanzen der Unterdienste `Jalousie`, `Lautsprecher`, `Lampe`, `Kaffeemaschine` und `Heizung` übergeben und diese Instanz anschließend starten. Anders als in den Schritten 1 bis 3 wird hierfür der MobileDeployer nicht mehr verwendet.

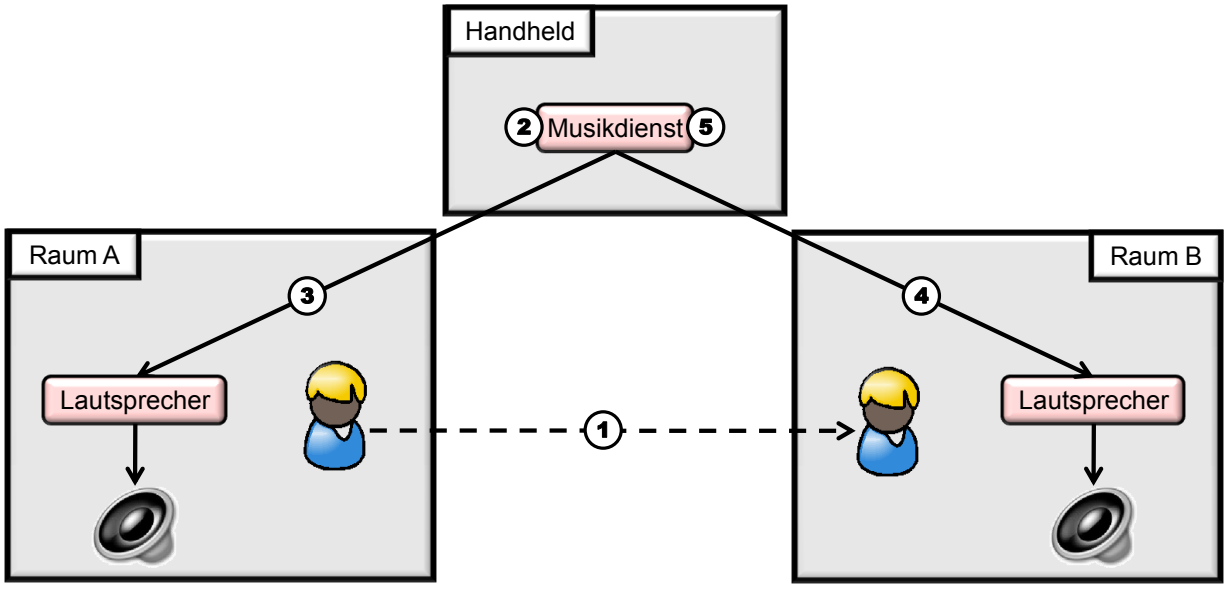


Abbildung 5.13: Umkonfigurierung des auf dem Handheld ausgeführten `Weckdienstes` im Rahmen der Intra-eHome-Mobilität.

Implementierungsdetails zu den Anpassungen am SCD-Prozess werden in Abschnitt 5.3 näher beschrieben. An dieser Stelle sei festgehalten, dass der SCD-Prozess, insbesondere das Deployment, angepasst wurde, um die neu aufgetretene Verteilungssituation zu unterstützen.

5.2.2.4 Unterstützung der Intra-eHome-Mobilität

In diesem Abschnitt wird erläutert, wie sich die Ausführung eines persönlichen Dienstes auf dem Handheld auf die Unterstützung der *Intra-eHome-Mobilität* auswirkt. In Abschnitt 4.3.2 wurden drei Typen personalisierbarer Dienste beschrieben, die bzgl. der Intra-eHome-Mobilität unterschiedlich konfiguriert werden können. Im Falle eines auf einem Handheld ausgeführten Dienstes handelt es sich um einen personengebundenen Dienst. Es wird also genau eine Instanz dieses Dienstes erstellt und an die Bezugsperson gebunden, bei dem es sich hier um den Besitzer des Handhelds handelt, auf dem der Dienst ausgeführt wird. Die Bindung wird im eHome-Modell über eine Beziehung zwischen dem Platzhalterobjekt für den mobil ausgeführten Dienst und dem Person-Objekt dargestellt, der den Benutzer repräsentiert.

Die konkrete Vorgehensweise zur Umkonfigurierung eines auf dem eHome-Gateway ausgeführten personengebundenen Dienstes im Rahmen der Intra-eHome-Mobilität wurde in Abschnitt 4.3.3 detailliert beschrieben. Der bisherige Ansatz ist jedoch nicht für die neue Verteilungssituation anwendbar, weil hierfür einige Besonderheiten berücksichtigt werden müssen. Wird beispielsweise der personengebundene `Musikdienst` auf dem Handheld ausgeführt, ergibt sich die in Abbildung 5.13 dargestellte Situation. Anfänglich ist der Dienst mit dem `Lautsprecher` im `Raum A` verbunden, in dem sich auch die Bezugsperson aufhält. Bewegt sich diese Person nun in `Raum B` (1), wird der `Musikdienst` zunächst pausiert

(2). Anschließend wird seine Bindung zum `Lautsprecher` in `Raum A` gelöst (3) und eine neue Bindung zum `Lautsprecher` in `Raum B` aufgebaut, um die Musik im neuen Raum abzuspielen. Schließlich wird der `Musikdienst` fortgesetzt.

Auf den ersten Blick unterscheidet sich diese Vorgehensweise nicht von der, die für personengebundene Dienste, die auf dem eHome-Gateway ausgeführt werden, beschrieben wurde. Die Unterschiede stecken im Detail der Realisierung. Um die Umkonfigurierung eines auf dem Handheld ausgeführten Dienstes realisieren zu können, mussten hier nämlich sowohl das Unterbrechen als auch das Fortsetzen der Ausführung an die Verteilungssituation angepasst werden. Das gilt auch für das Binden neuer Unterdienste über Dependency Injection. Auf die Einzelheiten wird in Abschnitt 5.3.6.2 näher eingegangen.

5.3 Realisierung

Die im eHome-Projekt eingesetzte Dienstplattform OSGi bietet lediglich die Möglichkeit, über einen integrierten Webserver HTTP-Schnittstellen für eHome-Dienste zu implementieren, die dann über einen Browser aufgerufen werden können. Dies ermöglicht aber keine bidirektionale Kommunikation, die zur Realisierung der in diesem Kapitel eingeführten Konzepte benötigt wird.

Daher wurde der eHome-Prototyp im Rahmen dieser Arbeit so erweitert, dass eine *verteilte*, *bidirektionale* Kommunikation zwischen Handhelds und eHomes unterstützt wird. Für das Auffinden von eHomes wurde ein spezieller *Discovery*-Mechanismus entwickelt, das auf dem Peer-to-Peer-Protokoll (P2P) *JXTA* basiert. Basierend auf diesem Protokoll wurde auch ein eigener Ansatz für entfernte Methodenaufrufe (engl. *Remote Method Invocation (RMI)*) entwickelt. Im Folgenden werden die Einzelheiten dieser Ansätze näher erläutert.

5.3.1 Grundlegende Eigenschaften von JXTA

Als grundlegendes Protokoll für die Kommunikation zwischen dem Handheld und dem eHome-Gateway wurde das P2P-Protokoll *JXTA* ausgewählt. Das JXTA-Projekt [BGKS02, Gon01b, OG02] wurde 2001 von Sun Microsystems ins Leben gerufen und hat als Ziel, P2P-Anwendungen durch frei zugängliche Protokolle zu standardisieren. Den Kern des Projekts bilden die JXTA-Protokolle, die auf XML basieren und daher unabhängig von Programmiersprachen, Betriebssystemen und Transportprotokollen (z. B. TCP/IP oder Bluetooth) spezifiziert werden können. Es existieren Referenzimplementationen von JXTA u. a. für die Sprachen C/C++, Java und J2ME. Abbildung 5.14 zeigt die Architektur von JXTA. Zu den zentralen Begriffen der JXTA-Spezifikation gehören:

- **Peer**: Jeder Teilnehmer im P2P-Netzwerk, der über JXTA kommuniziert, wird als *Peer* bezeichnet. Dies kann ein Computer sein, aber auch ein PDA oder Handy. Jeder Peer besitzt ferner eine eindeutige ID.

- **Peer Group**: Damit zwei Peers miteinander kommunizieren können, müssen sie sich in derselben *Peer Group* befinden. Diese gruppiert eine Menge von Peers, die im

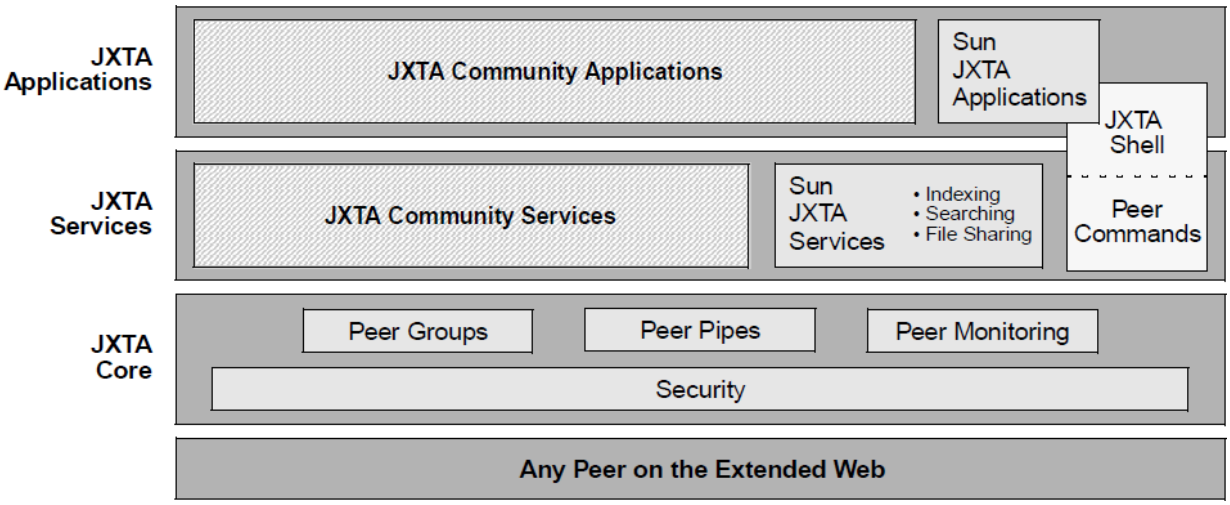


Abbildung 5.14: Die JXTA-Architektur (Quelle: [Gon01c]).

Allgemeinen eine gemeinsame Arbeit verrichten sollen. Die *Net Peer Group* ist die Standardgruppe, in der jeder Peer Mitglied ist und die dazu benutzt werden kann, andere Peers oder Gruppen zu finden. Abbildung 5.15 zeigt, wie unterschiedliche Peers in Gruppen organisiert sein können und wie sich die Topologie eines JXTA-Netzes zusammensetzt.

- **Message**: Peers derselben Peer Group können sich untereinander Nachrichten (*Message*) verschicken.

- **Pipe**: Als *Pipe* wird der Kommunikationskanal zwischen Peers oder Peer Groups bezeichnet, über den Nachrichten verschickt werden können.

- **Advertisement**: Um Ressourcen in einem Netzwerk bekannt zu machen, definiert JXTA sogenannte *Advertisements*. Dabei handelt es sich um XML-Dokumente, die eine Bezeichnung und eine Beschreibung der Ressource enthalten sowie ein Ablaufdatum. Mit solchen Advertisements können neben Peers, Peer Groups und Pipes auch *Services* bekannt gemacht werden.

- **Service**: Als *Service* wird ein Dienst bezeichnet, der von einem Peer oder einer Peer Group zur Verfügung gestellt werden kann.

JXTA bietet außerdem einen *Discovery*-Mechanismus, der auf die Dynamik eines P2P-Netzwerkes abzielt. Mit diesem Mechanismus ist es möglich, neu hinzugekommene Peers, Peer Groups oder auch Pipes über Advertisements zu finden. Ferner sind in JXTA verschiedene *Sicherheitsmechanismen* realisiert. So kann beispielsweise die Kommunikation über eine Pipe verschlüsselt oder das Betreten einer Peer Group erst nach erfolgreicher Authentifizierung erlaubt werden.

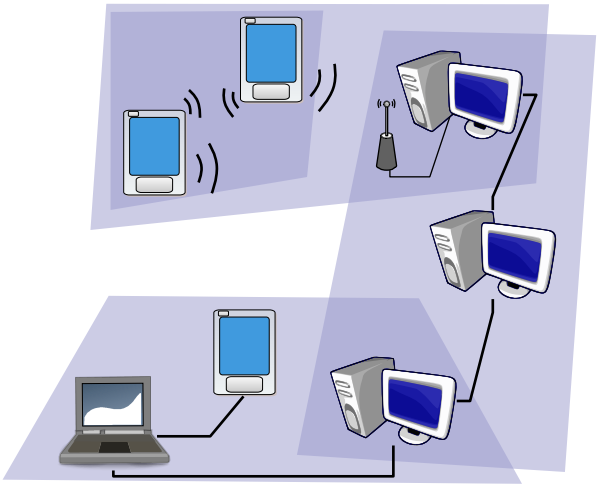


Abbildung 5.15: JXTA verbindet verschiedene Arten von Peers über eine Vielfalt von Netzwerken zu Peer Groups.

Die Entscheidung für JXTA lässt sich durch seine nützlichen Eigenschaften wie die Unabhängigkeit von Programmiersprachen und Betriebssystemen, frei verfügbare Referenzimplementierungen auch für ressourcenbeschränkte Geräte wie PDAs oder Mobiltelefone (*JXME*, s. [jxta]) und die Möglichkeit begründen, dass sich Peers mithilfe des Discovery-Mechanismus ohne manuellen Eingriff von Benutzern *ad-hoc* zusammenschließen und miteinander kommunizieren können.

Als Netzwerk für die Kommunikation wird *Wireless LAN* (WLAN) verwendet, da dies eine ebenso weit verbreitete wie verfügbare Netzwerktechnik ist. Durch JXTA wird jedoch prinzipiell vom konkreten Netzwerk abstrahiert, sodass es ohne großen Aufwand möglich ist, in Zukunft andere Netzwerktechniken einzusetzen. Drahtlose Netzwerke wie WLAN haben jedoch den Vorteil, dass der Benutzer sich frei bewegen kann und keine feste Verbindung, z.B. durch Einstecken eines Kabels, notwendig ist. Ebenso ist es möglich, dass das Handheld autonom nach kabellosen Netzwerken sucht, die sich in Reichweite befinden, und sich mit diesen verbindet.

5.3.2 Neu entwickelte Komponenten

Um eine Kommunikation zwischen dem Handheld und dem eHome-Gateway auf Basis von JXTA zu ermöglichen und mobile Benutzer bei der Personalisierung von eHome-Diensten zu unterstützen, wurden im Rahmen dieser Arbeit einige neue Komponenten entwickelt, die im Folgenden beschrieben werden.

5.3.2.1 Neu entwickelte Komponenten für das eHome-Gateway

Abbildung 5.16 zeigt eine grobe Einteilung der Komponenten, die auf einem eHome-Gateway oder auf einem Handheld ausgeführt werden. Auf der rechten Seite der Abbildung sind die Komponenten des eHome-Gateways abgebildet. Zunächst ist dargestellt, dass in

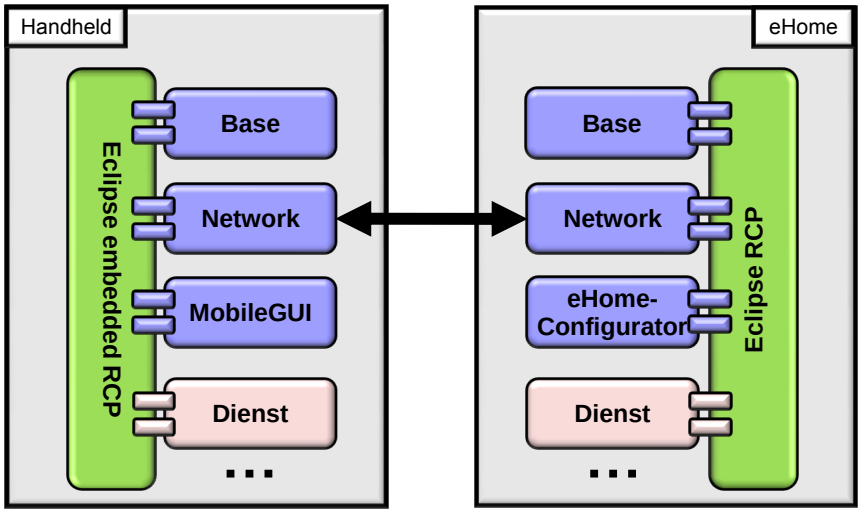


Abbildung 5.16: Neue entwickelte Komponenten für Handheld und eHome-Gateway.

eHomes als Dienstplattform das in Abschnitt 2.1.4.4 beschriebene und auf OSGi basierende *Eclipse RCP* eingesetzt wird. An diese Plattform sind neben den eHome-Diensten die Komponenten Base, Network und eHomeConfigurator gekoppelt. Die Komponenten Base und eHomeConfigurator existierten schon zu Beginn dieser Arbeit. Dabei enthält Base alle benötigten Basisfunktionalitäten für den SCD-Prozess und das eHome-Modell. Die Komponente eHomeConfigurator enthält hingegen die Benutzeroberfläche des gleichnamigen Werkzeugs.

Neu ist die Komponente Network, die für die Kommunikation zwischen eHomes und Handhelds zuständig ist. Sie realisiert den Discovery-Mechanismus auf Basis von JXTA und bietet verschiedene Handler für die Verwaltung von Peers und Peer Groups an. Auch der Aufbau von Pipes und das Erzeugen und Senden von Nachrichten wird in dieser Komponente realisiert. Dass die Kommunikation mit Handhelds über diese Komponente stattfindet, ist in Abbildung 5.16 durch den Pfeil angedeutet. Ferner muss sich das Handheld an diese Komponente wenden, um sich Zugriff auf die im Verlauf dieses Kapitels vorgestellten Bausteine InformationService, ServiceManager und UMSynchronizer zu verschaffen.

5.3.2.2 Neu entwickelte Komponenten für das Handheld

Für die Implementierung des Prototyps für Handhelds wurde ebenfalls Java als Programmiersprache gewählt. Weil PDAs, Mobiltelefone und ähnliche Geräte nur über beschränkte Ressourcen und damit eine limitierte Leistung verfügen, ist es nicht möglich, eine vollwertige Java Virtual Machine auf diesen Geräten auszuführen. Aus diesem Grund existiert die sogenannte *Java Micro Edition* [Sun], eine Umsetzung von Java für mobile Geräte, die im Vergleich zur sogenannten *Standard Edition* weniger Klassen und Pakete der Klassenbibliothek anbietet. Für die Implementierung des Prototyps für Handhelds wird eine Implementierung von Java ME der Firma IBM verwendet (s. [IBM10]), die zurzeit die am weitesten entwickelte und stabilste Laufzeitumgebung ist.

Weil auf den Handhelds auch eHome-Dienste ausführbar sein sollen, wurde als Dienstplattform für Handhelds auch OSGi eingesetzt. Die Entscheidung fiel auf die *Eclipse embedded Rich Client Platform* (eRCP) [Ecl08a], da sie, wie Eclipse RCP, eine vollständige Implementierung des OSGi-Standards in Version 4 enthält und auf ressourcenbeschränkte Geräte zugeschnitten ist. Dies ist auf der linken Seite von Abbildung 5.16 dargestellt.

Zusätzlich sind in Abbildung 5.16 drei weitere Komponenten dargestellt, die auf Handhelds vorhanden sein müssen, damit mobile Benutzer alle Funktionalitäten nutzen können, die ihnen geboten werden. Hierzu gehört zunächst die Komponente Base, die für allgemein verwendbare Funktionen vorgesehen ist. In dieser Komponente werden unter anderem Benutzerdaten abgelegt. Die Komponente MobileGUI realisiert die grafische Benutzeroberfläche. Diese wurde auf Basis von *eSWT* entwickelt, einer eingebetteten Variante des *Standard Widget Toolkits* [Ecl08b], die die Entwicklung grafischer Oberflächen mit besonderer Unterstützung für mobile Geräte ermöglicht.

Die Komponente *Network* auf dem Handheld enthält alle JXTA-bezogenen Funktionen und kapselt diese, um Implementierungsdetails vor anderen Komponenten zu verbergen. Nach ihrem Start wird der Discovery-Mechanismus in Gang gesetzt, um automatisch in regelmäßigen Intervallen nach neuen Peers, Peer Groups und anderen Objekten zu suchen. Befindet sich ein eHome-Gateway in Reichweite, findet dieser Mechanismus die Peer-Gruppe „eHome group“, die automatisch betreten und ebenfalls regelmäßig durchsucht wird. Sobald diese Gruppe gefunden wurde, wird die MobileGUI über ein Ereignis benachrichtigt. Diese wiederum informiert den Benutzer, dass ein eHome in der Nähe verfügbar ist, und zeigt ihm an, welche Aktionen er ausführen kann. Ferner muss sich der Deployer auf dem eHome-Gateway an diese Komponente wenden, um Zugriff auf den MobileDeployer zu erlangen.

5.3.3 SimpleRMI: RMI auf Basis von JXTA

Wie in Abbildung 5.16 dargestellt ist, müssen die Komponenten auf dem Handheld und dem eHome-Gateway miteinander kommunizieren können, beispielsweise die MobileGUI mit dem InformationService, um konkrete Informationen über das zugehörige eHome zu erlangen. Hierfür müssen einige Komponenten die Methoden anderer Komponenten der Gegenseite *entfernt* aufrufen. Dies ist jedoch nicht ohne Weiteres möglich, weil sie sich in unterschiedlichen Adressräumen befinden. In diesem Unterabschnitt wird erläutert, wie im Rahmen dieser Arbeit das entfernte Aufrufen von Methoden auf Basis von JXTA realisiert wurde.

Zur Kommunikation zwischen zwei Peers in einem Netzwerk bietet JXTA selbst nur die Möglichkeit, Daten binär oder in Form von XML-Nachrichten auszutauschen. Eine Möglichkeit, Methoden ähnlich zu Javas *Remote Method Invocation* (RMI) [PM01] entfernt aufzurufen, existiert hier nicht. Mit anderen Worten definiert JXTA kein *verteiltes Objektmodell* [SBJ03]. Solch ein Modell ist jedoch notwendig, um die verteilte Kommunikation von Komponenten auf dem Handheld und dem eHome-Gateway zu ermöglichen. Dabei muss der Verteilungsaspekt vor den beteiligten Komponenten verborgen sein. Es soll für den Methodenaufrufenden also nicht erkennbar sein, ob ein Aufruf lokal oder entfernt ausgeführt wird.

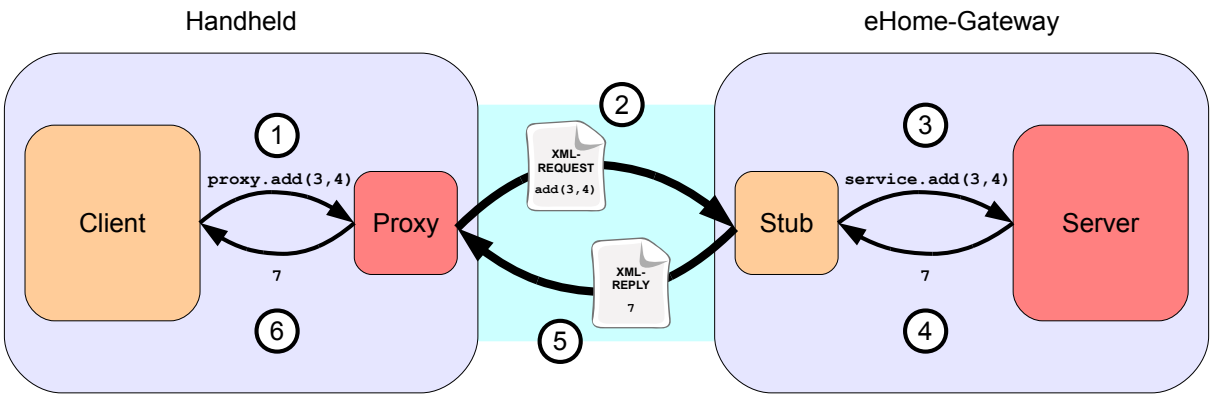


Abbildung 5.17: Funktionsweise von SimpleRMI.

Es gibt einige Projekte, die ähnliche Mechanismen wie RMI auf Basis von JXTA umsetzen, und auf den ersten Blick für den hier verfolgten Ansatz geeignet erschienen. Das Projekt *jxta-rmi* [jxtb] bietet beispielsweise eine RMI-API für JXTA, jedoch wurden die letzten Änderungen am Quellcode im Jahre 2001 vorgenommen. Die letzte verfügbare Version von jxta-rmi ist aus diesem Grund nicht mehr kompatibel mit der aktuellen Version von JXTA, sodass die Implementierung dieses RMI-Mechanismus nicht benutzbar ist. Inzwischen ist auch die Webseite dieses Projekts nicht mehr verfügbar. Ebenfalls existieren Projekte für die Implementierung des *XML-RPC*-Protokolls [All03, TA90] für JXTA [jxtd] und des *Simple Object Access Protocols* (SOAP) auf Basis von JXTA [jxtc], die jedoch ebenfalls seit mehreren Jahren keine Aktivität mehr zeigen.

Dieser Mangel an verfügbaren und funktionsfähigen RMI-ähnlichen Ansätzen führte dazu, eine eigene Variante auf Basis von JXTAs XML-Nachrichten zu entwickeln. Die Implementierung trägt den Namen *SimpleRMI*. Die Funktionsweise von SimpleRMI ist in Abbildung 5.17 dargestellt. Der hier entwickelte Ansatz, und damit auch die verwendete Terminologie, basiert auf Java *Dynamic Proxy Classes* [Mic] von Sun. Diese ermöglichen die Interaktion zweier Objekte über einen Mittler, dem Proxy, der zur Laufzeit generiert werden kann. Mit solch einem Proxy kann der Zugriff auf Objekte kontrolliert werden. Hierzu gehört unter anderem auch die Kommunikation über ein Netzwerk mit Objekten, die sich in einem anderen Adressraum befinden.

Wie in Abbildung 5.17 beispielhaft dargestellt ist, ermöglicht SimpleRMI einem Objekt Client auf einem Handheld den entfernten Aufruf einer Methode, in diesem Fall `add(3,4)`, des Objekts Server im eHome-Gateway über einen Proxy. Der Aufruf wird also nicht direkt auf dem Server durchgeführt, sondern landet zunächst beim Proxy. Dieser weiß, dass sich der Server auf dem eHome-Gateway befindet und nicht auf dem Handheld. Daher bereitet er den Methodenaufruf so vor, dass er über das Netzwerk zum eHome-Gateway transportiert wird. Die Aufbereitung basiert hier auf JXTAs XML-Nachrichten, d. h., der Aufruf wird in einen XML-Request umgewandelt und über JXTA verschickt. Dieses Verfahren, eine Menge von Daten vor dem Versand an einen Empfänger in ein transportfähiges Format umzuwandeln,

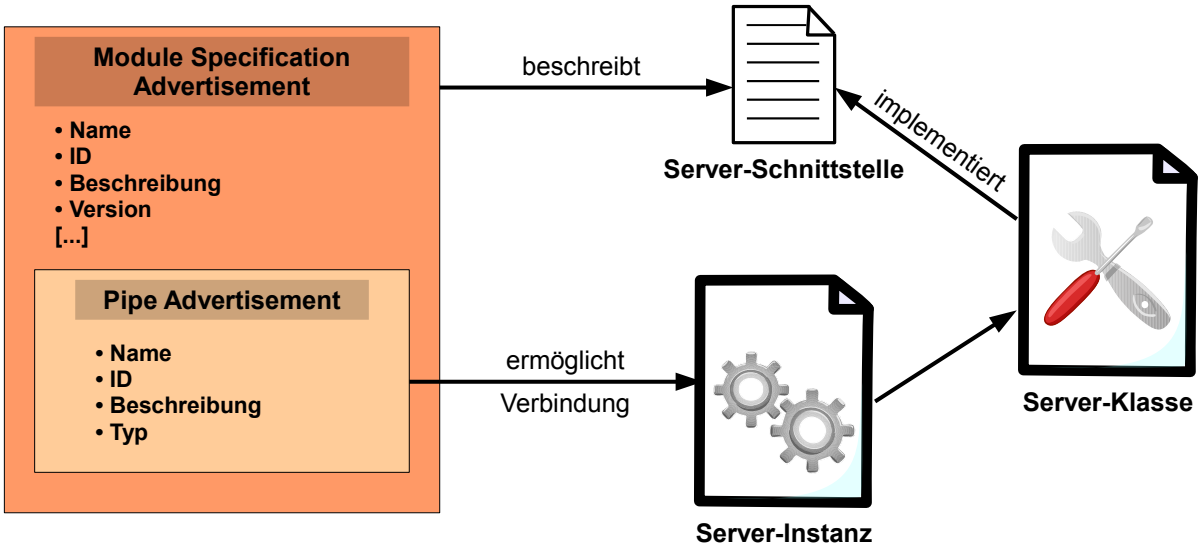


Abbildung 5.18: Advertisements werden verwendet, um RMI-Server im Netzwerk zu veröffentlichen.

ist auch unter dem Begriff *Marshalling* bekannt. Auf dem eHome-Gateway wird der XML-Request nicht direkt vom Server angenommen, sondern von einem Stub, der das Gegenstück zum Proxy darstellt. Der Stub wandelt die Nachricht wieder in seine ursprüngliche Form um (*Unmarshalling*) und ruft die entsprechende Methode des *Server*-Objekts auf. Dieser kann nun den Aufruf abarbeiten und das Ergebnis dem Stub mitteilen. Der Stub verpackt das Ergebnis selbst wiederum in ein *XML-Reply* und schickt es zurück an den Proxy, der das Ergebnis dann dem Client in einer verständlichen Form mitteilt.

Beim Entwurf von SimpleRMI wurde darauf geachtet, dass folgende Anforderungen erfüllt werden:

- **Transparenz**: SimpleRMI ermöglicht, dass OSGi-Bundles über Gateway-Grenzen hinweg miteinander kommunizieren können. Weder das Client- noch das Server-Objekt müssen sich um die Verteilung kümmern. Im Gegenteil: Sie haben stets das Gefühl, als wäre der Gegenüber auf dem gleichen Gateway.

- **Dynamik**: Damit dieser Ansatz für alle Arten von Diensten und Objekten funktioniert, ist es wichtig, dass sowohl die Stub- als auch die Proxy-Objekte *dynamisch* zur Laufzeit generiert werden können, also erst bei Bedarf. Diese Eigenschaft wird, wie oben schon erwähnt, durch die Verwendung von Java Dynamic Proxy Classes erfüllt.

- **Bidirektionalität**: Die Kommunikationsrichtung für die in den vorhergehenden Abschnitten vorgestellten Konzepte ist *bidirektional*. Es finden also nicht nur Aufrufe vom Handheld zum eHome, sondern auch in der umgekehrten Richtung statt. Daher wurde SimpleRMI so konzipiert, dass eine bidirektionale Kommunikation möglich ist.

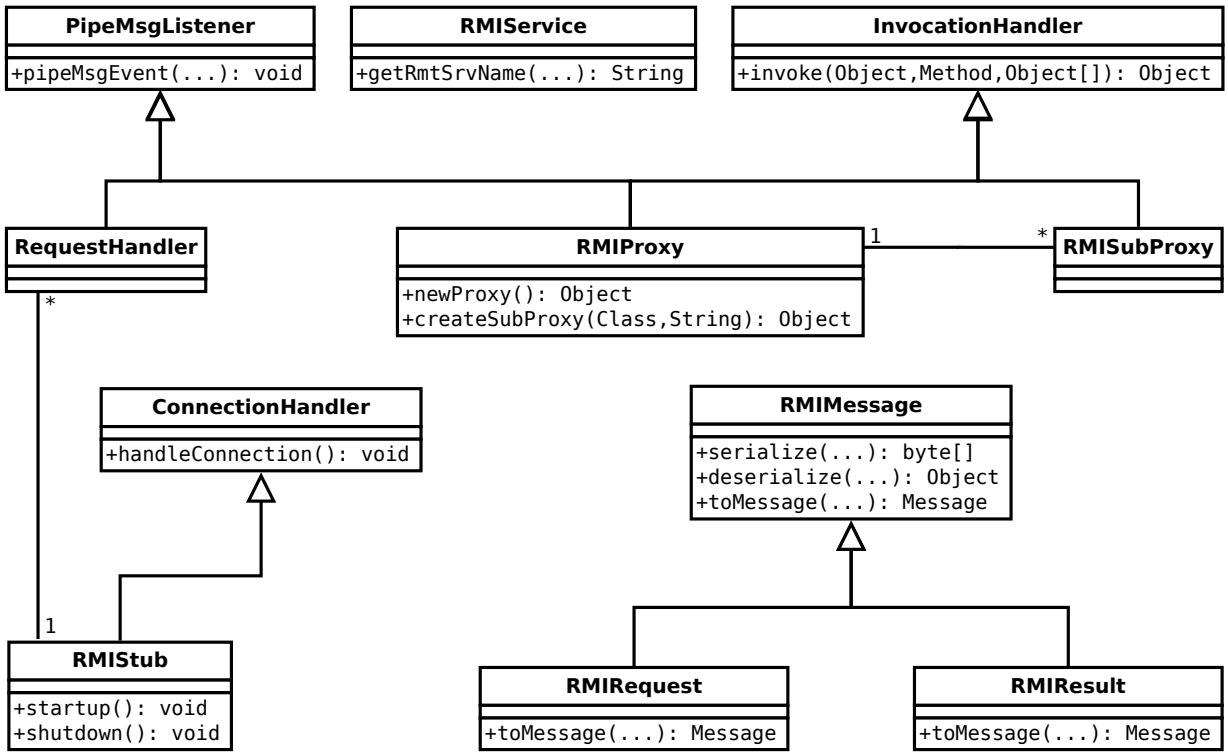


Abbildung 5.19: SimpleRMI-Klassendiagramm.

Um RMI-Server im Netzwerk zu veröffentlichen, verwendet SimpleRMI JXTA *Module Specification Advertisements* (s. Abbildung 5.18). Ein solches Advertisement beschreibt allgemein die Spezifikation eines Moduls (in diesem Falle eines RMI-Objekts) und dient u. a. dazu, dieses Objekt entfernt verfügbar zu machen. Für die Beschreibung der Kommunikationsmöglichkeiten mit dem Server wird ein *Pipe Advertisement* verwendet, das in ein Module Specification Advertisement eingebettet wird und mit dem eine Verbindung zum Server aufgebaut werden kann. Wurden beide Advertisements erzeugt, werden sie in der entsprechenden Peer Group veröffentlicht und können so mittels Discovery gefunden werden.

5.3.4 Architektur von SimpleRMI

SimpleRMI selbst ist Teil der Network-Komponenten, die in Abbildung 5.16 dargestellt sind. Die Architektur von SimpleRMI besteht aus mehreren Klassen und Schnittstellen, die im Folgenden beschrieben werden (s. Abbildung 5.19):

- **RMIStub**: Mit den oben erzeugten Advertisements aus Abbildung 5.18 kann für einen Server automatisch ein Stub erzeugt werden, der Anfragen aus dem Netzwerk entgegennimmt und als Methodenaufrufe an den Server weiterleitet. SimpleRMI bietet dafür die Klasse `RMIStub` an, die automatisch aus dem zugehörigen Pipe Advertisement eine sogenannte JXTAServerPipe erzeugt, mit der auf eingehende

Verbindungsanfragen reagiert werden kann. Der Methodenaufruf auf dem Server wird mithilfe von JAVA-Reflection durchgeführt. Das Resultat des Methodenaufrufs wird über die gleiche Pipe zurückgeschickt (s. auch Abbildung 5.17). Eine RMIStub-Instanz kann zu einem beliebigen Server erzeugt werden, der sich im selben Adressraum befindet. Da der Stub ständig auf neue Verbindungsanfragen wartet, wird er in einem eigenen Thread ausgeführt. Daher bietet die Klasse `RMIStub` die Methoden `startup()` und `shutdown()`, um den Stub starten bzw. beenden zu können.

- **RequestHandler**: Versucht ein Proxy eine Verbindung zu einem Stub aufzubauen, wird für diese Verbindung eine neue bidirektionale Pipe erzeugt, sodass der Stub für jeden Proxy eine *eigene* Verbindung verwaltet. Für diese Verbindung wird dann der sogenannte `RMIRequestHandler` gestartet, der die Anfragen des Proxies über die zugehörige Pipe entgegennimmt, in Methodenaufrufe umsetzt und das Resultat oder eine mögliche Ausnahme zurückliefert. Somit ist es möglich, dass mehrere Clients einen Server gleichzeitig verwenden. Erforderliche Synchronisierungsmaßnahmen müssen dann jedoch in der Implementierung des Servers getroffen werden.

- **RMIProxy**: Zum Erzeugen eines Proxies für einen entfernten Dienst bietet die Klasse RMIProxy die statische Methode `newProxy()` an, die Javas Dynamic-Proxy-Mechanismus verwendet. Dadurch kann zur Laufzeit ein Objekt erzeugt werden, das eine bestimmte Schnittstelle implementiert – in diesem Fall die Schnittstelle des entfernten Servers. Das so erzeugte Objekt kann lokal wie der entfernte Server verwendet werden, wobei die Verteilungssituation vor dem Client verborgen wird. Bei der Erzeugung des Proxies wird durch das Pipe Advertisement, das im Module Specification Advertisement enthalten ist, eine Verbindung zum Stub aufgebaut, über die Nachrichten verschickt werden können. Die Verbindung zum Stub bleibt während der ganzen Lebenszeit des Proxies bestehen, muss also nicht für jeden Methodenaufruf neu aufgebaut werden. Wird ein Methodenaufruf auf dem Proxy ausgeführt, wandelt er diesen Aufruf in eine XML-Nachricht um, die den Namen der Methode und die serialisierten Parameter enthält, und leitet diese über eine Pipe dem entsprechenden RMIStub auf der anderen Seite weiter. Nachdem der Server die eigentliche Methode ausgeführt hat, wandelt der Stub das Ergebnis des Aufrufs oder eine auftretende Ausnahme ebenfalls in eine XML-Nachricht um, die an den Proxy zurückgeschickt wird. Dieser gibt das Ergebnis an den Aufrufer zurück bzw. löst eine entsprechende Ausnahme aus, sodass der Methodenaufruf transparent delegiert wurde.

- **RMISubProxy**: Wenn jeder entfernt aufrufbare Dienst als eigener JXTA-Peer veröffentlicht würde, müssten jeweils auch ein RMIProxy und eine Pipe für die Kommunikation zwischen Proxy und Stub aufgebaut werden. Bei einer großen Anzahl an entfernt aufrufbaren Diensten steigt somit also auch die Anzahl der Pipes. Diese Tatsache wirkt sich negativ auf die Performanz aus, insbesondere weil die Handhelds ressourcenbeschränkt sind. Deshalb wurde die Klasse RMISubProxy eingeführt, um die Anzahl an Peers und Pipes möglichst gering zu halten. Dabei lassen sich zu jedem RMIProxy beliebig viele Instanzen von RMISubProxy erzeugen. Sie sind damit

zwar keine eigenständige Peers im JXTA-Kontext, können dennoch auf Client-Seite als vollwertige Stellvertreter-Objekte für den entfernt aufzurufenden Server agieren. Hierfür verwenden mehrere RMISubProxy-Instanzen dieselbe Pipe wie die zugehörige RMIProxy-Instanz.

- **RMIService**: Das Interface RMIService bildet die Grundlage für alle Objekte, die nach dem Prinzip der Remote Method Invocation über das Netzwerk entfernt aufrufbar sein sollen. Mit anderen Worten muss jede Klasse, für deren Instanzen RMIStub- und RMIProxy-Instanzen erzeugt werden sollen, die Schnittstelle RMIService implementieren. Hierzu gehören sowohl übliche eHome-Dienste als auch der InformationService, der ServiceManager, der UMSynchronizer und der MobileDeployer.

- **RMIMessage**: Die abstrakte Klasse RMIMessage bildet die Grundlage für verschiedene Typen von Nachrichten, die über eine Pipe verschickt werden können. Hierfür bietet sie Methoden zum Serialisieren und Deserialisieren von Nachrichten an, sodass diese zunächst in ein XML-Dokument umgewandelt (*Marshalling*), über eine Pipe verschickt und anschließend wieder beim Empfänger entpackt (*Unmarshalling*) werden können.

- **RMIRequest**: Die Klasse RMIRequest ist eine Spezialisierung von RMIMessage, die zum Verpacken entfernter *Methodenaufrufe* verwendet wird. Zu diesem Zweck implementiert RMIRequest die abstrakte Methode `RMIMessage.toMessage()`: Ein RMIRequest besteht aus dem Namen der entfernt aufzurufenden Methode und deren Parametern. Der Methodenname und die Parameter werden in der toMessage()-Methode serialisiert und können anschließend als XML-Nachricht über eine Pipe verschickt werden.

- **RMIResult**: Die Klasse RMIResult ist eine weitere Spezialisierung von RMIMessage. Anders als ein RMIRequest besteht ein RMIResult nur aus einem Wert: dem *Rückgabewert* der entfernt aufgerufenen Methode. Hierzu implementiert auch diese Klasse die abstrakte Methode RMIMessage.toMessage(): Da ein RMIResult lediglich den Rückgabewert der entfernt aufgerufenen Methode oder eine aufgetretene Ausnahme enthält, muss auch nur dieser Wert in der toMessage()-Methode serialisiert werden. Anschließend kann das RMIResult als XML-Nachricht über die Pipe zurückgeschickt werden.

5.3.5 MobileCommunicator und eHomeCommunicator

Um die im Verlauf dieses Kapitels vorgestellten Bausteine InformationService, ServiceManager, UMSynchronizer und MobileDeployer sowie übliche eHome-Dienste entfernt zugreifbar zu machen, müsste auf den ersten Blick für jeden von ihnen ein eigener RMIProxy und RMIStub erstellt werden. Sie wären dann als eigene Peers im zugrunde liegenden JXTA-Netwerk erreichbar. Weil für jede Proxy-Stub-Verbindung eine eigene Pipe benötigt wird, sollte die Anzahl von RMIProxies und RMIStubs möglichst gering gehalten werden, um die Performanz von Handhelds nicht negativ zu beeinflussen. Stattdessen sollten RMISubProxies eingesetzt werden.

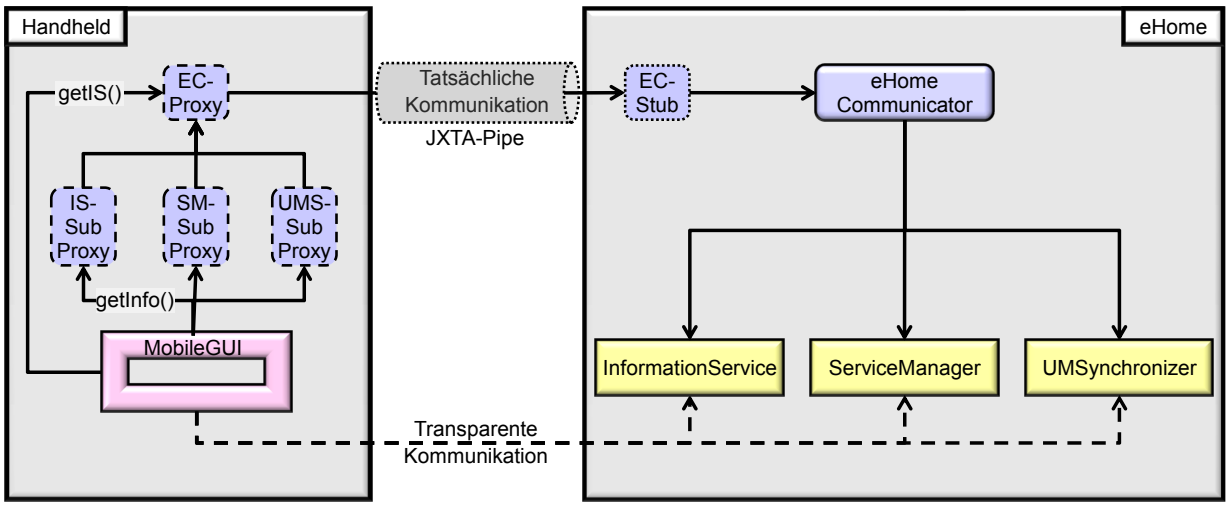


Abbildung 5.20: Der eHomeCommunicator stellt eine eigene JXTA-Pipe für Kommunikationsanfragen von Clients auf Handhelds bereit.

Aus diesem Grund wurden zwei Bausteine entwickelt, die jeweils für das eHome-Gateway oder das Handheld als Anlaufstelle für Kommunikationsanfragen dienen. Der entfernte Zugriff auf die übrigen Bausteine wird über sie abgewickelt.

Der für das eHome-Gateway entwickelte Baustein ist der in Abbildung 5.20 dargestellte eHomeCommunicator. Der eHomeCommunicator agiert als Mittler für Clients auf dem Handheld, die mit Objekten auf dem eHome-Gateway kommunizieren möchten. Beispielhaft ist als Client auf dem Handheld die MobileGUI dargestellt. Möchte die MobileGUI beispielsweise mit dem InformationService kommunizieren, sieht der Ablauf wie folgt aus: Nachdem das Handheld ein eHome erkannt hat, wird die MobileGUI benachrichtigt. Gleichzeitig wird auf dem Handheld dynamisch der EC-Proxy erstellt, der als Stellvertreter für den eHome-Communicator dient[1]. Gleichzeitig wird eine Pipe aufgebaut, die den EC-Proxy mit dem EC-Stub verbindet, der zuvor auf dem eHome-Gateway erzeugt wurde. Um mit dem InformationService zu kommunizieren, muss die MobileGUI beim eHomeCommunicator zunächst die Referenz auf den InformationService anfordern, dargestellt durch den Methodenaufruf `getIS()`. Diese Anfrage wird zunächst vom EC-Proxy entgegengenommen und als XML-Nachricht über die zugehörige Pipe an den EC-Stub weitergeleitet. Dieser führt die Anfrage auf dem eHomeCommunicator aus und sendet die Referenz auf dem umgekehrten Weg zurück. An dieser Stelle sorgt SimpleRMI dafür, dass für die Referenz, die als Rückgabewert geliefert wurde, automatisch ein RMISubProxy, in diesem Fall IS-SubProxy, erstellt wird. Die Referenz auf den IS-SubProxy wird schließlich der MobileGUI übergeben, sodass sie die gewünschten Methoden des InformationService aufrufen kann.

Obwohl der IS-SubProxy somit als vollwertiger Stellvertreter für den InformationService agiert, wird für ihn keine neue Pipe erstellt. Es wird stattdessen die Pipe verwendet, die schon

[1]Wie weiter oben beschrieben wurde, muss der eHomeCommunicator hierfür durch entsprechende Advertisements spezifiziert sein.

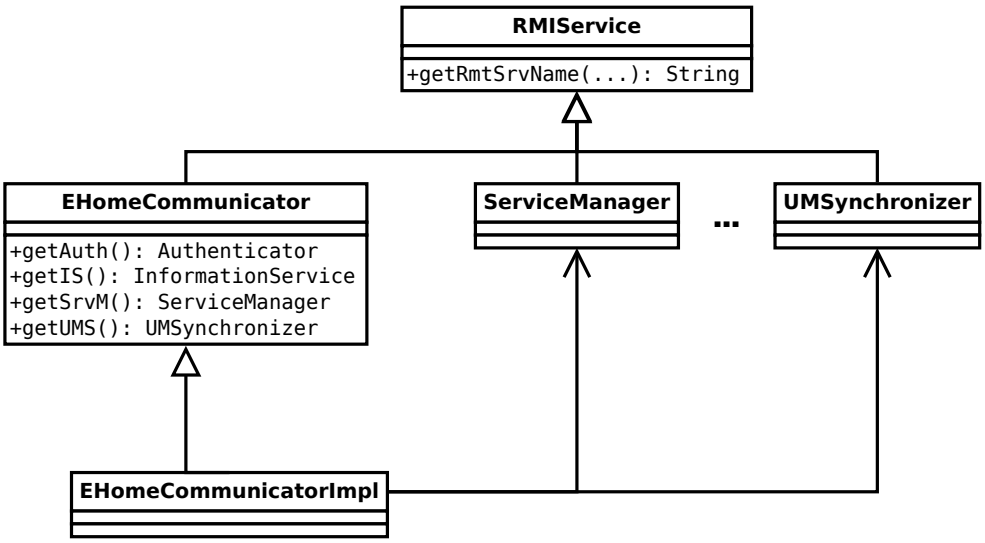


Abbildung 5.21: Klassendiagramm zum eHomeCommunicator.

zwischen dem EC-Proxy und dem EC-Stub besteht. Dies gilt auch für die übrigen Bausteine auf dem eHome-Gateway, die von Clients auf dem Handheld genutzt werden. Wie in Abbildung 5.20 dargestellt ist, werden auch für den ServiceManager und den UMSynchronizer lediglich RMISubProxies erzeugt, die keine eigene Pipe benötigen. Somit reicht eine Pipe für Anfragen aus, die von einem Handheld ausgehen.

Abbildung 5.21 zeigt das Klassendiagramm zum eHomeCommunicator. Implementiert wird die Funktionalität des eHomeCommunicator von der Klasse EHomeCommunicatorImpl. Diese Klasse besitzt auch die Referenzen zu den vom Handheld benötigten Bausteinen wie ServiceManager und UMSynchronizer. Auf Anfrage werden die Referenzen auf die Objekte dieser Bausteine zur Laufzeit an die Clients übergeben. Die dafür vorgesehenen Methoden sind auch dargestellt. Auffällig ist in Abbildung 5.21 zudem, dass sowohl der eHomeCommunicator als auch der ServiceManager usw. die Schnittstelle RMIService implementieren. Dies liegt daran, dass alle Klassen, deren Objekte zur Laufzeit entfernt aufrufbar sein sollen, RMIService implementieren müssen. Nur so ist sichergestellt, dass SimpleRMI zur Laufzeit die zugehörigen RMIStubs und RMIProxies bzw. RMISubProxies erstellt.

Für die andere Richtung wurde ein Baustein entwickelt, der auf dem Handheld als Anlaufstelle für Clients auf dem eHome-Gateway dient. Dieser Baustein wird als MobileCommunicator bezeichnet und ist in Abbildung 5.22 dargestellt. Wichtige Beispiele für Clients, die auf dem eHome-Gateway vorhanden sind und mit Objekten auf dem Handheld kommunizieren, sind der *Deployer*, der UMSynchronizer und Unterdienste auf dem eHome-Gateway, die von Diesten auf dem Handheld benutzt werden. Beispielsweise ist für den Deployer dargestellt, dass dieser den MobileCommunicator über seinen zugehörigen MC-Proxy anfragen muss, um eine Referenz auf den MobileDeployer zu erhalten. Letzterer wird auf dem eHome-Gateway durch den dynamisch erzeugten MD-SubProxy repräsentiert.

Das Klassendiagramm zur Realisierung des MobileCommunicator ist in Abbildung 5.23 dargestellt. Implementiert wird die Funktionalität des MobileCommunicator von der Klasse

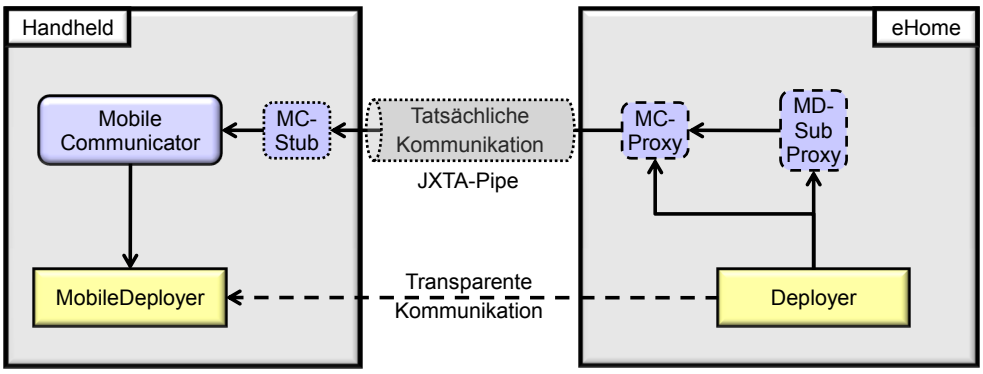


Abbildung 5.22: Der MobileCommunicator stellt eine eigene JXTA-Pipe für Kommunikationsanfragen von Clients auf dem eHome-Gateway bereit.

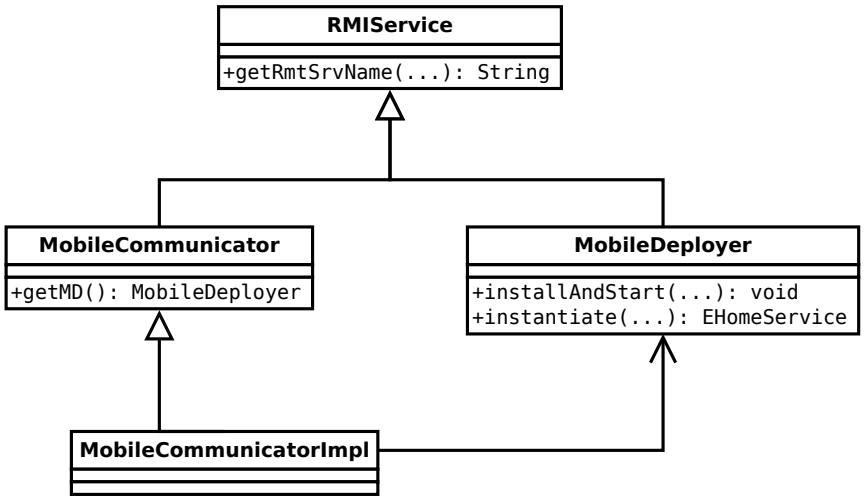


Abbildung 5.23: Klassendiagramm zum MobileCommunicator.

MobileCommunicatorImpl. Diese Klasse besitzt auch die Referenz zum MobileDeployer, dessen Methoden entfernt zum Installieren und Starten von Diensten auf dem Handheld aufgerufen werden können. Auch der MobileCommunicator und der MobileDeployer implementieren RMIService, um zur Laufzeit entfernt aufrufbar zu sein.

Listing 5.1 zeigt einen Ausschnitt aus der Implementierung zum Erzeugen und Veröffentlichen des EC-Stubs. Zunächst wird in Zeile 8 die Peer Group `eHome group` erstellt. In Zeile 12 wird diese Peer Group über das passende Advertisement im JXTA-Netzwerk veröffentlicht und kann von Handhelds gefunden werden. In Zeile 15 wird der eHomeCommunicator instanziiert. Dabei werden ihm die Referenzen der Objekte übergeben, die von einem Handheld angefragt werden können. Während der InformationService, der ServiceManager und der UMSynchronizer bereits in diesem Kapitel beschrieben wurden, wird der Authenticator im nächsten Kapitel eingeführt. Letzterer ist u. a. für die Authentifizierung der Benutzer zuständig. Damit der eHomeCommunicator über das Netzwerk angesprochen werden kann, muss auch dieser veröffentlicht werden. Dies geschieht in Zeile 21. Anschließend wird für

```
1 public class JXTAManager extends Thread
2   implements NetworkManager {
3   [...]
4   private void createGroupsAndServices() throws Exception {
5     [...]
6
7     // eHome group erstellen
8     eHomeGroup = netPeerGroup.newGroup(groupID, groupAdv,
9       EHOME_GROUP_NAME, "Join to connect to the eHome.");
10
11    // eHome group veröffentlichen
12    netPeerGroup.getDiscoveryService().remotePublish(groupAdv);
13
14    // eHomeCommunicator erstellen
15    EHomeCommunicatorImpl eHomeCommunicator =
16      new EHomeCommunicatorImpl(infoService,
17        authenticator, serviceManager,
18          umSynchronizer);
19
20    // eHomeCommunicator in eHome group veröffentlichen
21    publishService(eHomeCommunicator, eHomeGroup);
22
23    // EC-Stub erzeugen und starten
24    RMIStub eHomeCommunicatorRMIStub =
25      new RMIStub(eHomeCommunicator);
26    eHomeCommunicatorRMIStub.startup();
27    [...]
28  }
29  [...]
30 }
```

Listing 5.1: Erzeugen und Starten des EC-Stubs.

den eHomeCommunicator der EC-Stub erstellt (Zeile 24) und gestartet (Zeile 26). Bei der Erstellung des Stubs wird implizit auch das Pipe Advertisment veröffentlicht, sodass der eHomeCommunicator für Client-Anfragen bereit ist.

Listing 5.2 zeigt hingegen, wie das EC-Proxy, anhand der in Listing 5.1 veröffentlichten Informationen, auf dem Handheld erzeugt wird. Zunächst wird in Zeile 6 überprüft, ob das aktuell geprüfte Advertisement dem des eHomeCommunicator entspricht. Für diesen Zweck besitzt das Interface des eHomeCommunicator ein statisches Attribut `MODULE_SPEC_ID`, das als eindeutige ID des Module Specification Advertisements verwendet wird, um die Advertisements voneinander unterscheiden zu können. Entspricht das aktuelle Advertisement dem des eHomeCommunicator, wird mit der Methode `newProxy()` der Klasse RMIProxy das EC-Proxy erstellt (Zeile 10). Diese Methode ist auch dafür verantwortlich, eine Pipe zwischen dem EC-Proxy und dem zugehörigen EC-Stub auf dem eHome-Gateway aufzubauen. In den folgenden Zeilen werden die Referenzen der Objekte bezogen, die von dem Handheld aus benutzt werden, also InformationService, ServiceManager, Authenticator und UMSynchronizer.

```
1 public class EHomeGroupHandler implements PeerGroupHandler {
2   [...]
3   private void handleAdvResponse(
4     final DiscoveryResponseMsg response) {
5     [...]
6     if (EHomeCommunicator.MODULE_SPEC_ID
7               .equals(moduleSpecAdv.getID().toString())) {
8       [...]
9       // MC-Proxy erzeugen
10      eHomeCommunicator = (EHomeCommunicator) RMIProxy.
11            newProxy(eHomeCommunicatorClass,
12                     eHomeCommunicatorClass.getName(),
13                     moduleSpecAdv, group);
14
15      // Liefert Informationen über eHome
16      infoService =
17        eHomeCommunicator.getInformationService();
18      // Liefert u.a. Liste verfügbarer Dienste
19      serviceManager =
20        eHomeCommunicator.getServiceManager();
21      // Verwantwortlich für die Anmeldung von Benutzern
22      authenticator =
23        eHomeCommunicator.getAuthenticator();
24      // Dient der Synchronisierung von Benutzerdaten
25      umSynchronizer =
26        eHomeCommunicator.getUMSynchronizer();
27      [...]
28    }
29    [...]
30  }
31  [...]
32 }
```

Listing 5.2: Erzeugen des EC-Proxies und Beziehen von Referenzen.

Hinter diesen Referenzen stecken RMISubProxies, die eine transparente Kommunikation von Clients und den eigentlichen Objekten auf dem eHome-Gateway ermöglichen.

Die Implementierung zum Erzeugen und Veröffentlichen des MobileCommunicator auf dem Handheld und das Erzeugen der zugehörigen MC-Proxy und EC-Stub sowie des RMI-SubProxy für den MobileDeployer verläuft analog. Daher wird der zugehörige Ausschnitt der Implementierung hier nicht gezeigt.

5.3.6 Verteilte Dienstkomposition und -kommunikation

In diesem Abschnitt wird beschrieben, welche Änderungen bzw. Erweiterungen am eHome-Prototyp vorgenommen wurden, um die verteilte Dienstkomposition und -kommunikation zu ermöglichen. Zunächst wird erläutert, wie Dienstschnittstellen erweitert wurden, um sie

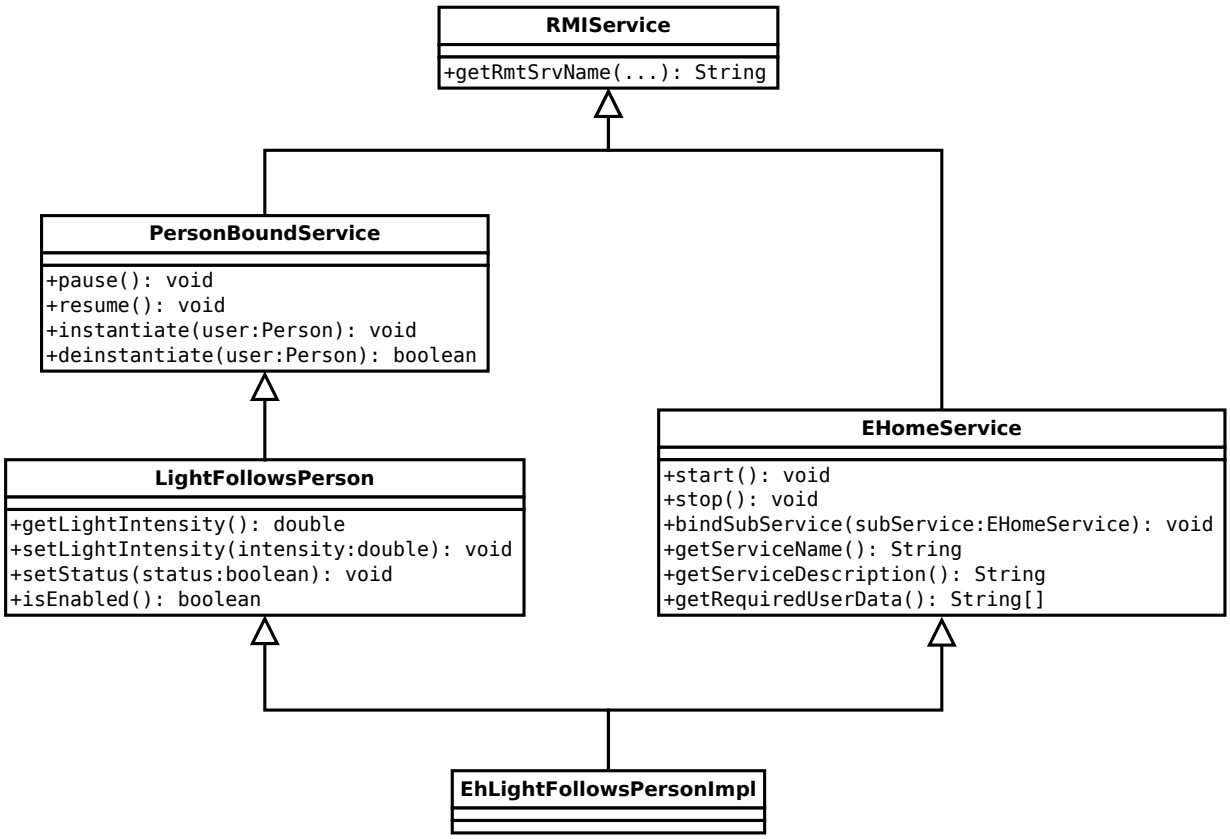


Abbildung 5.24: Klassendiagramm zum personengebundenen Lichtdienst.

sowohl für die verteilte Komposition als auch für die verteilte Kommunikation vorzubereiten. Anschließend wird beschrieben, wie die verteilte Komposition konkret umgesetzt wurde. Danach werden wichtige Aspekte der verteilten Kommunikation von Diensten beschrieben, die besonders berücksichtigt werden mussten.

5.3.6.1 Erweiterung von Dienstschnittstellen

Die im Rahmen dieser Arbeit vorgenommenen Erweiterungen an den Dienstschnittstellen werden anhand des Klassendiagramms zum personengebundenen `Lichtdienst` beschrieben, das in Abbildung 5.24 dargestellt ist.

Am unteren Ende der Abbildung ist die Klasse EhLightFollowsPersonImpl dargestellt, die für die Realisierung der konkreten Funktionalität des Dienstes zuständig ist. Dafür erweitert sie das Interface LightFollowsPerson, das von der konkreten Implementierung abstrahiert und allgemeine Methoden vorgibt, die jeder Dienst realisieren muss, der die Funktionalität `Musik-folgt-Benutzer` anbietet[2]. Zu diesen Funktionalitäten gehören bei-

[2]Diese Art der Entkopplung von Funktionalitäten und Realisierung ermöglicht einen flexiblen Austausch von konkreten Dienstimplementierungen. Im Rahmen der Inter-eHome-Mobilität wirkt sich diese Tatsache besonders günstig aus, weil in unterschiedlichen eHomes unterschiedliche Implementierungen eingesetzt werden können, wenn sie sich an die gleiche Schnittstelle halten.

spielsweise das *Aktivieren* (`setStatus(true)`) und *Deaktivieren* (`setStatus(false)`) sowie das *Setzen* (`setLightIntensity()`) der gewünschten Lichtintensität. Diese Funktionalitäten sind *dienstspezifisch* und werden nur von den Diensten realisiert, die für die Beleuchtung verantwortlich sind.

Das Interface EHomeService legt hingegen *allgemeine* Funktionalitäten fest, die jeder eHome-Dienst anbieten muss. Diese Funktionalitäten werden von der OSGi-Dienstplattform oder von Werkzeugen im Rahmen des SCD-Prozesses verwendet. Die Methoden `start()`, `stop()` und `bindSubService()` werden beispielsweise vom Deployer aufgerufen, um den Dienst zu starten, zu stoppen oder um ihm Referenzen auf seine Unterdienste zu übergeben. Diese Methoden existierten schon zu Beginn dieser Arbeit. Die übrigen Methoden `getServiceName()`, `getServiceDescription()` und `getRequiredUserData()` wurden im Rahmen dieser Arbeit hinzugefügt und werden vom ServiceManager aufgerufen, um mobilen Benutzern Informationen über die verfügbaren Dienste im eHome anzuzeigen. Insbesondere die letztgenannte Methode spielt eine besondere Rolle für den Ansatz zum Schutz der Privatsphäre, der im nächsten Kapitel beschrieben wird.

Weil es sich bei dem hier betrachteten `Lichtdienst` um einen *personengebundenen* Dienst handelt, der im Rahmen der Intra-eHome-Mobilität zur Laufzeit umkonfiguriert werden kann, muss er weitere Vorgaben erfüllen, die im Interface PersonBoundService gemacht werden. Die Methoden `pause()` und `resume()` müssen implementiert werden, um vom Configurator über eine anstehende Umkonfigurierung informiert zu werden und darauf geeignet reagieren zu können. Die Methoden `instantiate()` und `deinstantiate()` werden aufgerufen, wenn der Dienst für eine Person instanziiert wird bzw. deinstanziiert wird.

Zusammenfassend kann festgehalten werden, dass sich die Schnittstelle eines eHome-Dienstes aus verschiedenen Teilen zusammensetzt. Dazu implementiert er mehrere Interfaces, von denen jedes Vorgaben für einen anderen Aspekt macht. Eine genaue Betrachtung der oben erwähnten Aspekte macht deutlich, dass jedes Interface aus Abbildung 5.24 auch entfernt aufrufbar sein muss, um die angestrebte verteilte Dienstkomposition und -kommunikation zu realisieren. Um die gesamte Schnittstelle eines Dienstes entfernt aufrufbar zu machen, wurde erneut RMIService verwendet. Sowohl EHomeService als auch PersonBoundService erben von RMIService. Dadurch sorgt SimpleRMI dafür, dass alle Dienste, die diese Schnittstelle direkt oder indirekt implementieren, entfernt aufrufbar sind. Dies gehört zu den wichtigsten Erweiterungen der Dienstschnittstellen, die im Rahmen dieser Arbeit umgesetzt wurden.

5.3.6.2 Verteilte Dienstkomposition

In diesem Abschnitt wird erläutert, wie die verteilte Komponierung von Diensten umgesetzt wurde, die auf einem Handheld ausgeführt werden sollen. Zunächst wird beschrieben, welche Funktionalitäten der ServiceManager für die Komponierung der Dienste bereitstellt. Diese Funktionalitäten betreffen alle drei Phasen des SCD-Prozesses. Abbildung 5.25 zeigt das Klassendiagramm zum ServiceManager, der als Spezialisierung von RMIService entfernt aufrufbar ist. Realisiert wird der ServiceManager durch die Klasse ServiceManagerImpl. Folgende Methoden des ServiceManager werden vom Handheld aus aufgerufen:

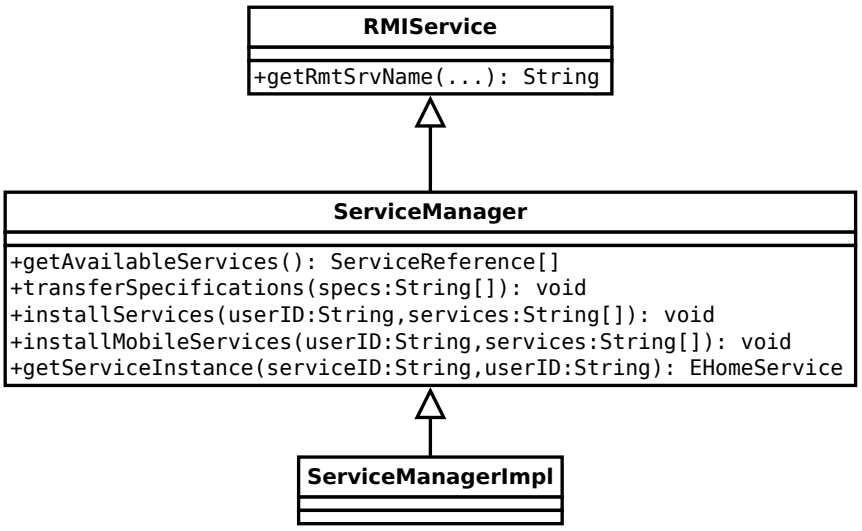


Abbildung 5.25: Klassendiagramm zum ServiceManager.

- **Abrufen der Liste verfügbarer Dienste**: Die Methode `getAvailableServices()` wird von der MobileGUI aufgerufen und liefert eine Liste der im eHome verfügbaren Dienste.

- **Übertragen von Dienstspezifikationen**: Die Methode `transferSpecifications()` ist der Spezifizierungsphase des SCD-Prozesses zuzuordnen. Falls der Benutzer mitgebrachte Dienste auf dem Handheld oder auf dem eHome-Gateway ausführen möchte, können mithilfe dieser Methode die Spezifikationen der mitgebrachten Dienste auf das eHome-Gateway übertragen werden.

- **Installieren von Diensten auf dem eHome-Gateway**: Die Methode `installServices()` ist den Phasen Konfigurierung und Deployment des SCD-Prozesses zuzuordnen und erlaubt dem Benutzer, die (inkrementelle) Konfigurierung und das anschließende Deployment der mitgebrachten Dienste für die Ausführung auf dem eHome-Gateway anzustoßen. Die Spezifikationen der Dienste müssen zuvor mithilfe der Methode `transferSpecifications()` übertragen worden sein. Als Parameter dieses Methodenaufrufs werden dabei einerseits der Benutzername (bzw. ein Pseudonym) und eine Liste von eHome-Dienst-Namen übergeben. Dadurch signalisiert der Benutzer, dass genau diese mitgebrachten Dienste für ihn zur Ausführung auf dem eHome-Gateway konfiguriert werden sollen. Der ServiceManager leitet die Anfragen entsprechend an den Configurator und an den Deployer weiter.

- **Installieren von Diensten auf dem Handheld**: Die Methode `installMobileServices()` ist ähnlich zur vorigen Methode. Sie unterscheidet sich nur darin, dass die Konfigurierung und das Deployment der Dienste für die Ausführung auf dem Handheld angestoßen werden. Da die Dienste diesmal mobil ausgeführt werden sollen, markiert der Configurator diese Dienste im eHome-Modell als Platzhalter durch das Setzen des Attributs `remote` (s. Abschnitt 5.2.2.3). Dadurch wird dem Deployer signalisiert, dass

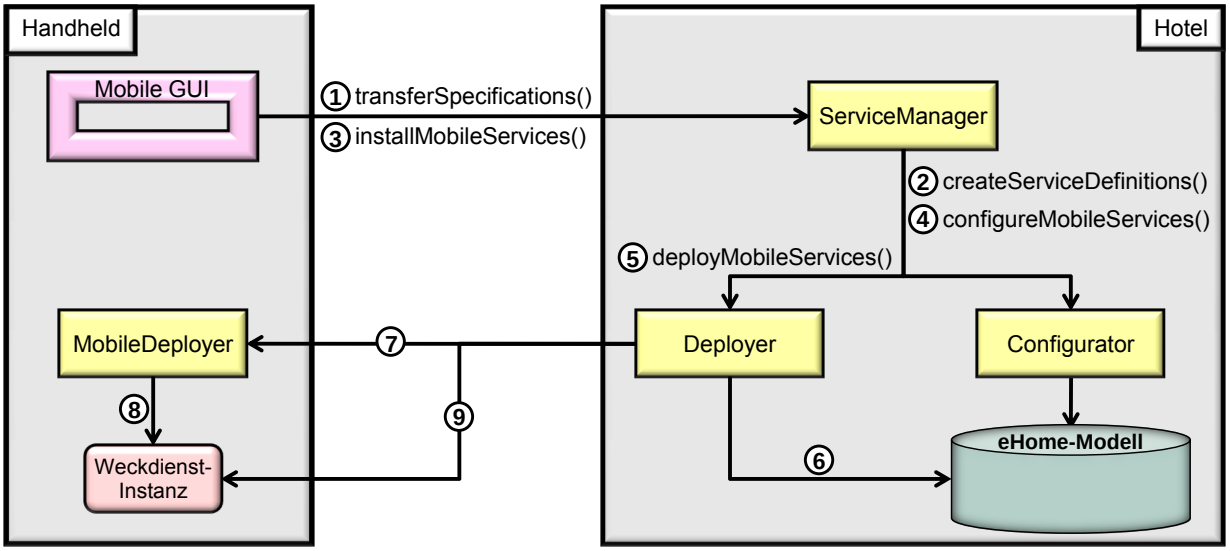


Abbildung 5.26: Beteiligte Komponenten an der Konfigurierung und dem Deployment des `Weckdienstes`.

diese Dienste im anschließenden Deployment entfernt auf dem Handheld installiert und gestartet werden müssen.

- **Anfragen einer Dienstreferenz**: Mit der Methode `getServiceInstance` kann eine Referenz auf eine im eHome ausgeführte Dienstinstanz angefragt werden. Dies ist beispielsweise zum Interagieren des Benutzers mit einem Dienst über die MobileGUI nötig.

Der ServiceManager führt den SCD-Prozess also nicht selbst durch. Stattdessen dient er als Mittler zwischen dem Handheld und den bereits existierenden Werkzeugen Configurator und Deployer auf dem eHome-Gateway. Abbildung 5.26 zeigt, wie die Ausführung der mitgebrachten Dienste auf dem Handheld mithilfe der eben erläuterten Methoden umgesetzt wird:

1. **Übertragen der Dienstspezifikationen**: Zuerst werden die Dienstspezifikationen der für die Ausführung auf dem Handheld ausgewählten Dienste von der MobileGUI an den ServiceManager übergeben.

2. **Erzeugen der Dienstspezifikationen im eHome-Modell**: Der ServiceManager leitet die Spezifikationen an den Configurator weiter. Dieser erzeugt dann im aktuellen eHome-Modell entsprechende Dienstspezifikationen, die anschließend für die Konfigurierung benutzt werden können.

3. **Entferntes Anstoßen der Konfigurierung**: Nun wird die Konfigurierung der entsprechenden Dienste von der MobileGUI angestoßen.

4. **Konfigurierung der Dienste**: Der ServiceManager leitet daraufhin die Konfigurierung der Dienste durch den Configurator ein.

5. **Deployment der konfigurierten Dienste**: Schließlich ruft der ServiceManager den Deployer auf, um das Deployment der zuvor konfigurierten Dienste anzustoßen. Letzterer ermittelt im eHome-Modell die Platzhalter der mobil auszuführende Dienste (6) und sorgt für das Deployment dieser im Zusammenspiel mit dem MobileDeployer, wie im Folgenden erläutert wird.

Wurde ein Dienst im eHome-Modell für die Ausführung auf dem Handheld markiert, interagiert der Deployer mit dem im Rahmen dieser Arbeit entwickelten MobileDeployer (7), um die Dienste auf dem Handheld zu deployen. Hierfür bietet der MobileDeployer folgende zwei Methoden an (s. auch Abbildung 5.23):

1. **installAndStart()**: Diese Methode ist für die ersten beiden Schritte zuständig, die beim Deployment für jeden Dienst durchgeführt werden. Innerhalb dieser Methode wird das OSGi-Bundle, das die entsprechende Dienstimplementation enthält, entfernt auf dem Handheld installiert und anschließend gestartet. Als Parameter werden der Methode der Name des Dienstes und der Name des Bundles übergeben. Diese Informationen werden aus der Dienstspezifikation bezogen, die zuvor auf das eHome-Gateway übertragen wurde.

2. **instantiate()**: Diese Methode ist für den dritten Schritt zuständig: Innerhalb dieser Methode wird aus dem zuvor entfernt installierten und gestarteten OSGi-Bundle der entsprechende Dienst entfernt auf dem mobilen Gerät instanziiert (8). Dazu wird als Parameter wiederum der Name des Dienstes übergeben. Als Rückgabewert liefert diese Methode eine Referenz auf die Dienstinstanz vom Typ EHomeService an den Deployer auf dem eHome-Gateway zurück.

Diese Methoden bieten also die benötigte Funktionalität für die ersten drei Schritte an, die im Deployment für jeden Dienst nun auch entfernt durchgeführt werden können. Sie entsprechen dem *plattformspezifischen Teil* des Deployments, der in Abschnitt 5.2.2.3 beschrieben wurde. Um die beiden letzten Schritte (*dienstspezifischer Teil*: Binden der Unterdienste und Starten der Dienstinstanz) entfernt durchführen zu können, greift der Deployer anschließend *nicht* mehr auf den MobileDeployer zurück, sondern ruft die entsprechenden Methoden der erzeugten Dienstinstanz auf dem Handheld direkt auf. Dies ist durch den Schritt (9) in Abbildung 5.26 angedeutet. Denn dadurch, dass die Dienstschnittstellen wie oben beschrieben erweitert wurden, ist es dem Deployer möglich, direkt auf die entfernt ausgeführte Dienstinstanz zuzugreifen. Das Binden der Unterdienste geschieht durch den Aufruf der Methode `bindSubService()` und das Starten durch den Aufruf der Methode `start()`, die jeder Dienst durch Implementierung der Schnittstelle EHomeService bereitstellt.

Zusätzlich zur erstmaligen Konfigurierung und zum erstmaligen Deployment wird auch die *Umkonfigurierung* von Diensten zur Laufzeit betrachtet. Wie die Umkonfigurierung

eines mobil ausgeführten Dienstes konzeptionell aussieht, ist in Abbildung 5.13 dargestellt. Die konkrete Umsetzung läuft folgendermaßen ab:

1. Bewegt sich die Bezugsperson in einen neuen Raum, informiert das Benutzermodell den Configurator. Dieser wiederum ruft die in PersonBoundService definierte Methode `pause()` des entfernt ausgeführten Dienstes auf. Somit wird die Ausführung der Dienstinstanz auf dem Handheld *angehalten*.

2. Anschließend werden im eHome-Modell die bestehenden Bindungen zu den Unterdiensten im alten Raum gelöst und zu passenden neuen Unterdiensten im neuen Raum aufgebaut. Damit ist die Umkonfigurierung auf *Modellebene* abgeschlossen.

3. Als Nächstes wird die Umkonfigurierung auf *Instanzebene* nachgezogen. Hierfür teilt der Deployer der unterbrochenen Dienstinstanz die Referenzen auf seine neuen Unterdienste über die in EHomeService definierte Methode `bindSubService()` mit.

4. Schließlich kann der Dienst fortgesetzt werden. Hierfür ruft der Deployer die Methode `resume` auf, die ebenfalls in PersonBoundService definiert ist.

Zusammenfassend für diesen Abschnitt kann festgehalten werden, dass der SCD-Prozess um die verteilte Komposition von persönlichen Diensten erweitert wurde. Die Realisierung der Erweiterungen wurden dabei hauptsächlich in der Netzwerkkomponente gekapselt, sodass bestehende Werkzeuge nicht zu stark von diesen Erweiterungen betroffen sind. Der Configurator muss lediglich Dienste, die auf einem Handheld ausgeführt werden sollen, über eine boolesche Variable markieren. Der Deployer hingegen muss einige Schritte des Deployments für diese markierten Dienste mithilfe des neu eingeführten MobileDeployer durchführen. Für die übrigen Schritte, die durch den Aufruf von Methoden der Dienstinstanzen durchgeführt werden können, wird der MobileDeployer nicht benötigt. Hierzu zählt auch die Umkonfigurierung der Dienste zu Laufzeit.

5.3.6.3 Entfernte Dependency Injection und Listener-Registrierung

In diesem Abschnitt werden zwei Aspekte der verteilten Dienstausführung betrachtet, die eine gesonderte Behandlung komplexer Objekte durch SimpleRMI erfordern:

1. **Entfernte Dependency Injection**: Die Bindung von Diensten an Ihre Unterdienste geschieht durch den Aufruf der Methode (`bindSubService()`). Dieser Vorgang wurde als *Dependency Injection* bezeichnet. Um einen eHome-Dienst auf einem Handheld ausführen zu können, musste sichergestellt sein, dass diese Dependency Injection auch entfernt durchgeführt werden kann. Mit anderen Worten musste der Aufruf der Methode `bindSubService()` über SimpleRMI ermöglicht werden. Dabei war zu berücksichtigen, dass diese Methode für den nicht-verteilten Fall Parameter vom Typ EHomeService erwartet, die Referenzen auf die Unterdienste darstellen. Eine besondere Anforderung hierbei war die Kapselung der Verteilungssituation durch SimpleRMI, damit sich der Deployer bei der Dependency Injection nicht um die Verteilung kümmern muss.

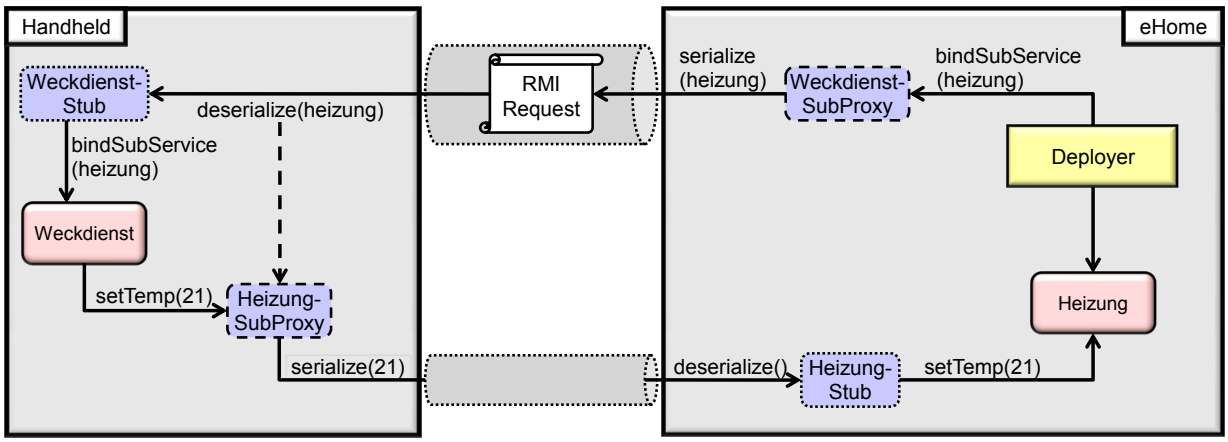


Abbildung 5.27: Entfernte Dependency Injection am Beispiel des `Weckdienstes`.

2. **Entfernte Listener-Registrierung**: Viele eHome-Dienste nutzen Sensoren, um auf Kontextänderungen im eHome zu reagieren. Zu diesem Zweck erzeugt solch ein Dienst eine ServiceListener-Instanz und registriert diese beim entsprechenden Sensortreiber. Dadurch ist es dem Unterdienst möglich, den Listener bei einer Zustandsänderung über die Methode `notify()` zu informieren. Dieses Konzept ist auch als *Observer Pattern* oder *Beobachter-Muster* bekannt [GHJV08]. Für den Fall, dass der Top-Level-Dienst auf dem Handheld ausgeführt wird, muss die Listener-Registrierung durch das entfernte Aufrufen der zugehörigen Methode möglich sein.

In beiden Fällen liegt die Besonderheit darin, dass, bei der Ausführung eines Top-Level-Dienstes auf einem Handheld, Methoden entfernt mit *komplexen Objekten* (vom Typ EHomeService bzw. ServiceListener) aufgerufen werden. SimpleRMI wurde daher so realisiert, dass solche komplexen Objekte geeignet berücksichtigt werden: Soll bei einem entfernten Methodenaufruf als Parameter bzw. Rückgabewert eine Objektreferenz vom Typ EHomeService bzw. vom Typ ServiceListener übertragen werden, so wird zu diesem Objekt automatisch eine entsprechende RMISubProxy-Instanz erzeugt, um es für entfernte Aufrufe verfügbar zu machen.

Abbildung 5.27 zeigt am Beispiel des `Weckdienstes`, wie die entfernte Dependency Injection mit SimpleRMI funktioniert[3]. Nebenbei sei bemerkt, dass die folgenden Erläuterungen sich auf den Schritt (9) aus Abbildung 5.26 beziehen. Nachdem der `Weckdienst` auf dem Handheld instanziiert wurde, werden entsprechend ein Weckdienst-Stub und ein Weckdienst-Proxy erzeugt. Ein Beispiel hierfür ist der Aufruf `bindSubService(heizung)` durch den Deployer. Durch diesen Aufruf wird dem `Weckdienst` die Referenz auf den Treiberdienst `Heizung` übergeben, der beispielsweise für die Steuerung der Heizung im Badezimmer zuständig ist. Der Aktualparameter `heizung` ist dabei vom Typ Heizung, der wiederum eine Spezialisierung von EHomeService ist. Daher wird beim Serialisieren (`serialize(heizung)`) und Deserialisieren speziell darauf geachtet, dass nicht eine Kopie

[3]Die entfernte Listener-Registrierung funktioniert analog.

des Objekts heizung übertragen wird, sondern eine Referenz. Die Konsequenz daraus ist, dass beim Deserialisieren des Methodenaufrufs auf dem Handheld für den Parameter heizung automatisch das Heizung-SubProxy vom Typ RMISubProxy erzeugt wird. Dieser kann dann die Methodenaufrufe, die für die Heizung im eHome bestimmt sind, geeignet weiterleiten. Hierfür muss der Weckdienst-Stub dem `Weckdienst` zunächst jedoch die Referenz auf den neu generierten SubProxy übergeben, wie in Abbildung 5.27 dargestellt ist. Anschließend kann der Weckdienst beispielsweise die Temperatur im Raum auf 21°C stellen, bevor er die Person weckt. Weil der Parameter für die Temperatur den einfachen Datentyp real besitzt, wird dieser durch eine einfache Kopie übertragen, hier besteht also kein Bedarf für ein Stellvertreterobjekt.

SimpleRMI muss also zwischen einfachen und komplexen Objekten unterscheiden, um das Serialisieren bzw. Deserialisieren adäquat umzusetzen. Beim Serialisieren wird ein Methodenaufruf in eine JXTA-Nachricht umgewandelt, die folgende Informationen enthält:

1. Den Methodennamen
2. Die Anzahl der Aktualparameter
3. Für jeden Aktualparameter wird ein 2-Tupel bestehend aus dem Typ und dem eigentlichen Wert angelegt, wobei der Typ des Aktualparameters als String- und der Wert selbst als byte-Array vorliegen. Das Argument wird dabei mithilfe eines ObjectOutputStream serialisiert.

Genau bei der Serialisierung in Schritt 3 wirkt sich der Typ der Aktualparameter auf die Wirkungsweise von SimpleRMI beim Serialisieren und Deserialisieren aus, wie im Folgenden erläutert wird.

- **Serialisieren von Objekten des Typs RMIService**: Um Aktualparameter vom Typ RMIService in der Methode `serialize()` geeignet zu behandeln, wurde eine Klasse RMIObjectOutputStream implementiert, die ObjectOutputStream erweitert, die von Java zum Serialisieren von Objekten angeboten wird. Dabei wurde insbesondere die geerbte Methode `replaceObject(Object)` so überschrieben, dass Objekte vom Typ RMIService gesondert behandelt werden, bevor sie auf den OutputStream geschrieben werden[4]. Listing 5.3 zeigt einen Ausschnitt der Methode zum Serialisieren. Das Objekt vom Typ RMIService wird durch ein Objekt vom Typ RMIServiceReference ersetzt, bevor es auf den OutputStream geschrieben wird. Die RMIServiceReference-Instanz besitzt alle nötigen Informationen, um später der RMIService-Instanz wieder zugeordnet werden zu können: eine eindeutige ID (Zeilen 9 und 10) und den Klassennamen für den entfernten Aufruf (Zeilen 11 und 12). Abschließend wird die RMIService-Instanz, die als Aktualparameter übergeben werden soll, mit der genannten ID bei der entsprechenden RMIStub-Instanz registriert (Zeile 13). Alle übrigen Objekte, die nicht vom Typ RMIService sind, werden nicht gesondert behandelt (Zeile 16). Sie werden also kopiert und versendet.

[4]Es sei hier daran erinnert, dass alle Objekte, die entfernt aufrufbar sein sollen, wie etwa eHome-Dienste, direkt oder indirekt die Schnittstelle RMIService implementieren müssen.

```
1 public class RMIObjectOutputStream extends ObjectOutputStream {
2   [...]
3   protected final Object replaceObject(final Object object) {
4     if (object instanceof RMIService) {
5       // replace with service reference
6       RMIService service = (RMIService) object;
7       RMIServiceReference reference =
8         new RMIServiceReference();
9       reference.setServiceID(
10        String.valueOf(service.hashCode()));
11      reference.setServiceClass(
12        service.getRemoteClassName());
13      addSubService(reference.getServiceID(), service);
14      return reference;
15    } else {
16      return object;
17    }
18  }
19  [...]
20 }
```

Listing 5.3: Gesonderte Behandlung von Objekten vom Typ RMIService beim Serialisieren.

```
1 public class RMIObjectInputStream extends ObjectInputStream {
2   [...]
3   protected final Object resolveObject(final Object object) {
4     Object resolvedObject = object;
5     if (object instanceof RMIServiceReference) {
6       RMIServiceReference reference =
7         (RMIServiceReference) object;
8       Class serviceClass =
9         Class.forName(reference.getServiceClass());
10      resolvedObject = createSubProxy(
11        serviceClass, reference.getServiceID());
12    }
13    // return the resolved object
14    return resolvedObject;
15  }
16  [...]
17 }
```

Listing 5.4: Gesonderte Behandlung von Objekten vom Typ RMIServiceReference beim Deserialisieren.

- **Deserialisieren von Objekten des Typs RMIServiceReference**: Auch das Pendant zur Methode `serialize()` am anderen Ende der JXTA-Pipe, die Methode `deserialize()`, unterscheidet zwischen den Typen der zu entpackenden Objekte, um geeignete Maßnahmen treffen zu können. Sie ist in der speziell für die De-Serialisierung entwickelten Klasse RMIObjectInputStream enthalten. Diese ist eine Spezialisierung der Klasse ObjectInputStream und überschreibt die Methode resolveObject(Object), um Objekte des Typs RMIServiceReference gesondert behandeln zu können, wie in Listing 5.4 zu sehen: Falls ein zu entpackendes Objekt den Typ RMIServiceReference besitzt, wird es durch ein RMISubProxy-Objekt ersetzt (Zeilen 5–11). Gleichzeitig wird diesem Objekt die ID und der Klassenname des eigentlichen (entfernten) Objekts zugeordnet. Diese Informationen waren zuvor beim Serialisieren dem RMIServiceReference-Objekt mitgegeben worden. Nun kann das erzeugte RMISubProxy-Objekt als Stellvertreter für das Objekt agieren, das entfernt auf Aufrufe wartet.

Zusammenfassend kann daher festgehalten werden, dass SimpleRMI sowohl einfache als auch komplexe Objekte, die RMIService implementieren, im Rahmen von entfernten Methodenaufrufen übertragen kann. Falls es sich um komplexe Objekte vom Typ RMIService handelt, werden auf der Client-Seite dynamisch RMISubProxy-Objekte als Stellvertreter erzeugt, um entfernte Methodenaufrufe zu ermöglichen. Dies wird insbesondere bei der entfernten Dependency Injection und Listener-Registrierung benötigt. Insgesamt wurde somit eine verteilte Dienstkomposition über die Grenzen einzelner OSGi-Gateways hinweg realisiert.

5.3.7 Realisierung von Benutzeroberflächen

In Abschnitt 5.1.3 wurde bereits konzeptionell beschrieben, wie ein Benutzer durch Benutzung seines Handhelds mit eHome-Diensten interagieren kann. In diesem Abschnitt werden nun die Details zur Realisierung von dienstspezifischen Benutzeroberflächen erläutert.

Die Realisierung von Benutzeroberflächen basiert auf der Idee, dass jeder Dienst, der für die Bedienung über ein Handheld vorgesehen ist, mit einer entsprechenden Benutzeroberfläche ausgestattet ist. Die Benutzeroberfläche wird mit in das Bundle gepackt, in dem sich auch die restlichen Ressourcen des Dienstes befinden. Dadurch wird Dienstentwicklern die Möglichkeit gegeben, speziell auf die Funktionalitäten ihrer Dienste zugeschnittene Benutzeroberflächen zu entwickeln.

Als Realisierungstechnik bietet sich hierfür wiederum Java an, weil die Prototypen sowohl für das Handheld als auch für das eHome-Gateway in Java geschrieben sind. Daher wurde der eHome-Prototyp im Rahmen dieser Arbeit so erweitert, dass Benutzeroberflächen in Form von Java-Archiven (JAR) ausgeliefert werden, die spezielle Java-Klassen für die Darstellung der Benutzeroberfläche enthalten. Diese können dann entweder auf speziellen Displays im eHome oder auf den Handhelds mobiler Benutzer angezeigt werden. Für den zweiten Fall werden die Benutzeroberflächen erst bei Bedarf dynamisch auf das Handheld übertragen. So wird gewährleistet, dass nur die benötigten Oberflächen auf dem Handheld verfügbar sind.

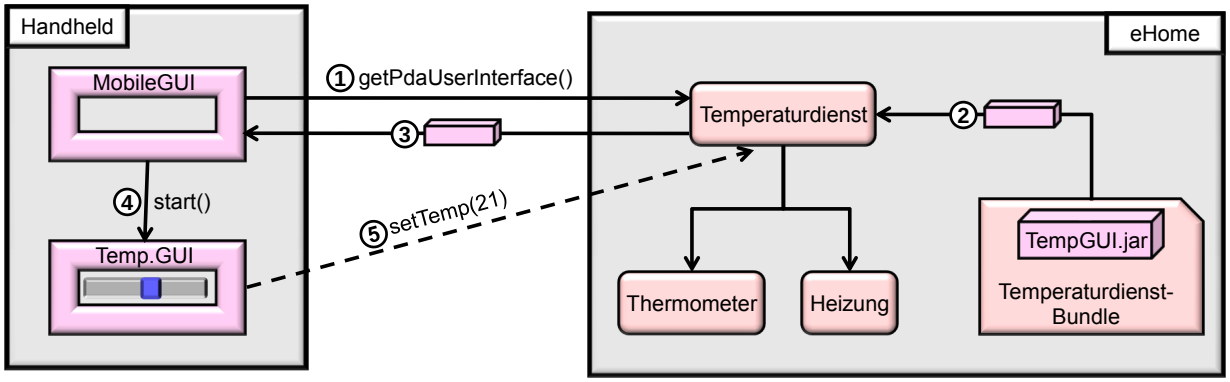


Abbildung 5.28: Übertragen von dienstspezifischen Benutzeroberflächen auf das Handheld und ihre Ausführung am Beispiel des `Temperaturdienstes`.

Der konkrete Ablauf der Übertragung von Benutzeroberflächen auf das Handheld ist in Abbildung 5.28 am Beispiel des `Temperaturdienstes` dargestellt. Hat der Benutzer den Dienst zur Interaktion ausgewählt, muss die MobileGUI sicherstellen, dass die zugehörige Benutzeroberfläche vom eHome auf das Handheld übertragen und dort dargestellt wird. Hierfür muss jeder Dienst mit einer Benutzeroberfläche die Methode `getPdaUserInterface()` implementieren, die dann von der MobileGUI aufgerufen wird (1). Der Dienst selbst holt sich das Java-Archiv (TemGUI.jar), in dem die Implementation der Benutzeroberfläche enthalten ist, aus seinem zugehörigen Bundle (2) und leitet es weiter an die MobileGUI (3). Letztere erzeugt aus dem Archiv eine Instanz der Benutzeroberfläche (Temp. GUI) und startet diese durch den Aufruf der Methode `start(EHomeService service, String token, ...)` (4). Dabei sind nur die zwei wichtigsten Parameter angegeben, die folgende Bedeutungen haben:

- **service**: Dient als Referenz auf den tatsächlichen Dienst im eHome. Über diese Referenz kann die Benutzeroberfläche Benutzeraktionen wie etwa die Erhöhung der Raumtemperatur an den Dienst weiterleiten.

- **token**: Enthält das Pseudonym des Benutzers, unter dem er im eHome bekannt ist. Dieser Parameter ist nur für personalisierbare Dienste interessant, weil sie ihre Anzeige an den aktuellen Benutzer anpassen können. Nicht-personalisierbare Dienste bieten jedem Benutzer dagegen dieselben Informationen an, sodass das Token für sie nicht benötigt wird. Im nächsten Kapitel werden Tokens genauer betrachtet.

Die übrigen Parameter werden hier nicht erläutert, weil sie lediglich von technischer Bedeutung sind. Sie enthalten etwa Informationen über das Java-Archiv, aus dem weitere Ressourcen wie Bilder geladen werden können.

Nachdem nun eine solche Benutzeroberfläche dynamisch erzeugt wurde, wird sie auf dem Handheld angezeigt, und der Benutzer kann darüber mit dem Dienst interagieren. In dem Beispiel des `Temperaturdienstes`, der in Abbildung 5.28 dargestellt ist, wird dem Benutzer

ein Schieberegler angezeigt, mit dem er die Raumtemperatur einstellen kann. Hat er eine Temperatur gewählt, beispielsweise `21°C`, wird die Auswahl über einen entsprechenden Methodenaufruf an den Dienst weitergeleitet (5). Dabei sind in der Abbildung die Details der genauen Kommunikation über SimpleRMI nicht dargestellt, sie verläuft analog zu den Ausführungen in Abbildung 5.27.

5.4 Zusammenfassung

In diesem Kapitel wurden die entwickelten Konzepte beschrieben, die mobile Benutzer im Rahmen der Inter-eHome-Mobilität unterstützen. Die Motivation für solch eine Unterstützung wurde dadurch begründet, dass sich die Benutzer im Alltag nicht nur in einem einzigen eHome aufhalten, sondern zwischen mehreren eHomes wechseln. Dabei wurden verschiedene Aspekte aufgegriffen, die besonders wichtig sind.

Zuerst wurde argumentiert, dass mobilen Benutzern eine Möglichkeit zur bequemen Interaktion mit eHome-Diensten geboten werden muss. Ausgehend von der Annahme, dass sie ein Handheld mit sich führen, wurde ein Lösungsansatz vorgeschlagen, der erlaubt, dass sich die Benutzer mit dem Handheld beim besuchten eHome anmelden, sich eine Liste verfügbarer Dienste anzeigen lassen und mit ausgewählten Diensten direkt über geeignete Benutzeroberflächen interagieren können. Dadurch entfällt die Notwendigkeit für die mühsame Verwendung des bereits vorhandenen eHomeConfigurator, der keine explizite Unterstützung für mobile Benutzer bietet.

Anschließend wurde der Aspekt der Personalisierung aufgegriffen. In diesem Zusammenhang wurden zwei Möglichkeiten identifiziert, die die Personalisierung ermöglichen. Als erste Möglichkeit wurde das Bereitstellen von Benutzerdaten für personalisierbare Dienste diskutiert, die sich im besuchten eHome bereits in Ausführung befinden. Eine besondere Anforderung an den Lösungsansatz bestand darin, dass eine eHome-übergreifende und nicht-redundante Bereitstellung der Daten möglich sein muss. Nach der Analyse dreier möglicher Alternativen wurde aus Gründen der besseren Eignung zum Schutz der Privatsphäre die Einführung eines mobilen Benutzermodells vorgeschlagen. Dadurch können die Benutzer ihre Daten auf ihren Handhelds mitnehmen und unterwegs den eHomes zur Verfügung stellen. Bei der Integration dieses Ansatzes in den bestehenden eHome-Prototyp wurde erläutert, wann und wie Benutzerdaten zwischen dem mobilen und dem Benutzermodell auf dem eHome-Gateway ausgetauscht werden.

Als zweite Möglichkeit der Personalisierung wurde die Nutzung persönlicher Dienste erläutert, die der Benutzer zusätzlich zu Benutzerdaten auf seinem Handheld mitführt. Diese Dienste können dann verwendet werden, wenn die von ihm gewünschten Funktionalitäten in dem aktuellen eHome nicht angeboten werden, aber die zur Erfüllung der Funktionalitäten benötigten Geräte und zugehörige Treiberdienste dort vorhanden sind. Der diesbezüglich vorgeschlagene Lösungsansatz bietet zwei Möglichkeiten: Entweder werden die Dienste ins eHome übertragen und dort ausgeführt oder sie werden auf dem Handheld ausgeführt. Für den ersten Fall konnte der bereits vorhandene SCD-Prozess mit den zugehörigen Werkzeugen unverändert eingesetzt werden. Für den zweiten Fall waren einige Erweiterungen

des Prozesses und der Werkzeuge nötig, um eine verteilte Komposition von Diensten über Gateway-Grenzen hinweg zu ermöglichen.

Schließlich wurden Realisierungsdetails der entwickelten Konzepte beschrieben. Wichtige Bausteine, die neu entwickelt wurden, sind der InformationService, der ServiceManager und der UMSynchronizer, die für die Koordination zwischen Handheld und eHome-Gateway zuständig sind. Ferner wurde beschrieben, wie die drahtlose Kommunikation umgesetzt wurde. Für die Vernetzung von Handhelds und eHome-Gateways wurde JXTA eingesetzt. Für die verteilte Kommunikation von Objekten wurde SimpleRMI entwickelt, das ähnlich wie Javas RMI entfernte Methodenaufrufe zwischen Objekten ermöglicht. Dies war nötig, weil zur Entwicklungszeit kein ähnlicher auf JXTA basierender Ansatz existierte, der hätte eingesetzt werden können. Beim Entwurf von SimpleRMI wurde insbesondere darauf geachtet, dass die Verteilungssituation von Diensten und übrigen Werkzeugen/Bausteinen verborgen wird, also die Kommunikation transparent durchgeführt wird. Abschließend wurde erklärt, wie Benutzeroberflächen zur Laufzeit dynamisch vom eHome-Gateway auf ein Handheld übertragen und dort dargestellt werden können.

Im nächsten Kapitel werden diese Konzepte erweitert, um den Schutz der Privatsphäre im Rahmen der Inter-eHome-Mobilität zu gewährleisten.

Kapitel 6

Privatsphäre

Während im letzten Kapitel erläutert wurde, wie mithilfe eines Handhelds mobile Benutzer im Rahmen der Inter-eHome-Mobilität dabei unterstützt werden können, eHome-Dienste zu bedienen und zu personalisieren, widmet sich dieses Kapitel den Konzepten zum Schutz der Privatsphäre.

Damit personalisierbare eHome-Dienste ihre Funktionalitäten an ihre Benutzer anpassen können, benötigen sie unterschiedliche Arten von Benutzerdaten. Doch immer dort, wo persönliche Daten anfallen, besteht die Gefahr der Verletzung der Privatsphäre. Damit die Privatsphäre gewahrt werden kann, muss jeder Benutzer u. a. jederzeit überwachen und bestimmen können, wer wann welche Informationen über ihn erhält. Ferner sollten Benutzerdaten nur *sparsam* herausgegeben und nur *zweckgebunden* verwendet werden (s. Abschnitt 2.5). Es soll weiterhin unbefugter Zugriff auf Benutzerdaten vermieden werden.

Übertragen auf eHomes ergeben sich konkret folgende Anforderungen bezüglich der Verarbeitung von Benutzerdaten: Die Daten eines Benutzers sollen

- nur von dem beabsichtigten *eHome*,
- nur durch die von dem Benutzer beabsichtigten *Dienste* und
- nur für den vorgesehenen *Zweck*

verwendet werden können. Diese Aspekte wurden in den Vorgängerarbeiten des eHome-Projekts nicht behandelt und sind Thema dieser Arbeit.

Die meisten Ansätze in der Literatur, die sich mit dem Schutz der Privatsphäre befassen, versuchen dieses Ziel dadurch zu erreichen, dass sich die Daten sammelnden Parteien an Datenschutzrichtlinien halten (s. etwa [CLMR06]). Die Erfahrung zeigt jedoch, dass es im Prinzip keine Möglichkeit mehr gibt, zu kontrollieren, was mit personenbezogenen Daten geschieht, wenn sie einmal in den Besitz von Dritten geraten. Oft werden diese dann doch *zweckentfremdet* verarbeitet, an Unbefugte weitergegeben bzw. weiterverkauft oder sie kommen aufgrund von Datenlecks abhanden, wie die Fälle in [Hei09, Wel09, sue10] zeigen.

Daher verfolgt diese Arbeit einen etwas anderen Ansatz, wie im Folgenden erläutert wird. Zunächst wird in Abschnitt 6.1 ein aushandlungsbasiertes Identitätsmanagement vorgestellt, der zur Datensparsamkeit beiträgt. Dieses Konzept wird anschließend um die Möglichkeit

der anonymen Authentifizierung erweitert, wodurch der Personenbezug der Daten gelöscht und ihre Unverkettbarkeit zwischen mehreren eHomes gefördert wird. Anschließend wird in Abschnitt 6.2 der Ausbau des Benutzermodells um eine selektive Zugriffskontrolle und um die Verwaltung von Pseudonymen beschrieben, womit die Herausgabe von Benutzerdaten an eHome-Dienste kontrolliert werden kann. Als Nächstes werden Einzelheiten der Realisierung dieser Konzepte erörtert. Schließlich wird dieses Kapitel mit einer Zusammenfassung abgeschossen.

6.1 Schutz auf eHome-Ebene

In diesem Abschnitt werden die Konzepte beschrieben, die zum Schutz der Privatsphäre auf eHome-Ebene entwickelt wurden. Zunächst werden die Konzepte vorgestellt, welche die Benutzer bei der Umsetzung der Bestandteile des „normativen Schutzprogramms“ unterstützen. Hierzu gehören insbesondere *Erforderlichkeit*, *Zweckbindung*, *Mitwirkung*, *Einwilligung*, *Minimierung* und *Transparenz* (s. Abschnitt 1.3.4). Anschließend wird die erzielte Unverkettbarkeit von Benutzerdaten durch Anonymität beschrieben.

6.1.1 Datensparsamkeit

Im vorigen Kapitel wurde beschrieben, wie Benutzerdaten automatisch zwischen einem Handheld und einem eHome synchronisiert werden können, um sie den personalisierbaren Dienste des eHomes zur Verfügung zu stellen. Nun kann genau hier angesetzt werden, um den Benutzer in den Prozess der Datenfreigabe einzubeziehen. Dadurch wird der Benutzer dabei unterstützt, die Menge der im Rahmen der Inter-eHome-Mobilität offengelegten Daten zu minimieren.

Im Folgenden wird erläutert, wie ein Benutzer mit dem eHome aushandeln kann, welche Dienste er dort verwenden möchte und entsprechend seiner Auswahl die Teilidentität bestimmen kann, die freigegeben wird. Dieser Ansatz wird als *Identitätsmanagement* bezeichnet (s. auch Abschnitt 2.4.5). Anschließend wird dargelegt, wie die Ausführung von persönlichen Diensten auf dem Handheld zur Datensparsamkeit beitragen kann.

6.1.1.1 Identitätsmanagement

Wie bereits im Einführungskapitel erwähnt, werden Benutzer im Rahmen der Inter-eHome-Mobilität eHomes *unterschiedlichen* Charakters besuchen. Angefangen von privaten Haushalten über Hotels, Arbeitsplätzen und öffentlichen Gebäuden sind viele Formen von intelligenten Umgebungen denkbar. Diese Vielfalt von eHome-Umgebungen wirkt sich auch auf die Arten personalisierbarer Dienste aus, die in jedem eHome angeboten werden. Beispielsweise können in einem Hotel ein Musikdienst und ein Weckdienst mit personalisierbaren Funktionalitäten angeboten werden. Solche Dienste sind für den Arbeitsplatz hingegen eher weniger geeignet. Hier wäre ein Dienst, der automatisch erkennt, in welchem Büro sich ein Mitarbeiter befindet und seine Anrufe direkt dorthin weiterleitet, von größerem Interesse.

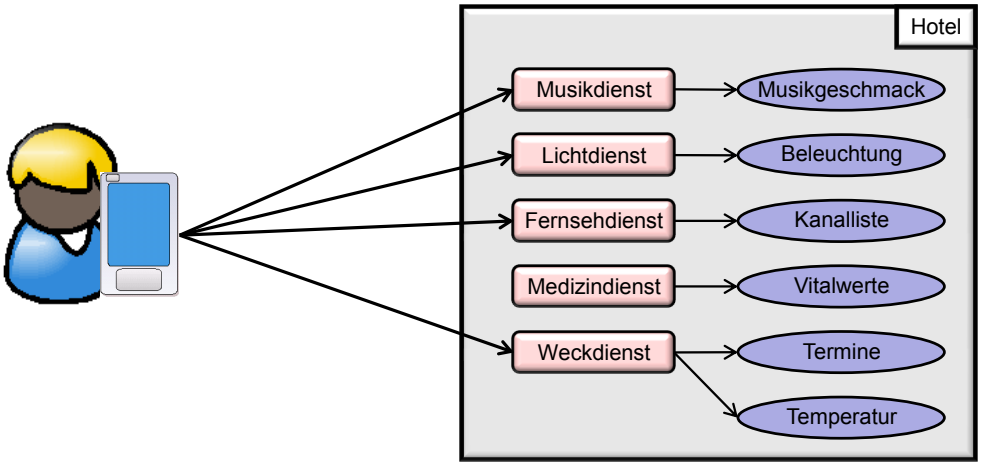


Abbildung 6.1: Benutzer nutzt nicht alle personalisierbaren Dienste.

Offensichtlich folgt hieraus, dass ein Benutzer in unterschiedlichen eHomes unterschiedliche eHome-Dienste, insbesondere auch unterschiedliche personalisierbare Dienste, nutzt. In diesem Zusammenhang wird auch der Begriff *Dienstnutzungsprofil* verwendet, mit der die Menge der Dienste bezeichnet wird, die ein Benutzer aktuell nutzt. Abbildung 6.1 zeigt einen Benutzer, der in seinem Hotelzimmer mehrere personalisierbare Dienste zur Auswahl hat. Zudem sind die von jedem Diensten benötigten Benutzerdaten angegeben. Der Benutzer hat sich in diesem Beispiel dafür entschieden, alle Dienste bis auf den `Medizindienst` zu verwenden. Also sind in dem Dienstnutzungsprofil des Benutzers die Dienste `Musikdienst`, `Lichtdienst`, `Fernsehdienst` und `Weckdienst` enthalten.

Wenn ein Benutzer aber nicht alle verfügbaren Dienste in Anspruch nimmt, muss er dem eHome auch nicht seine gesamten Benutzerdaten bereitstellen. Stattdessen reicht es aus, nur die Daten offenzulegen, die von den tatsächlich genutzten Diensten benötigt werden, wie beispielsweise in Abbildung 6.2 dargestellt ist.

Das Dienstnutzungsprofil eines Benutzers kann nicht nur von eHome zu eHome variieren, sondern auch von Sitzung zu Sitzung im selben eHome. Daher kann dann auch die Menge offenzulegender Benutzerdaten zwischen mehreren Sitzungen variieren. Beispielsweise könnte ein Hotel den `Weckdienst` nur für Gäste anbieten, die entweder einen Mehrpreis zahlen oder ein spezielles Zimmer buchen. Ob ein Gast diesen Dienst nutzt, könne davon abhängen, ob er privat oder geschäftlich unterwegs ist.

Wie in Abschnitt 2.4 erläutert wurde, kann die Menge der offengelegten Daten immer als *Teilidentität* aufgefasst werden. In Abbildung 6.2 tritt der Benutzer dem eHome demnach mit der Teilidentität entgegen, die nur die *Identitätsattribute* bzgl. `Termine`, `Temperaturpräferenz`, `Beleuchtungspräferenz`, `Kanalliste`, und `Musikgeschmack` enthält. Daher erhält das Hotel in diesem Beispiel keine weiteren Informationen über dem Benutzer.

Die *Datensparsamkeit* gegenüber einem eHome wird im Rahmen dieser Arbeit dadurch erreicht, indem der Benutzer die Möglichkeit erhält, die gewählte Teilidentität möglichst minimal zu wählen. Mit anderen Worten soll die freigegebene Datenmenge minimiert werden. Da die Teilidentität aber auch davon abhängt, welche Dienste der Benutzer nutzen möchte,

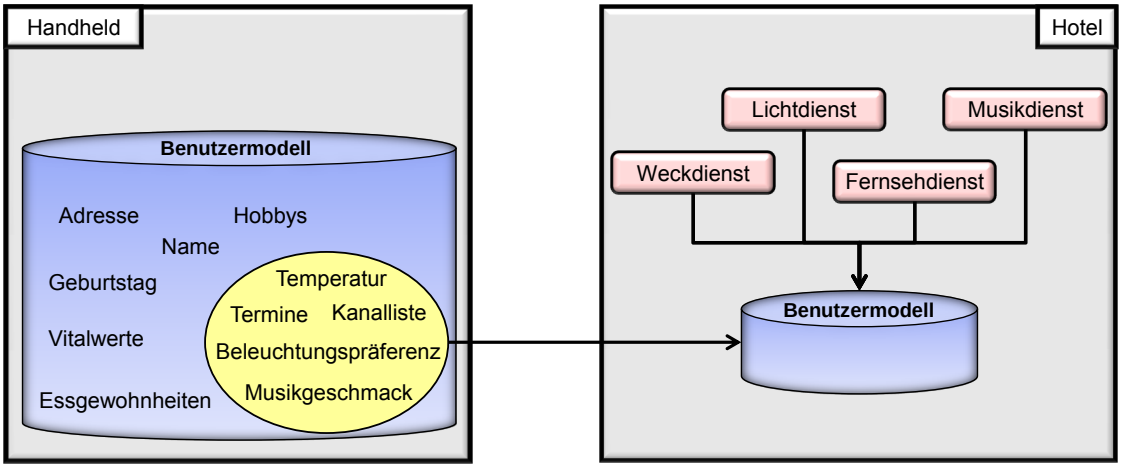


Abbildung 6.2: Daten werden entsprechend der genutzten Dienste offengelegt.

muss er mit dem eHome *aushandeln* können, welche Dienste er nutzen und welche Daten er offenlegen möchte.

Um diese Aushandlung zu ermöglichen, wurde im Rahmen dieser Arbeit eine Komponente namens IdentityManager entwickelt. Der IdentityManager hat unterschiedliche Aufgaben. Zum einen bietet er dem Benutzer die Möglichkeit, seine Identitäten auf seinem Handheld zu verwalten. Er kann neue Identitätsattribute anlegen, die Werte bestehender Attribute bearbeiten oder sie ganz löschen. Aus den Attributen der Hauptidentität kann er neue Teilidentitäten erzeugen, diese bearbeiten oder auch löschen. Der IdentityManager bietet dem Benutzer auch die Möglichkeit, eine Teilidentität für die anstehende Sitzung auszuwählen. Ferner sorgt der IdentityManager dafür, dass der UMSynchronizer (s. Abschnitt 5.2.1.2) dem eHome nur die Identitätsattribute zur Verfügung stellt, die in der gewählten Teilidentität enthalten sind.

Der Ablauf der Aushandlung von Diensten und Identitäten ist grob in Abbildung 6.3 dargestellt. Im ersten Schritt (1) erhält der Benutzer die Liste der im eHome verfügbaren Dienste (s. auch Abschnitt 5.1.2). Zusätzlich erhält der Benutzer Information darüber, welche Benutzerdaten jeder einzelne Dienst benötigt. Abhängig von seinen Wünschen, und den von jedem einzelnen Dienst geforderten Daten, kann der Benutzer im zweiten Schritt (2) nun auswählen, welche Dienste er benutzen möchte. Bei dieser Entscheidung muss er einen *Kompromiss* zwischen dem gebotenen Komfort durch zusätzliche Dienste und dem Schutz seiner Privatsphäre durch Minimierung freigegebener Daten finden. Der IdentityManager unterstützt ihn dabei, indem er ihm die zu den ausgewählten Diensten passenden Teilidentitäten zeigt. Aus diesen Teilidentitäten kann der Benutzer dann eine auswählen. Falls jedoch keine passende Teilidentität existiert, kann auch dynamisch eine generiert werden, die genau auf die aktuelle Situation zugeschnitten ist. Hier überlässt ihm der IdentityManager die volle *Kontrolle*. Im dritten Schritt (3) wird die gewählte Teilidentität dem eHome mitgeteilt.

Offensichtlich entsteht dem Benutzer durch die Aushandlung ein gewisser *Mehraufwand*. Doch dieser Mehraufwand ist für einen besseren Schutz der Privatsphäre nötig und kann

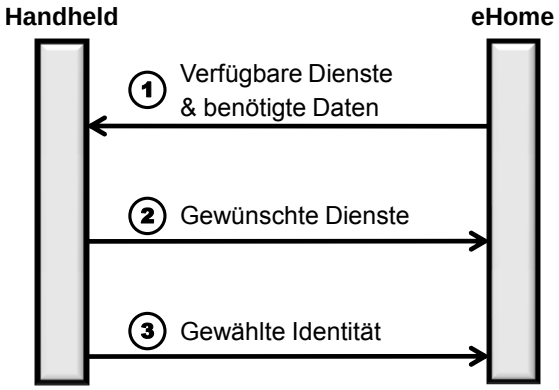


Abbildung 6.3: Grober Ablauf der Aushandlung von Diensten und Identitäten.

nie ganz vermieden werden. Um ihn jedoch möglichst reduzieren zu können, wurde der IdentityManager so konzipiert, dass das Ergebnis der Aushandlung auf dem Handheld gespeichert werden kann. Besucht der Benutzer dasselbe eHome später noch mal, kann der IdentityManager automatisch die passende Teilidentität vorschlagen, die beim letzten Mal gewählt wurde. Dadurch wird der Komfortverlust durch das Identitätsmanagement teilweise wieder kompensiert.

Zusammengefasst können mobile Benutzer im Rahmen der Inter-eHome-Mobilität durch Identitätsmanagement nun sparsamer mit der Freigabe ihrer Daten umgehen.

6.1.1.2 Ausführung persönlicher Dienste auf dem Handheld

In Abschnitt 5.2.2.2 wurde ein Ansatz vorgestellt, der mobilen Benutzern ermöglicht, persönliche Dienste auf ihrem Handheld auszuführen. Dort war die Motivation also der Mangel an Diensten im eHome. Die Ausführung persönlicher Dienste auf dem Handheld trägt aber auch *Datensparsamkeit* bei, wie am Beispiel des `Weckdienstes` dargelegt werden kann. Abbildung 5.12 zeigt die Ausführung des `Weckdienstes` auf dem Handheld. Insbesondere stellt die Abbildung klar, dass die von diesem Dienst benötigten Benutzerdaten auf dem Handheld verbleiben können und nicht an das eHome übertragen werden müssen.

Es macht daher Sinn, das Hotelbeispiel aus Abbildung 6.1 noch mal aufzugreifen. Dort hatte der Benutzer entschieden, den vom Hotel angebotenen `Weckdienst` zu nutzen. Aus diesem Grund musste er dem eHome seine `Termine` und Informationen über seine `Temperaturpräferenz` mitteilen. Während die Bekanntgabe der `Temperaturpräferenz` nicht besonders sensibel zu sein scheint, ist es bei `Terminen` offensichtlich anders. Dort sind nämlich Informationen über den Ort und die Uhrzeit des Termins und möglicherweise über den Terminpartner enthalten. Es ist offensichtlich, dass diese Daten nach Möglichkeit nicht preisgegeben werden sollten.

Bezogen auf die `Temperaturpräferenz` bietet die Ausführung des `Weckdienstes` auf dem Handheld hinsichtlich der Privatsphäre keinen Mehrwert. Denn diese Präferenz kann vom Hotelbetreiber trotzdem ermittelt werden, weil die gewünschte Temperatur der `Heizung` im Hotelzimmer übermittelt werden muss und dort abgefangen werden kann. Anders ist

es aber bei den Informationen zu den `Terminen`. Weil die Routenberechnung auch auf dem Handheld ausgeführt werden kann, um die optimale Weckzeit zu berechnen, wird das Hotel diese Informationen nicht in Erfahrung bringen können. Die einzige Information, die diesbezüglich gewonnen werden könnte, ist die Weckzeit. Daraus ist aber keine Information über einen konkreten Termin ableitbar.

Durch die Kombination des aushandlungsbasierten Identitätsmanagements mit der Ausführung persönlicher eHome-Dienste auf dem Handheld erlangen die Benutzer nun die Kontrolle über ihre Daten. Sie können nun komfortabel bestimmen, wer wann welche Daten von ihnen erhält.

6.1.2 Unverkettbarkeit durch Anonymität

Im vorigen Abschnitt wurde diskutiert, wie Datensparsamkeit zum Schutz der Privatsphäre beiträgt. Dadurch wird die Menge offengelegter Benutzerdaten minimiert. Dennoch haben diese Daten einen *Personenbezug*, der zur Verkettbarkeit der Daten herangezogen werden kann. In diesem Abschnitt werden die zum Entfernen des Personenbezugs entwickelten Mechanismen vorgestellt.

6.1.2.1 Das Problem der Verkettbarkeit

Eine erste Idee zur Authentifizierung von Benutzern beim eHome sah zwei Möglichkeiten vor. Entweder sind die Benutzer dem eHome schon bekannt und können sich mit einem vorgegebenen Pseudonym und dem zugehörigen Passwort bei dem eHome anmelden. Dieses Verfahren kann beispielsweise zu Hause oder am Arbeitsplatz angewendet werden. Alternativ kann der Benutzer unter Vorlage gewisser Dokumente (Ausweis, Führerschein, usw.) gewisse Eigenschaften von sich nachweisen, um Zugriff auf gewünschte Dienste zu erlangen.

Beide Ansätze haben jedoch den Nachteil, dass die Aktionen und Daten (bzw. Teilidentitäten) eines Benutzers *verkettbar* sind. Im zweiten Fall liegt lediglich Level-5-Anonymität vor, das eHome kann hier den Benutzer also eindeutig identifizieren (s. Abschnitt 2.4.3). Im ersten Fall ist der Benutzer dem eHome zwar unter einem Pseudonym bekannt, aber die Verbindung zur reellen Identität des Benutzers ist dem eHome meist auch bekannt (Level-3- oder Level-4-Anonymität).

Die Tatsache, dass die Teilidentitäten, mit denen ein Benutzer unterschiedlichen eHomes oder demselben eHome in unterschiedlichen Sitzungen gegenübertritt, auf die echte Identität des Benutzers zurückgeführt werden können, kann den obigen Ansatz zur Datensparsamkeit untergraben. Denn wenn die eHomes ohne Wissen des Benutzers zusammenarbeiten und ihre Daten miteinander verketten, können sie mehr Informationen über den Benutzer erlangen, als dieser ihnen offenlegen wollte.

In Abbildung 6.4 ist ein Benutzer abgebildet, der im Rahmen der Inter-eHome-Mobilität mit mehreren eHomes in Kontakt tritt und dabei persönliche Daten hinterlässt. Ferner sind zwei Möglichkeiten der Verkettbarkeit gezeigt. Im ersten Fall (1) kooperieren mehrere eHomes, um ihr Wissen über den Benutzer zu vereinigen. Das können sie machen, weil sie die echte Identität des Benutzers kennen oder sie ermitteln können. Im zweiten Fall (2)

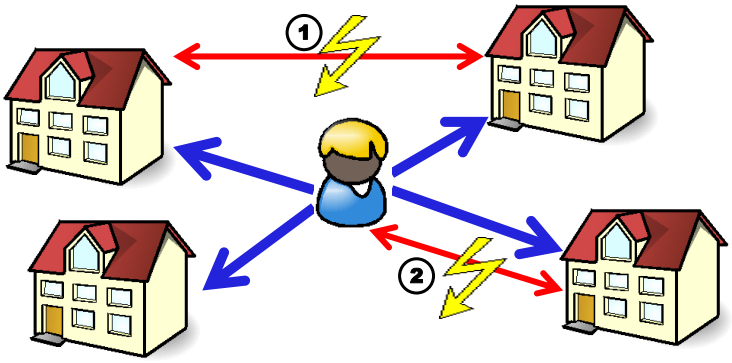


Abbildung 6.4: Verkettbarkeit im Rahmen der Inter-eHome-Mobilität.

sammelt ein einzelnes eHome Informationen über den Benutzer über mehrere Sitzungen hinweg. Um den Grad der Privatsphäre zu erhöhen, sollte auch diese Art der Verkettbarkeit verhindert werden. In Abschnitt 1.2.2 und Abschnitt 1.3.4 wurde anhand einiger Beispiele bereits erläutert, welche Risiken mit der Verkettbarkeit verbunden sind.

Eine erste Idee, um die Verkettbarkeit *zwischen* eHomes zu verhindern, ist die Nutzung von Pseudonymen, die nicht auf die reelle Identität des Benutzers zurückgeführt werden können. Solche Pseudonyme erlauben die Verkettung verschiedener Sitzungen eines Benutzers, ohne seine echte Identität offenzulegen. Es liegt dann Level-2-Anonymität vor. Falls der Benutzer dann für jedes eHome ein anderes Pseudonym verwendet (Beziehungspseudonym, s. Abschnitt 2.4.4), können diese ihre Daten nicht mehr einfach miteinander verketten. Aber über die Verkettung mehrerer Sitzungen *im selben* eHome könnte eine Historie der Dienstnutzungsprofile des Benutzers und der dabei anfallenden Daten aufgezeichnet werden. Weil so die Menge der gesammelten Informationen steigt, wird eine auch die Verkettbarkeit zwischen den eHomes dadurch vereinfacht. Je mehr Daten die eHomes nämlich über ihre Benutzer kennen, desto einfacher können Data-Mining-Algorithmen die Gemeinsamkeiten unterschiedlicher Profile erkennen.

In einem weiteren Schritt wird der Schutz der Privatsphäre durch die Verwendung von unterschiedlichen Pseudonymen für unterschiedliche Sitzungen verbessert. Dadurch kann der Benutzer während einer Sitzung zwar von anderen Benutzern unterschieden werden. Nach seiner Abmeldung vom System bzw. vom eHome kann er jedoch nicht mehr erkannt werden. Ein solcher Ansatz ermöglicht Level-1-Anonymität. Somit würden die Möglichkeiten der Verkettung mehrerer Sitzungen eines Benutzers weiter erschwert werden. Natürlich bietet auch solch ein Ansatz nur dann einen geeigneten Schutz der Privatsphäre, wenn die in den einzelnen Sitzungen offengelegten Daten nicht ohne Weiteres zur eindeutigen Identifizierbarkeit des Benutzers führen. Beispielsweise ist die Anonymitätsmenge (s. Abschnitt 2.4.3) besonders groß, wenn nur Präferenzen bezüglich allgemeiner Attribute wie Temperatur, Beleuchtung oder Musik angegeben werden. Umgekehrt wird diese Menge kleiner, wenn Name und Geburtsdatum zusammen angegeben werden.

Durch Einführung der Level-1-Anonymität wird einerseits die Benutzerakzeptanz für eHomes erhöht und andererseits werden gesetzliche Restriktionen bzgl. der Verarbeitung

von Benutzerdaten aufgeweicht [Sch01]. Natürlich macht die Anwendung einer solchen Anonymität im häuslichen Umfeld nicht viel Sinn, weil dort einerseits die Anonymitätsmenge zu klein ist und andererseits Anonymität dort eher nicht gewünscht ist. Dagegen eignet sich so etwas beispielsweise viel besser in Hotels, in öffentlichen Orten wie Bahnhöfen oder Flughäfen und in Bürogebäuden. Denn in solchen eHomes existieren erstens große Benutzergruppen und Benutzerfluktuationen. Zweitens ergeben sich in solchen Umgebungen oft kurzzeitige Anwendungsbeziehungen, wo die Benutzer keinen großen Komfortverlust aufgrund der Anonymisierung befürchten müssen.

6.1.2.2 Trusted-Third-Party-Credentials

Die Realisierung der Level-1-Anonymität im Rahmen der Inter-eHome-Mobilität ist eines der wesentlichen Ziele dieser Arbeit. Dieses Ziel kann durch den Einsatz *anonymer Credentials* erreicht werden. Sie bieten nämlich die erforderlichen Eigenschaften, insbesondere die Kombination von Zurechenbarkeit und Anonymität, um die Personalisierung von eHomes mit dem Schutz der Privatsphäre durch die Unverkettbarkeit von Teilidentitäten in Einklang zu bringen.

Im Rahmen dieser Arbeit werden zwei Arten von Credentials unterschieden, sprich *TTP-Credentials* und *Session-Credentials*, von denen Letztere im nächsten Abschnitt erläutert werden. TTP-Credentials sind solche, die von vertrauenswürdigen Dritten, sogenannten *Trusted Third Parties* (TTP), ausgestellt werden. Mobile Benutzer können diese verwenden, um sich an eHomes zu authentifizieren bzw. nachzuweisen, dass sie die für die Nutzung der ausgehandelten Dienste erforderlichen Rechte besitzen.

Solch ein Credential kann sich der Benutzer von einer geeigneten *Zertifizierungsstelle* (Aussteller) ausstellen lassen. Beispiele für solche Stellen sind vielfältig. Das Einwohnermeldeamt kann etwa Credentials ausstellen, die ein Mindestalter des Benutzers oder seinen Wohnort attestieren. Das Straßenverkehrsamt kann ein Credential anfertigen, das die Berechtigung zum Führen bestimmter Fahrzeuge bescheinigt. Das Studierendensekretariat einer Universität kann hingegen ein Credential erstellen, das nachweist, dass es sich bei dem Besitzer um einen eingeschriebenen Studenten handelt. Ein Lehrstuhl wiederum kann seinen Mitarbeitern Credentials gewähren, die sie für die Nutzung bestimmter Ressourcen wie etwa dem Faxgerät benötigen.

In Abbildung 6.5 ist das Beispiel eines Benutzers dargestellt, der sich am `Lehrstuhl i3` authentifiziert. Der Benutzer ist im Besitz mehrerer TTP-Credentials, die er von unterschiedlichen Ausstellern erhalten hat. Für den Zugriff auf den `Faxdienst` sei angenommen, dass er sein Mitarbeiterverhältnis am Lehrstuhl nachweisen muss. Wie in Abschnitt 2.8 bereits erläutert, wird dieser Nachweis über einen Zero-Knowledge-Beweis durchgeführt. In der Abbildung ist der Nachweis in Form eines Hexagons mit der Beschriftung `M. i3` dargestellt. In diesem Fall befindet sich der Benutzer in der Anonymitätsmenge der Mitarbeiter. Das eHome weiß also, dass es sich bei dem Benutzer um einen Mitarbeiter handelt, aber nicht um welchen *konkreten* Mitarbeiter. Möchte der Benutzer beispielsweise den `Faxdienst` nicht benutzen, könnte er sich mit dem Credential authentifizieren, das ihn als Student ausweist. Angenommen ist hier ein Student, der am Lehrstuhl seine Diplomarbeit schreibt

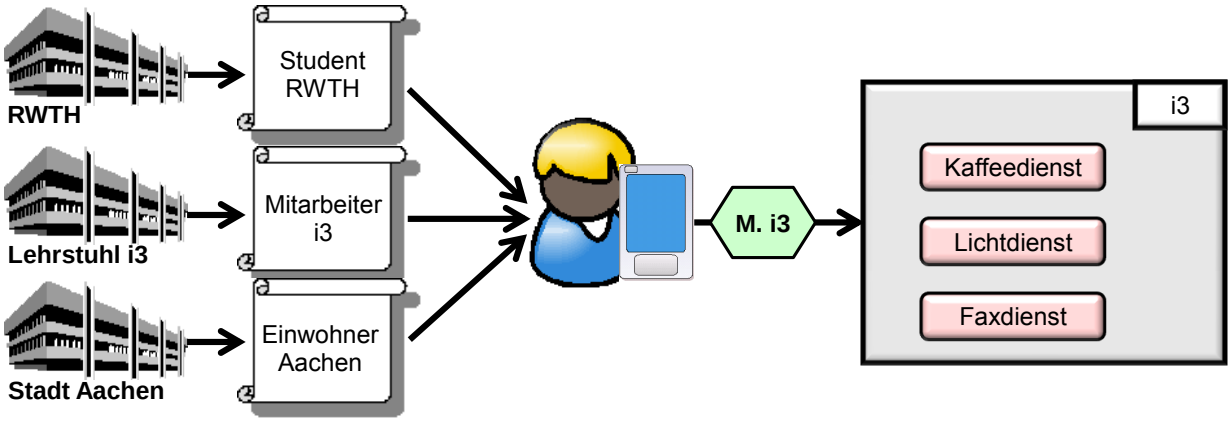


Abbildung 6.5: Auswahl eines TTP-Credentials am Beispiel eines Lehrstuhls.

und auch dort Kaffee trinken kann. Somit würde sich der Mitarbeiter in einer größeren Anonymitätsmenge befinden, die sowohl Mitarbeiter als auch Studenten enthält.

Ein anderes Beispiel, das auch im Einführungskapitel beschrieben wurde, ist in Abbildung 6.6 dargestellt. Hier möchte der Benutzer in einem Hotel einchecken und muss daher nachweisen können, ein Zimmer in diesem Hotel gebucht zu haben. Es sei angenommen, dass er über ein Reisebüro ein Zimmer mit `All-inclusive`-Leistungen gebucht hat. Den Buchungsnachweis hat er in Form eines anonymen TTP-Credentials erhalten. Außerdem besitzt er noch ein weiteres TTP-Credential von der Stadt Aachen, die ihm seinen Wohnort attestiert. Für das Hotel ist jedoch, wie dargestellt, nur der Nachweis der ersten Credentials nötig. Durch diesen Nachweis weiß das Hotel, dass der Benutzer beim bekannten Reisebüro gebucht und seine Zahlung geleistet hat.

In beiden Beispielen erhält der Benutzer ein oder mehrere TTP-Credentials von einem Aussteller und zeigt diese im Rahmen der Inter-eHome-Mobilität einem eHome vor. Die eHomes wiederum nehmen die Rolle des Verifizierers an und verifizieren die Gültigkeit der Credentials. Wichtig ist hier, dass sowohl der Benutzer als auch der Verifizierer dem Aussteller vertrauen. Wenn also ein Benutzer ein Credential von einem vertrauenswürdigen Aussteller vorzeigt, genügt dieses Credential als Nachweis für die Autorisierung des Benutzers für die Nutzung von Diensten.

Durch den Einsatz von TTP-Credentials wird somit die Verkettbarkeit von Teilidentitäten verhindert. Denn so wird ein Benutzer zwar authentifiziert, die eHomes erhalten aber keine Informationen über die echte Identität des Benutzers, auch nicht durch heimliches Zusammenarbeiten. Um jedoch die Aktionen des Benutzers für die Dauer einer Sitzung zueinander zuzuordnen, wird ihm nach der Authentifizierung ein Pseudonym zugewiesen. Diese Pseudonyme sind aber nur für eine Sitzung gültig, es handelt sich hier also um eine eingeschränkte Art von Rollenbeziehungspseudonymen (s. Abschnitt 2.4.4). Das gilt auch für den speziellen Fall, in dem der Aussteller eines Credentials gleichzeitig auch der Verifizierer desselben Credentials ist.

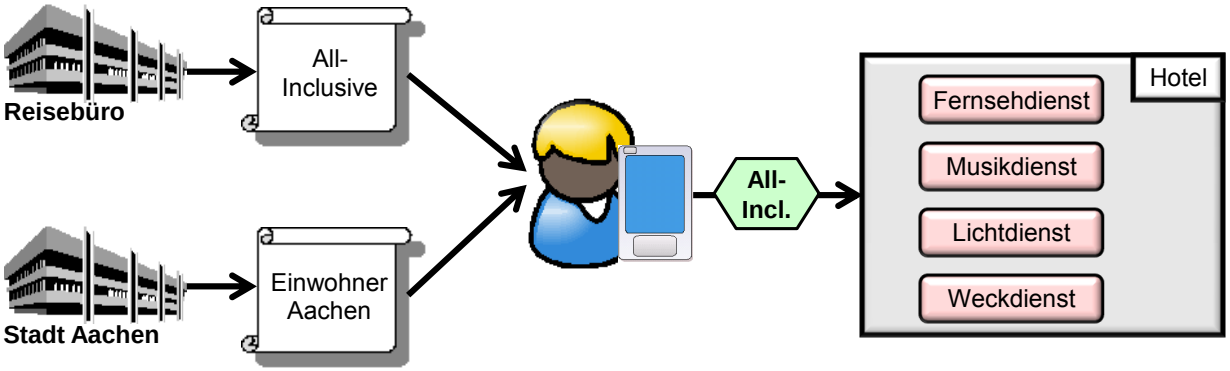


Abbildung 6.6: Auswahl eines TTP-Credentials am Beispiel eines Hotels.

6.1.2.3 Session-Credentials

Hat sich ein Benutzer mit einem TTP-Credential erfolgreich an einem eHome angemeldet, befindet er sich in einer aktiven Sitzung. Nun hat er zwei Möglichkeiten, mit den Diensten zu interagieren. Entweder indirekt über seine Benutzerdaten (s. Abschnitt 5.2.1) oder direkt über Benutzeroberflächen der Dienste (s. Abschnitt 5.1.3). Nun stellt sich die Frage, wie das eHome zuverlässig unbefugten Zugriff verhindern kann. Beispielsweise sollen in einem Lehrstuhl nur die Mitarbeiter das Faxgerät verwenden oder in einem Hotel nur die Gäste die Sauna nutzen dürfen, die dafür bezahlt haben.

Bei jedem Zugriff auf einen Dienst muss der Benutzer sich also einer Zugriffskontrolle unterziehen. Weil dabei die Privatsphäre der Benutzer möglichst gut geschützt werden soll, bietet es sich an, anonyme Credentials auch für die Zugriffskontrolle innerhalb einer Sitzung zu verwenden. Beispielsweise soll der Hotelbetreiber aus Abschnitt 1.2.1 nicht erkennen können, welche Gäste gemeinsam in die Sauna gehen. Auch wenn die echten Identitäten der Gäste dem Hotelbetreiber unbekannt sind, weil sie sich mit TTP-Credentials authentifiziert haben, könnten zusätzliche Informationen über ihr Dienstnutzungsprofil sich negativ auf ihre Privatsphäre auswirken.

Auf den ersten Blick können hierfür die TTP-Credentials eingesetzt werden, weil aufgrund ihrer Eigenschaften verschiedene, mit demselben Credential ausgeführte Transaktionen nicht miteinander verkettbar sind. Es macht jedoch Sinn, für jede Sitzung ein eigenes Credential zu erstellen, weil dadurch mehr Flexibilität bei der Rechtevergabe erreicht werden kann. Diese Credentials werden im Folgenden als *Session-Credentials* bezeichnet.

Die Idee hinter Session-Credentials ist die, dass das eHome dem Benutzer nach der aushandlungsbasierten Authentifizierung ein neues Credential ausstellt. Demnach ist ein gültiges TTP-Credential Voraussetzung für ein Session-Credential. Während das TTP-Credential aber prinzipiell von einem externen Aussteller stammt und auch in mehreren eHomes eingesetzt werden kann, stammt das Session-Credential von einem eHome und kann auch nur in diesem verwendet werden.

Ein Session-Credential wird in der Regel so ausgestellt, dass es nur während einer Sitzung gültig ist. Verlässt der Benutzer das eHome, wird auch sein Session-Credential

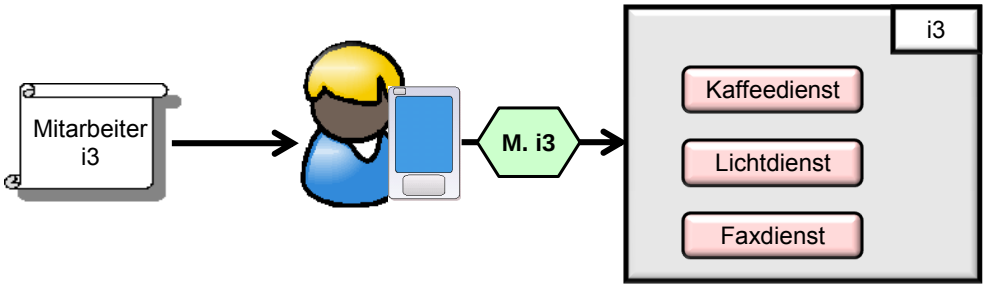


Abbildung 6.7: Rollenbasiertes Session-Credential.

ungültig. Ferner kann ein Session-Credential zeitlich begrenzt werden, beispielsweise auf einen Arbeitstag im Büro oder auf die Dauer der Buchung in einem Hotel. Ein TTP-Credential hingegen hat oft eine längere Gültigkeit, weil es für allgemeinere Zwecke ausgestellt wird. Ferner erlauben Session-Credentials eine genauere Zuteilung von Rechten innerhalb einer Sitzung, weil bei dessen Ausstellung spezifische Anforderungen des eHomes oder Aspekte der Aushandlung berücksichtigt werden können. Beispielsweise kann einen Lehrstuhlmitarbeiter verlangen, dass sein Session-Credential für die aktuelle Sitzung ihm nur Zugriff auf die Dienste erlaubt, die auch Studenten nutzen dürfen. Somit zählt er in der aktuellen Sitzung zu einer größeren Anonymitätsmenge, als sein TTP-Credential es suggerieren würde. Mit einem Session-Credential können zum Beispiel auch Rechte verbunden sein, bestimmte Räume zu betreten. Mit einem TTP-Credential, das von einem externen Aussteller stammt, lassen sich solch speziellen Anforderungen nicht einfach umsetzen.

Nachdem nun begründet wurde, warum Session-Credentials verwendet werden sollten, stellt sich die Frage nach ihrem Aufbau. Der Aufbau eines solchen Credentials ist frei gestaltbar. Im Folgenden werden zwei Möglichkeiten diskutiert:

- **Rollenbasiert**: Eine erste Möglichkeit sieht vor, die rollenbasierte Zugriffskontrolle mit *anonymen* Credentials nachzuahmen. Solch ein Credential enthält demnach nur ein Attribut, das die *Rolle* wiedergibt, die dem Benutzer zugewiesen wurde. Diese Rolle berechtigt ihn, die ausgehandelten Dienste zu nutzen. Beispiele für solche Rollen sind `Student`, `Mitarbeiter`, `Gast_mit_Vollpension` oder `Erwachsene`. Wenn der Benutzer nun auf einen Dienst zugreifen möchte, muss er mit seinem Session-Credential nachweisen, dass er die entsprechende Rolle besitzt. Abbildung 6.7 zeigt dies am Beispiel des Lehrstuhls. Es ist hier angenommen, dass er sich als Mitarbeiter authentifiziert hat. Daher hat er auch ein Session-Credential erhalten, dass die Rolle Mitarbeiter enthält. Nun muss er den Besitz dieser Rolle bei jedem Dienstzugriff vorzeigen, egal ob er auf den `Lichtdienst` oder auf den `Faxdienst` zugreifen möchte. Das eHome prüft dann, ob die nachgewiesene Rolle das Recht für den Zugriff auf den gewünschten Dienst enthält. Erst dann wird ihm der Zugriff gewährt oder verweigert.

  Der Nachteil dieser Variante ist jedoch der, dass das eHome den Benutzer sofort einer Rolle zuordnen kann. Und wenn zu einer bestimmten Zeit nur wenige Mitglieder dieser Rolle im eHome aktiv sind, besteht die Gefahr der Zuordnung von Dienstzugriffen auf

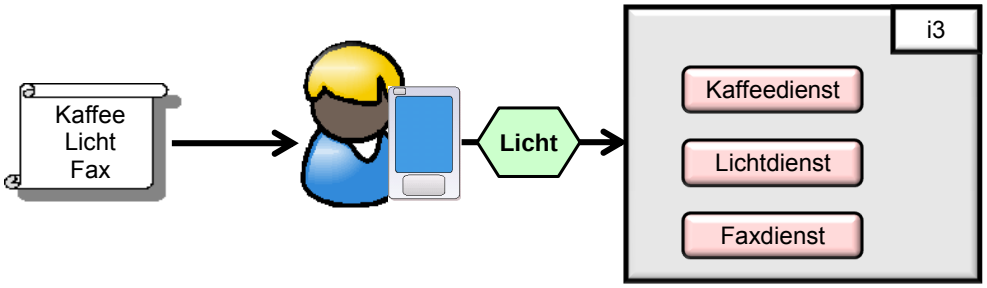


Abbildung 6.8: Dienstbasiertes Session-Credential.

eine bestimmte Person. Wenn beispielsweise die Rolle `Chef` nur einer einzigen Person zugeordnet ist, ist die Identifizierung dieser Person einfach möglich. Damit ist auch die Anonymität innerhalb einer Sitzung gefährdet.

- **Dienstbasiert**: Die zuvor genannten Schwächen lassen sich beheben, wenn das Session-Credential für jeden ausgehandelten Dienst ein eigenes Attribut enthält. Am Beispiel des Lehrstuhls würde das dann wie in Abbildung 6.8 dargestellt aussehen. Der Benutzer hat diesmal ein Session-Credential erhalten, in dem drei Attribute `Kaffee`, `Licht` und `Fax` enthalten sind. Wenn er nun beispielsweise auf den Lichtdienst zugreifen möchte, muss er nur noch nachweisen, dass er die Berechtigung zur Nutzung dieses einen Dienstes hat. Der Nachweis enthält daher nur das Attribut `Licht`. Dass er im Besitz weiterer Attribute ist, die ihn zur Nutzung weiterer Dienste berechtigen, muss er dem eHome nicht offenbaren. Die übrigen Attribute kann er deshalb geheim halten, weil anonyme Credentials die selektive Freigabe unterstützen (s. Abschnitt 2.8).

  Dadurch, dass der Benutzer nun nur noch die Berechtigung für die Nutzung eines einzelnen Dienstes nachweisen muss, kann das eHome den Benutzer nicht mehr einer bestimmten Personengruppe zuordnen. Dies gilt natürlich nur, wenn der Dienst von Personen unterschiedlicher Gruppen verwendet werden darf. Den `Lichtdienst` können beispielsweise Studenten als auch Mitarbeiter nutzen. Somit ist die Anonymitätsmenge größer, als wenn der Benutzer nur die Rolle nachweisen müsste, die ihn dann konkret der entsprechenden Gruppe zuordnet. Ferner erlaubt diese Variante, dass derselbe Benutzer in zwei verschiedenen Sitzungen unterschiedliche Session-Credentials erhält.

  Dienstbasierte Session-Credentials haben jedoch den Nachteil, dass ihre Ausstellung aufgrund der, im Vergleich zu rollenbasierten Session-Credentials, größeren Anzahl von Attributen länger dauert (s. hierzu [Pet09, CH02]). Eine weitere Eigenschaft, die sich nachteilig auswirken könnte, ist die, dass sich der Benutzer während einer aktiven Sitzung nicht spontan entscheiden kann, einen zuvor nicht ausgehandelten Dienst zu nutzen, obwohl er durchaus die Berechtigung hierfür hätte. Falls der Benutzer im gegebenen Beispiel sich als Student, und nicht als Mitarbeiter, am `i3` angemeldet und den `Faxdienst` nicht in sein Session-Credential aufgenommen hat, kann er diesen auch nicht nutzen. Möchte er diesen Dienst doch nutzen, muss er sich zunächst abmelden und einen erneuten Aushandlungsprozess durchlaufen. Diese Tatsache sollte

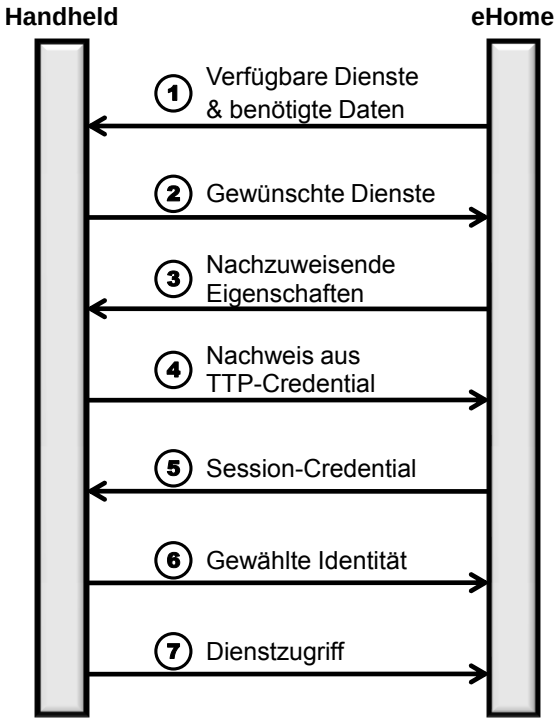


Abbildung 6.9: Um anonyme Credentials erweiterter Aushandlungsprozess.

der Benutzer berücksichtigen. Wenn er sich nicht sicher ist, ob er einen Dienst nutzen wird, sollte er ihn möglicherweise trotzdem in sein Session-Credential aufnehmen lassen, um ein späteres Durchlaufen des Aushandlungsprozesses zu vermeiden.

6.1.2.4 Auswirkung auf den Aushandlungsprozess

Die Verwendung anonymer Credentials wirkt sich auf den *Aushandlungsprozess* aus, der in Abschnitt 6.1.1.1 eingeführt wurde. Der Ablauf des erweiterten Prozesses ist in Abbildung 6.9 dargestellt. Die ersten zwei Schritte sind gleich geblieben. Ab Schritt (3) hat sich der Prozess jedoch verändert. Nachdem der Benutzer nun die gewünschten Dienste ausgewählt hat (2), fordert ihn das eHome auf, sich durch Nachweis bestimmter Eigenschaften zu authentifizieren (3). Diese Eigenschaften weisen seine Zugehörigkeit zum Kreis der berechtigten Personen nach und müssen ihm zuvor über TTP-Credentials attestiert worden sein (4). Ist der Nachweis vom eHome erfolgreich verifiziert worden, stellt dieser dem Benutzer wiederum das Session-Credential aus, dass ihm für die aktuelle Sitzung Zugriff auf genau die ausgehandelten Dienste gewährt (5). Erst danach gibt der Benutzer seine gewählte Teilidentität frei, überträgt also die darin enthaltenen Benutzerdaten (6). Im Laufe der Sitzung kann der Benutzer mithilfe des Session-Credentials nun anonym mit den Diensten interagieren.

6.1.2.5 Diskussion

Durch den Einsatz anonymer Credentials wird ein stärkerer Grad der Anonymität erreicht als bei Verwendung von etwa Benutzernamen und Passwort. Dabei liegt Level-1-Anonymität

vor, weil die eHomes ihre Benutzer nun unter nicht-wiedererkennbarer Pseudonyme kennen. Somit sind zwei Sitzungen eines Benutzers nicht mehr miteinander verkettbar. Beispielsweise kann ein Gast dasselbe Hotel mehrfach besuchen, ohne das festgestellt werden kann, ob er schon zuvor Gast gewesen ist. Damit wird auch die Erstellung eines Bewegungsprofils im Rahmen der Inter-eHome-Mobilität, wie es in Abschnitt 1.2.2 beschrieben wurde, erschwert.

Durch den Einsatz von dienstbasierten Session-Credentials kann zudem die Erstellbarkeit eines Dienstnutzungsprofils innerhalb einer Sitzung verhindert bzw. erschwert werden. Dies funktioniert insbesondere dann gut, wenn es sich um nicht-personalisierbare Dienste handelt, also die Dienste keine Benutzerdaten erhalten müssen und somit auch keine Verkettung von Transaktionen erforderlich ist. Wenn in einem eHome der Zutritt zu Räumen beispielsweise durch einzelne Dienste kontrolliert würde, könnte so auch das Erstellen von Bewegungsprofilen innerhalb eines eHomes verhindert werden.

Weil eHome-Benutzer jedoch auch personalisierbare Dienste verwenden, ergeben sich diesbezüglich einige Aspekte, die beachtet werden sollten. Nach der Authentifizierung legt der Benutzer dem eHome eine Teilidentität offen, die darin enthaltenen Identitätsattribute werden dem Benutzermodell im eHome zur Verfügung gestellt. Falls der Benutzer dem eHome gegenüber anonym bleiben möchte, sollte er darauf achten, einerseits sparsam mit den Daten umzugehen und andererseits keine Daten mit Alleinstellungsmerkmalen offenzulegen. Denn so wär eine Verkettbarkeit über die Teilidentitäten möglicherweise leichter. Ferner kann der Benutzer seine Wiedererkennbarkeit dadurch erschweren, indem er die Ausprägungen seiner Identitätsattribute vor der nächsten Anmeldung beim selben eHome ändert. Er kann beispielsweise seinen Musikgeschmack von Zeit zu Zeit variieren. Im nächsten Abschnitt wird das Zusammenspiel des Benutzermodells im eHome mit den personalisierbaren Diensten bezüglich des Schutzes der Privatsphäre genauer behandelt.

Einen besonderen Vorteil bieten anonyme Credentials in Bezug auf die Handhabung mobiler Benutzer, die sich von einem eHome in ein anderes begeben. In den bisherigen Arbeiten der eHome-Gruppe wurde stets davon ausgegangen, dass alle Benutzer dem eHome vor dessen Einrichtung schon bekannt sind. Zumindest war es notwendig, dass ein Administrator einen neuen Benutzer manuell anlegen und seine Benutzerdaten einpflegen muss, damit dieser Benutzer (personalisierbare) Dienste nutzen konnte. Diese Tatsache stellt jedoch eine große Hürde für die Inter-eHome-Mobilität dar, wie auch in Abschnitt 3.4.3 diskutiert wurde. Der Ansatz anonymer Credentials mindert dieses Problem insofern, dass konkrete Benutzer dem eHome nicht mehr a-priori bekannt sein müssen. Der Administrator muss nur noch vorgeben, welche Eigenschaften neue Benutzer nachweisen müssen, um bestimmte Dienste nutzen zu dürfen. Diese Eigenschaften können die Benutzer dann durch TTP-Credentials zuverlässig nachweisen, ohne zuvor dem eHome bekannt gewesen zu sein. Dies ist insbesondere in größeren eHomes wie Hotels oder Kinos von Vorteil, die ein besonderes Maß an Personenfluktuation vorweisen.

Ein Nachteil anonymer Credentials besteht hinsichtlich ihrer Beeinflussung der Performanz. Sie bieten zwar einen besseren Schutz der Privatsphäre, dieser bringt jedoch einen Performanzverlust mit sich. Dieser Aspekt ist in der Literatur ausführlich diskutiert, s. hierzu etwa [CH02, BDD07, Lap08]. Aus diesem Grund bietet es sich an, einen Kompromiss zwischen der Stärke der Anonymität und dem Grad der Benutzbarkeit zu finden. Es kann

nämlich sein, dass die Anonymität bzw. Unverkettbarkeit von Dienstzugriffen nicht in jedem eHome unbedingt notwendig ist, wie etwa zu Hause im privaten Umfeld. Daher wird dieser Ansatz in Abschnitt 7.1 so erweitert, dass die Benutzer selbst entscheiden können, ob und wie anonym sie einem eHome gegenübertreten möchten.

6.2 Schutz auf Dienstebene

Im vorigen Abschnitt wurden Konzepte vorgestellt, die zur Datensparsamkeit und Unverkettbarkeit von Identitäten im Rahmen der Inter-eHome-Mobilität beitragen. Dennoch müssen mobile Benutzer persönliche Daten – wenn auch nur in geringen Mengen – offenlegen, falls sie personalisierbare Dienste nutzen möchten.

In diesem Abschnitt werden diese Daten genauer betrachtet. Es wird untersucht, wie das in Kapitel 4 vorgestellte *Benutzermodell* zum weiteren Schutz der Privatsphäre *während einer Sitzung* beitragen kann.

Für die folgende Diskussion wird dabei vorausgesetzt, dass die Benutzer dem Benutzermodell im eHome mehr vertrauen, als den eHome-Diensten desselben eHomes. Hierfür wird angenommen, dass das Benutzermodell von einem vertrauenswürdigen Hersteller stammt und in vielen eHomes standardmäßig eingesetzt wird. Ferner kann das Benutzermodell von einem TTP signiert sein, der die Vertrauenswürdigkeit des Benutzermodells bescheinigt. Bei der großen Vielfalt möglicher eHome-Dienste wird es hingegen so sein, dass im selben eHome die Dienste unterschiedlicher Hersteller gemeinsam ausgeführt werden.

Es ist aber allgemein bekannt, dass viele Unternehmen legal oder illegal an der extensiven Sammlung personenbezogener Daten interessiert sind, weil sie dadurch unterschiedliche Ziele erreichen können. Die gesammelten Daten können beispielsweise weiterverkauft, für Werbung verwendet, zur weiteren Informationsgewinnung eingesetzt werden oder im schlimmsten Fall sogar zum Identitätsraub führen.

Im Rahmen dieser Arbeit wird daher das Ziel verfolgt, nicht allen Diensten in einem eHome die gleichen Daten zur Verfügung zu stellen, die zuvor zwischen dem Benutzer und dem eHome ausgehandelt wurden. Stattdessen sollen personalisierbare Dienste nur auf die Daten zugreifen dürfen, die für ihre Funktionalität notwendig sind. Der Benutzer soll zudem in den Prozess eingebunden werden, sodass er interaktiv bestimmen kann, welcher Dienst auf welche konkreten Daten zugreifen darf.

Um das verfolgte Ziel zu erreichen, wurde das Benutzermodell um die Geheimhaltung von Benutzerdaten erweitert. Dies wird zum einen durch selektive Zugriffskontrolle und zum anderen durch den Einsatz von Pseudonymen ermöglicht. Letztere unterstützen die Vermeidung von Informationsflüssen zwischen autorisierten und nicht-autorisierten Diensten. Mit anderen Worten wird die Verkettbarkeit von Benutzerdaten durch mehrere Dienste unterbunden. Die genauen Details hierzu werden in den folgenden Unterabschnitten beschrieben.

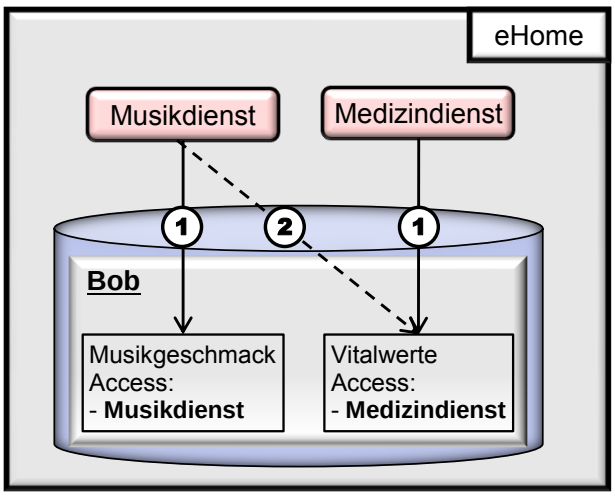


Abbildung 6.10: Vertraulichkeit durch selektiven Zugriff auf Benutzerdaten.

6.2.1 Selektive Zugriffskontrolle für Benutzerdaten

Das Benutzermodell in seiner Form aus Kapitel 4 verwaltet Benutzerdaten und stellt sie personalisierbaren Diensten ohne eine Zugriffskontrolle zur Verfügung. Daher muss das Benutzermodell so erweitert werden, dass ein Dienst nur noch auf die Daten zugreifen kann, die für seine eigene Funktionalität notwendig sind.

Hierfür eignet sich das Konzept der selektiven Zugriffskontrolle (s. [Sch01]). Die Selektive Zugriffskontrolle dient der Vertraulichkeit von Einträgen in einem Benutzermodell, wodurch explizit bestimmt wird, welche Dienste auf welche Einträge zugreifen dürfen. Nicht erwähnte Dienste sollen demnach keinen Zugriff haben.

Abbildung 6.10 zeigt an einem Beispiel, wie das Konzept der selektiven Zugriffskontrolle am Benutzermodell aussieht. Es sind die personalisierbaren Dienste `Musikdienst` und `Medizindienst` abgebildet, die auf die Daten des Benutzers Bob zugreifen möchten. Es ist ferner auch das Benutzermodell mit einem Auszug aus den Benutzerdaten von Bob dargestellt. Konkret sind die Identitätsattribute `Musikgeschmack` und `Vitalwerte` abgebildet. In dieser Abbildung ist nun neu, dass die Attribute um ein weiteres Feld, nämlich `Access`, erweitert wurden. Mit diesem Feld kann eine sehr *feingranulare* Zugriffskontrolle für Benutzerdaten realisiert werden. Es nimmt nämlich die Rolle einer *Zugriffskontrollliste* ein (engl. *Access Control List*). Für jeden Dienst, der auf das entsprechende Attribut zugreifen darf, wird ein Eintrag in dieser Liste vorgenommen. Alle anderen Dienste dürfen dieses Attribut nicht lesen oder schreiben.

Im konkreten Beispiel aus Abbildung 6.10 enthalten die Zugriffskontrolllisten beider Attribute jeweils einen Eintrag. Damit hat das Benutzermodell alle nötigen Informationen, um die selektive Zugriffskontrolle durchzuführen. Möchte nun der `Medizindienst` auf das Attribut `Vitalwerte` oder der `Musikdienst` auf das Attribut `Musikgeschmack` zugreifen, wird das Benutzermodell dies erlauben, weil die Namen dieser Dienste jeweils in den Zugriffskontrolllisten der Attribute enthalten sind. Dies ist durch die beiden Pfeile mit durchgängigen Linien angedeutet (1). Möchte der `Musikdienst` hingegen versuchen, auf

die `Vitalwerte` von Bob zuzugreifen, würde das Benutzermodell diesen Zugriff verweigern, dargestellt durch den gestrichelten Pfeil (2).

Im vorigen Beispiel durfte immer nur ein Dienst auf ein Attribut zugreifen. Dies muss jedoch nicht zwingend so sein. Beispielsweise ist es möglich, dass sowohl der `Temperaturdienst` als auch der `Weckdienst` die Berechtigung erhalten, auf das Attribut `Temperaturpräferenz` zuzugreifen.

Nun stellt sich hier die Frage, wer überhaupt bestimmt, welcher Dienst auf welche Attribute zugreifen darf. Im Rahmen dieser Arbeit wurde der Ansatz verfolgt, dass der Benutzer interaktiv in den Entscheidungsprozess eingebunden wird. Der eHome-Prototyp wurde hierfür so erweitert, dass jedes Mal, wenn ein Dienst zum ersten Mal auf ein Attribut zugreift, der Benutzer hierüber auf seinem Handheld informiert wird. Gleichzeitig wird ihm die Möglichkeit gegeben, dem Zugriff zuzustimmen oder es zu verweigern.

Falls der Benutzer viele Daten hat und viele personalisierbare Dienste nutzt, kann die ständige Aufforderung zur Zustimmung lästig werden. Obwohl im Zusammenhang einer gesteigerten Kontrolle ein gewisser Komfortverlust nie ganz vermieden werden kann, wurden doch einige Strategien entwickelt, um den Komfortverlust möglichst gering zu halten. Diese Strategien sind im Folgenden erläutert:

- Zunächst lohnt es sich, den obigen Aushandlungsprozess vor der Authentifizierung erneut zu betrachten. Denn im Rahmen dieses Prozesses wird dem Benutzer schon vor seiner Anmeldung angezeigt, welche Dienste welche Daten benötigen. Abhängig von diesen Informationen wählt er im Anschluss an die Anmeldung die Teilidentität, die dann dem eHome bekannt gegeben wird. Damit hat der Benutzer implizit dem zugestimmt, dass die aufgelisteten Dienste auch auf die in der Aushandlung genannten Daten zugreifen können. Diese implizite Zustimmung wird daher automatisch auf die Attribute der ausgewählten Teilidentität übertragen. Konkret bedeutet dies, dass die `Access`-Felder dieser Attribute vor der Übertragung ins eHome automatisch erweitert wird. Somit wird eine explizite Zustimmung des Benutzers später überflüssig.

- Als Nächstes wird eine explizite Nachfrage beim Benutzer für die Attribute nicht durchgeführt, die von dem anfragenden Dienst selbst erstellt wurden. Mit anderen Worten wird der Dienst, der ein Attribut für einen Benutzer im Benutzermodell anlegt, automatisch in die Zugriffskontrollliste dieses Attributs aufgenommen. Solche Attribute benutzen die Dienste oft, um ihre Funktionalitäten besser realisieren zu können. Beispielsweise verwendet der `Musikdienst` Attribute zum Speichern der aktuellen Position in der Wiedergabeliste, wenn die Bezugsperson den Raum wechselt oder die Musik unterbricht. Solche Attribute sollten die Dienste ohne Zustimmung des Benutzers nutzen dürfen.

  Diese Attribute erinnern an *Cookies*, die Webseiten im Browser des Internetbenutzers anlegen, um Benutzeraktionen sowohl während einer Sitzung als auch über mehrere Sitzungen hinweg miteinander verketten zu können. Die Verkettung innerhalb einer Sitzung ist oft nicht vermeidbar, um bestimmte Transaktionen durchführen zu können. Über mehrere Sitzungen sollten Cookies jedoch nicht verkettet werden, weil so zu viele

Informationen über die Benutzer gesammelt werden können. Gängige Browser bieten daher die Möglichkeit, Cookies nach jeder Sitzung zu löschen. Ein ähnliches Vorgehen wird auch hier praktiziert. Verlässt der Benutzer nämlich das eHome, werden die von den Diensten erstellten Attribute nicht auf das Handheld übertragen und verlieren somit ihren Nutzen. Somit wird ein möglicher Missbrauch solcher Attribute für die Wiedererkennung anonymer Benutzer unterbunden.

- Eine besondere Rolle kommt auf das Attribut zu, dass den Aufenthaltsort des Benutzers im eHome enthält. Dieses Attribut ist für einen reibungslosen Ablauf im eHome essenziell, wie auch in Abschnitt 4.3 beschrieben wurde. Es wird von einem besonderen Dienst aktualisiert, nämlich dem `Personendetektor`. Das Benutzermodell kennt diese besondere Beziehung und erlaubt dem `Personendetektor` automatisch den Zugriff auf das Attribut, ohne den Benutzer hierüber zu informieren. Falls jedoch weitere Dienste auf dieses Attribut zugreifen möchten, wird der Benutzer gefragt.

- In allen anderen Fällen muss das Benutzermodell beim ersten Zugriff eines Dienstes auf ein Attribut um die Zustimmung des Benutzers fragen. Stimmt der Benutzer diesem Zugriff zu, wird der Dienst in die Zugriffskontrollliste des Attributs aufgenommen und kann fortan ohne Einschränkung auf dieses Attribut zugreifen, bis der Benutzer seine Zustimmung möglicherweise wieder entzieht.

- Als letzte Maßnahme zum Komfortgewinn bietet es sich an, die Zugriffskontrolllisten der Attribute auch in anderen eHomes zu verwenden. Somit müsste der Benutzer nur ein Mal dem Zugriff eines Dienstes auf ein Attribut zustimmen. Beispielsweise kann der Benutzer somit angeben, dass in jedem eHome ein `Lichtdienst` auf die `Beleuchtungspräferenz` zugreifen darf. Dadurch reduziert sich der Interaktionsaufwand des Benutzers mit dem Benutzermodell.

Durch die selektive Zugriffskontrolle wird somit verhindert, dass eHome-Dienste mehr Daten aus dem Benutzermodell erlangen können, als ihnen tatsächlich zusteht. Im Folgenden wird erläutert, dass zusätzlich eine mögliche Verkettbarkeit der Daten zwischen den Diensten verhindert werden sollte.

6.2.2 Unverkettbarkeit durch Pseudonyme

Die selektive Zugriffskontrolle stellt sicher, dass jeder eHome-Dienst nur die Benutzerdaten erlangen kann, die für seine Funktionalität nötig sind. Falls sich jedoch mehrere Dienste zusammenschließen, können sie ihre Daten miteinander verketten. Dies liegt vor allem daran, dass sowohl das Benutzermodell als auch die personalisierbaren Dienste jeden Benutzer unter dem gleichen Pseudonym kennen. In dem Beispiel aus Abbildung 6.10 ist ein Benutzer dem Benutzermodell unter dem Pseudonym `Bob` bekannt[1]. Auch die dargestellten Dienste

[1] Das Pseudonym `Bob` scheint sich um den reellen Namen des Benutzers zu handeln. Dies ist jedoch nicht der Fall. In diesem Kapitel wird angenommen, dass der Benutzer sich mit einem anonymen Credentials authentifiziert hat und für die aktuelle Sitzung das *zufällig* generierte Pseudonym `Bob` erhalten. Dieses Pseudonym ist für die Dauer der aktuellen Sitzung gültig und kann nicht auf die reelle Identität des Benutzers zurückgeführt werden.

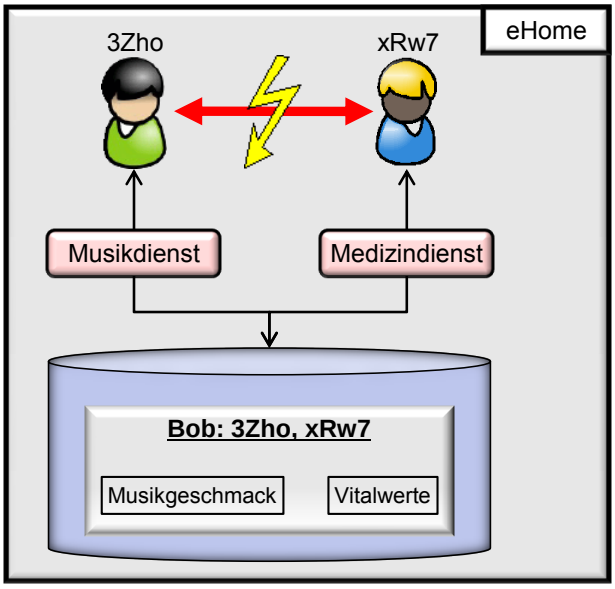


Abbildung 6.11: Unverkettbarkeit durch Pseudonyme.

kennen denselben Benutzer unter diesem Pseudonym, weil sie diesen als Referenz auf ihre Bezugspersonen erhalten, mit der sie Daten beim Benutzermodell anfragen können. Würden beide Dienste zusammenarbeiten, könnte der `Musikdienst` über Umwege an die `Vitalwerte` kommen und die selektive Zugriffskontrolle umgehen.

Dieses Problem kann dadurch gelöst werden, dass der *Informationsfluss zwischen Diensten unterbunden* wird. Um dieses Ziel zu erreichen, erteilt das Benutzermodell jedem personalisierbaren Dienst pro Benutzer, der den Dienst verwendet, ein zufällig generiertes, eindeutiges Pseudonym. Abbildung 6.11 zeigt das oben beschriebene Beispiel nun mit den unterschiedlichen Pseudonymen. Der Benutzer ist dem Benutzermodell zwar immer noch unter dem Hauptpseudonym `Bob` bekannt. Die Dienste haben hingegen unterschiedliche Pseudonyme. Der `Musikdienst` kennt denselben Benutzer nun unter dem Pseudonym `3Zho`, während der `Medizindienst` ihn unter dem Pseudonym `xRw7` kennt. Durch diese Maßnahme sind die Dienste nicht mehr in der Lage, zu erkennen, ob sie von demselben Benutzer verwendet werden oder von verschiedenen. Somit können sie ihre Daten mit denen anderer Dienste nicht mehr verketten. Darüber hinaus können die Dienste auch kein Dienstnutzungsprofil der Benutzer erstellen. Diese Informationen sind folglich nur noch dem Benutzermodell bekannt[2]. Somit besteht hier *Level-1-Anonymität* zwischen Diensten und Benutzern.

Das Benutzermodell nimmt hier somit die Rolle eines *Identitätstreuhänders* (engl. *Identity Broker*, s. [PH09]) ein. Üblicherweise verwaltet ein Identitätstreuhänder mehrere Pseudonyme

[2]Im Prinzip kann der eHomeConfigurator auch die Dienstnutzungsprofile der Benutzer ermitteln. Denn wenn sich ein Benutzer von einem Raum in einen Anderen bewegt, wird das Benutzermodell die Umkonfigurierung aller *personengebundenen* Dienste dieses Benutzers über den eHomeConfigurator anstoßen. Letzterer könnte dadurch möglicherweise erkennen, dass es sich um den gleichen Benutzer handelt, weil die Umkonfigurierung bestimmter Dienste immer zur gleichen Zeit stattfindet. Dies ist jedoch nicht weiter schlimm, weil der eHomeConfigurator als Infrastrukturdienst zur gleichen Vertrauensgruppe zugeordnet wird, wie das Benutzermodell.

eines Benutzers und kennt seine echte Identität. Somit könnte er bei Fehlverhalten der Benutzer die reelle Identität aufdecken. Weil sich die Benutzer bei den eHomes aber anonym anmelden können, trifft das für Benutzermodell in diesem Fall nicht zu. Es kennt nur die Zuordnung des Hauptpseudonyms zu den Pseudonymen, die den Diensten mitgeteilt werden.

Es sei hier noch bemerkt, dass die Funktionalität der Dienste hiervon nicht betroffen ist. Jeder personalisierbare Dienst erhält weiterhin ein Pseudonym, das er als Referenz nutzen kann, um Benutzerdaten beim Benutzermodell anzufragen. Da dieses Pseudonym für die Dauer der Benutzersitzung gültig ist, können sie ihre Funktionalitäten nach wie vor den Benutzerwünschen anpassen. Der Benutzer ist somit dem Dienst gegenüber anonym, aber während der Sitzung von anderen Benutzern unterscheidbar.

6.3 Verschlüsselung persistenter Benutzerdaten

Wie schon in Abschnitt 4.5.1 erwähnt, werden Benutzerdaten im Datenspeicher DataStorage persistent gespeichert. Hierfür werden die von Java angebotenen Techniken zur *Serialisierung* von Objektstrukturen verwendet. Folgende Bemerkungen beziehen sich auf mögliche Auswirkungen der persistenten Speicherung auf die Privatsphäre.

Eigentlich widerspricht die persistente Speicherung von Benutzerdaten dem Schutz der Privatsphäre. Denn je länger Benutzerdaten gespeichert werden und je öfter sich der gleiche Benutzer in einem eHome aufhält, desto höher ist die Wahrscheinlichkeit, dass die Daten über mehrere Sitzungen hinaus verknüpft werden können. Auf den ersten Blick scheint das sogar nützlich zu sein, weil das eHome ja dann sofort ohne erneute Konfigurierung seine Funktionalitäten an den Benutzer anpassen kann. Dies ist aber nur für eHomes richtig, die den Benutzer über ein *Beziehungspseudonym* kennen (s. Abschnitt 2.4.4). Solch ein Pseudonym sollte jedoch nur für eHomes angewendet werden, die wie das eigene Heim ein starkes Vertrauen des Benutzers genießen. Andernfalls sollten, wie oben vorgeschlagen, nur Pseudonyme verwendet werden, die nur für eine Sitzung gelten.

Weil im letzteren Fall ein eHome den Benutzer beim nächsten Mal nicht mehr erkennen wird, liefert die persistente Speicherung für anonym angemeldete Benutzer über eine Sitzung hinaus keinen Mehrwert. Daher wird hier der Ansatz verfolgt, dass Benutzerdaten automatisch gelöscht werden, wenn ein Benutzer das eHome verlässt. Außerdem wurde schon in Abschnitt 6.1.1.1 erwähnt, dass der IdentityManager auf dem Handheld die letzten Einstellungen des Benutzers für jedes eHome separat speichern kann. Bei der nächsten Anmeldung kann das Handheld die Aushandlung dann auf Basis dieser Einstellungen automatisch durchführen. Somit kann der Komfortverlust erheblich kompensiert werden.

In dem Fall, dass Benutzerdaten doch persistent gespeichert werden, muss der Schutz dieser Daten (Vertraulichkeit) vor Angreifern geschützt werden, die möglicherweise auf das Speichersystem das eHome-Gateways zugreifen können. Hierfür wurde die Serialisierung der Daten mit einem *Verschlüsselungsmechanismus* aus Basis von Triple-DES (s. [Uni99]) erweitert. Aus diesem Grund kann nur das Benutzermodell die gespeicherten Daten lesen. Daher können mögliche Angreifer keinen Nutzen aus den Daten ziehen. Weitere Details zum Benutzermodell können [Ros08] entnommen werden.

6.4 Sicherung der drahtlosen Kommunikation

In den Abschnitten zuvor wurde erläutert, wie die Privatsphäre von Benutzern vor eHomes oder eHome-Diensten geschützt werden kann. Eine weitere Gefahr besteht aber darin, dass Benutzerdaten während der drahtlosen Übertragung zwischen Handheld und eHome-Gateway abgegriffen werden können. In diesem kurzen Abschnitt wird erläutert, wie diese Übertragung gesichert werden kann.

Im Rahmen dieser Arbeit werden die Daten während der Übertragung *verschlüsselt*. Weil jedoch die auf den Handhelds eingesetzte Komponentenplattform eRCP bzw. JXME, JXTA für mobile Geräte, die Verschlüsselung von Daten bzw. Pipes nicht unterstützen, wurde im Rahmen dieser Arbeit hierfür eine externe Bibliothek eingebunden. Diese Bibliothek wird von *The Legion of the Bouncy Castle* bereitgestellt und bietet unterschiedliche Algorithmen zur Verschlüsselung an [Leg10].

Der erste Gedanke, die Daten vor der Übertragung mit einem asymmetrischen Verfahren zu verschlüsseln, wurde schnell verworfen. Zwar eignet sich dieses Verfahren besonders für das Mobilitätsszenario, da öffentliche Schlüssel spontan ausgetauscht werden können. Die eingesetzten Handhelds verfügen jedoch nicht über die für solch ein Verfahren benötigten Ressourcen. Daher wurde ein hybrides Verfahren angewendet. Somit werden die Daten bzw. Nachrichten mit einem symmetrischen Algorithmus, dem *AES*-Verfahren, verschlüsselt. Die dazu benötigten Schlüssel sind auf beiden Seiten gleich und müssen daher sicher ausgetauscht werden. Für den Austausch des symmetrischen Schlüssels wird daher das asymmetrische *RSA*-Verfahren eingesetzt.

Der Austausch des symmetrischen Schlüssels läuft dabei wie folgt ab: Vor der Authentifizierung schickt das eHome einen öffentlichen Schlüssel an das Handheld des Benutzers. Das Handheld wählt einen symmetrischen Schlüssel, verschlüsselt ihn mit dem öffentlichen Schlüssel des eHomes und sendet ihn zurück. Das eHome kann den symmetrischen Schlüssel mithilfe seines privaten asymmetrischen Schlüssels entschlüsseln. Somit sind beide im Besitz des symmetrischen Schlüssels und können sicher Daten und Nachrichten austauschen.

6.5 Realisierung

In diesem Abschnitt werden die wichtigsten Details zur Realisierung der zuvor vorgestellten Konzepte beschrieben. Zunächst wird erläutert, welche Rolle das Benutzermodell für die selektive Zugriffskontrolle und die Unverkettbarkeit von Benutzerdaten spielt. Anschließend wird dargelegt, wie das anonyme Credential-System idemix in den eHome-Prototyp eingebettet wurde. Hiernach wird erläutert, wie die aushandlungsbasierte Authentifizierung umgesetzt wurde. Dabei werden auch die Bausteine vorgestellt, die für diesen Zweck neu entwickelt wurden.

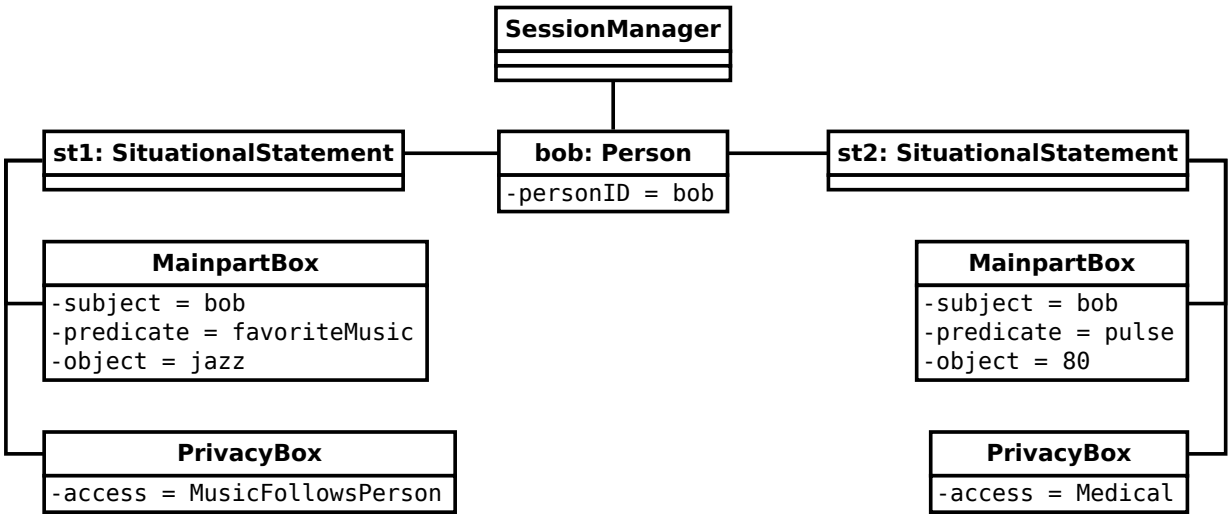


Abbildung 6.12: Realisierung von Zugriffskontrolllisten mit der PrivacyBox.

6.5.1 Selektive Zugriffskontrolle für Benutzerdaten und deren Unverkettbarkeit

Wie in Kapitel 4 beschrieben wurde, verwaltet das Benutzermodell die Daten aller eHome-Benutzer und dient als Zugriffspunkt für Produzenten und Konsumenten von Benutzerdaten. Es ist daher naheliegend, die Realisierung der selektiven Zugriffskontrolle und der Unverkettbarkeit von Benutzerdaten durch Pseudonyme dem Benutzermodell zu überlassen, wie im Folgenden beschrieben wird.

6.5.1.1 Selektive Zugriffskontrolle mithilfe von SituationalStatements

Für die Realisierung der *selektiven Zugriffskontrolle* können die in Abschnitt 4.4.3 eingeführten *SituationalStatements* eingesetzt werden. Jedes SituationalStatement hat mehrere *Schichten*, die neben der eigentlichen Aussage des jeweiligen Attributs noch zusätzliche Informationen enthalten können. Für den Schutz der Privatsphäre eignet sich dabei insbesondere die Schicht PrivacyBox (s. Abschnitt 4.5.5). Zwar bietet die PrivacyBox mehrere Attribute, die eine feingranulare Realisierung der Zugriffskontrolle unter Berücksichtigung unterschiedlicher Kriterien wie etwa Verwendungszweck oder Gültigkeitsdauer erlauben. Für das hier verfolgte Ziel der selektiven Zugriffskontrolle reicht das Attribut `access` jedoch aus.

In Abbildung 6.12 ist noch einmal das Beispiel aus Abschnitt 4.5.6 dargestellt, das zwei Identitätsattribute des Benutzers Bob aus dem Benutzermodell zeigt. Diesmal ist für jedes einzelne SituationalStatement zusätzlich zur MainPartBox auch die zugehörige PrivacyBox dargestellt. Für jede PrivacyBox ist die Variable `access` gezeigt, die die jeweilige Zugriffskontrollliste repräsentiert. Diese Liste enthält den Namen aller Dienste, die auf das zugehörige SituationalStatement zugreifen dürfen. Demnach kann der Dienst `MusicFollowsPerson` auf das linke SituationalStatement zugreifen, das die Musikpräferenz von Bob enthält. Auf

Token	Hauptpseudonym	Dienst
3Zho	bob	Musikdienst
xRw7	bob	Medizindienst

Tabelle 6.1: Auszug aus einer Liste von Tokens.

das rechte SituationalStatement, das den aktuellen Puls von Bob enthält, darf hingegen nur der Dienst `Medical` zugreifen. Dem Dienst `MusicFollowsPerson` würde demnach der Zugriff auf den Puls verweigert, weil er nicht in der Zugriffskontrollliste enthalten ist. Unter welchen Bedingungen ein Dienst auf diese Liste aufgenommen wird, wurde in Abschnitt 6.2.1 diskutiert.

6.5.1.2 Pseudonymisierung mit Tokens

Um die Verkettbarkeit von Benutzerdaten durch Dienste zu vermeiden, wurde das Benutzermodell so erweitert, dass Pseudonyme durch sogenannte *Tokens* realisiert werden. Es handelt sich hierbei um eine Art Marke, die der SessionManager (s. Abschnitt 4.5.3) erstellt und personalisierbaren Diensten aushändigt. Diese Dienste müssen fortan ein gültiges Token vorlegen, wenn sie auf Benutzerdaten zugreifen möchten.

Um die Rolle von Tokens im Benutzermodell zu verdeutlichen, wird das Beispiel aus Abbildung 4.9 erneut aufgegriffen. Wie das Sequenzdiagramm aus dieser Abbildung illustriert, musste dort ein Dienst den SessionManager mit der Methode `getStmnt()` aufrufen, um Benutzerdaten anzufragen. Hierfür musste er als Aktualparameter den Namen des Benutzers und dem Namen des gewünschten Identitätsattributs aufrufen. Der Name des Benutzers entsprach dabei dem *Hauptpseudonym* des Benutzers im Benutzermodell.

Dies ändert sich nun durch den Einsatz von Tokens. Von nun an erhält jeder Dienst für jeden Benutzer, der ihn nutzt, ein eigenes (zufällig generiertes) Token. Die Dienste selbst müssen hierfür nicht angepasst werden, weil Tokens genauso wie Hauptpseudonyme als Strings realisiert sind. Der SessionManager verwaltet intern die Liste aller herausgegebenen Tokens und stellt sicher, dass kein Token doppelt ausgestellt wird. Tabelle 6.1 zeigt solch eine Liste für das Beispiel aus Abbildung 6.11. Aus dieser Liste kann der SessionManager ermitteln, zu welchem Benutzer ein Token gehört und welcher Dienst das Token erhalten hat. Dadurch wird sichergestellt, dass Dienste ihre Tokens untereinander nicht weitergeben können. Bei Vorlage des Tokens kann der SessionManager somit (über spezielle Java-Mechanismen) überprüfen, ob es sich um einen berechtigten Dienst handelt.

Abbildung 6.13 zeigt nun ein erweitertes Sequenzdiagramm für das oben genannte Beispiel. Diesmal sind sowohl die selektive Zugriffskontrolle als auch Tokens berücksichtigt. Die Ziffern in den Kreisen markieren besondere Stellen in diesem Zusammenhang. Zunächst fällt auf, dass der Dienst die Methode `getStmnt()` nicht mehr mit dem Hauptpseudonym des Benutzers aufruft, sondern mit seinem Token `3Zho` (1). Über dieses Token ermittelt der SessionManager zunächst, welchem Benutzer dieses Token zugeordnet ist und ob der Dienst dieses Token tatsächlich erhalten hat (2). Nachdem der Benutzer ermittelt wurde, werden die übrigen Schritte mit Bezug auf sein Hauptpseudonym ausgeführt. Eine wichtige Neuigkeit ist, dass eine Referenz auf den aufrufenden Dienst bis auf das zugehörige

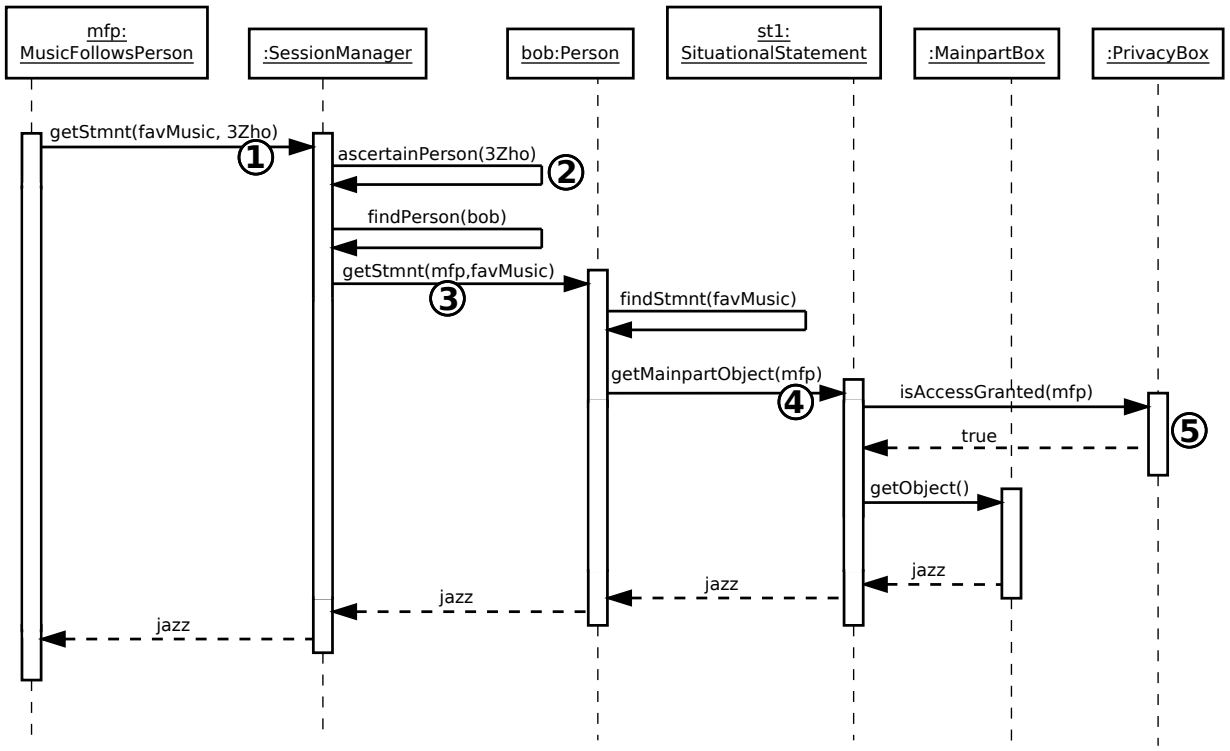


Abbildung 6.13: Möglicher Ablauf einer Abfrage von Benutzerdaten mit einem Token am Beispiel des `Musikdienstes`.

SituationalStatement weitergereicht wird (3,4). Dies ist nötig, weil jedes SituationalStatement seine eigene Zugriffskontrollliste hat und dort überprüft werden muss, ob der Dienst auf die Informationen zugreifen darf. Hier kommt die PrivacyBox mit der Methode `isAccessGranted()` ins Spiel (5). Erst dann, wenn diese Methode den Wert `true` liefert, erhält der Dienst die geforderte Information.

Es sei an dieser Stelle bemerkt, dass das Benutzermodell bei der Erstellung von Tokens die Typen personengebundener Dienste berücksichtigt (s. Abschnitt 4.3). Abhängig von diesem Typ kann im Rahmen der Intra-eHome-Mobilität die Art der Handhabung von Tokens unterschiedlich ausfallen:

- Im einfachsten Fall handelt es sich um einen *personengebundenen* Dienst, weil jede Instanz eines solchen Dienstes in seiner Laufzeit genau für eine Bezugsperson zuständig ist. Für jede Instanz eines personengebundenen Dienstes erstellt das Benutzermodell daher genau ein Token für die zugehörige Bezugsperson. Dieses Token bleibt dann für die gesamte Sitzungsdauer des Benutzers unverändert.

- *Umgebungsgebundene* Dienste hingegen können zur Laufzeit von mehreren Benutzern verwendet werden. Daher erstellt das Benutzermodell für jeden Benutzer, der zur Laufzeit hinzukommt, ein neues Token und teil es dem Dienst mit. Umgekehrt wird das zugehörige Token dem Dienst wieder entzogen, wenn ein Benutzer das eHome verlässt

oder den Dienst nicht mehr benutzen möchte. Dadurch weiß der Dienst stets für wie viele Benutzer er zuständig ist und mit welchen Tokens er auf die zugehörigen Daten zugreifen kann. Dabei musste keine Anpassung an diesen Diensten vorgenommen werden, weil umgebungsgebundene Dienste bereits Methoden für das Hinzufügen oder Entfernen von Benutzern mitbringen. Durch Aufruf der `addUser(String username)` kann das Benutzermodell einen neuen Benutzer ankündigen und durch Aufruf von `removeUser(String username)` dem Dienst mitteilen, dass ein Benutzer nicht mehr berücksichtigt werden muss. Diese Methoden können nun unverändert für Tokens verwendet werden, weil diese auch den Typ String haben.

- Eine Besonderheit weisen in diesem Zusammenhang *raumgebundene* Dienste auf. Da jede Instanz eines solchen Dienstes für seinen Raum zuständig ist, nutzt das Benutzermodell die gleichen Methoden wie bei umgebungsgebundenen Diensten, um den Dienst bezüglich der Benutzer, die seinen Raum betreten oder verlassen, auf dem aktuellen Stand zu halten. Die Besonderheit liegt nun darin, dass ein Token nicht wiederverwendet wird, wenn ein Benutzer seinen Aufenthaltsraum wechselt. Stattdessen erhält die nächste Instanz des gleichen Dienstes im neuen Raum ein neues Token. Somit können mehrere Instanzen desselben Dienstes nicht erkennen, in welchen Räumen sich der Benutzer zuvor aufgehalten hat und ein Mobilitätsprofil erstellen[3].

In Abschnitt 4.5.3.2 wurde erläutert, dass der SessionManager ermöglicht, dass sich ein Konsument als Beobachter für bestimmte Benutzerdaten registrieren kann. Auch in diesem Fall muss der Konsument sein Token verwenden. Im obigen Beispiel kann der Dienst sich mit seinem Token registrieren, um über Änderungen des Musikgeschmacks benachrichtigt zu werden.

Insgesamt ist somit sichergestellt, dass einerseits eHome-Dienste nur dann Informationen aus dem Benutzermodell erhalten, wenn sie dazu befugt sind. Andererseits wird durch den Einsatz von Tokens die Verkettbarkeit von Benutzerdaten außerhalb des Benutzermodells unterbunden.

6.5.1.3 Weitere Anmerkungen zum Benutzermodell

In Abschnitt 6.1.1.1 wurde erwähnt, dass der IdentityManager auf dem Handheld die Ergebnisse der Aushandlung von Benutzerdaten speichert, um sie beim nächsten Betreten desselben eHomes automatisch auszuwählen. Dadurch muss der Benutzer die Aushandlung nicht erneut durchlaufen. Um dies zu realisieren, wurde erneut die Datenstruktur SituationalStatement verwendet: Der IdentityManager trägt für alle Benutzerdaten, die dem eHome zur Verfügung gestellt werden sollen, den Namen des eHomes im `location`-Attribut der Schicht situation des zugehörigen SituationalStatements ein. Beim nächsten Anmelden am selben eHome können diese Benutzerdaten dem eHome automatisch zur Verfügung gestellt wurden. Das `location`-Attribut kann dabei eine Liste von eHomes enthalten, weil einige Benutzerdaten in mehreren eHomes benötigt werden können.

[3]Natürlich gilt dies nur dann, wenn die für diesen Dienst zugänglichen Benutzerdaten keine eindeutigen Unterscheidungsmerkmale für diesen Benutzer aufweisen.

In Abschnitt 6.2.1 wurde erläutert, dass es wichtig ist, zu unterscheiden, woher Benutzerdaten stammen. Dabei wurden unterschiedliche *Quellen* von Benutzerdaten identifiziert: Sie können vom Handheld des Benutzers stammen oder von personalisierbaren Diensten bzw. vom Personendetektor erstellt worden sein. Sowohl die Interaktion mit dem Benutzer für die Zugriffskontrolle auf die Daten als auch die Synchronisierung der Daten zwischen eHomes und Handhelds werden von diesen Information beeinflusst. Um diese Informationen im Benutzermodell geeignet abzulegen, wurde das Attribut `source` der Schicht `source` des zugehörigen SituationalStatements verwendet. Für die Präferenz der Beleuchtung würde der Eintrag beispielsweise die Form `subject=user` haben, weil diese Präferenz vom Benutzer stammt. Für die aktuelle Position in der Wiedergabeliste hingegen würde der Eintrag `subject=MusicFollowsPerson` darauf hindeuten, dass diese Information vom `MusicFollowsPerson`-Dienst stammt. Die zugehörigen SituationalStatements der Aufenthaltsorte der Benutzer würden alle den Eintrag `subject=Personendetektor` enthalten, weil diese Information stets vom `Personendetektor` aktualisiert wird.

6.5.2 Integration von idemix in den eHome-Prototyp

In diesem Abschnitt wird die Integration des anonymen Credential-Systems *idemix* in den eHome-Prototyp beschrieben. IBM stellte dem eHome-Projekt den idemix-Quellcode freundlicherweise vollständig zur Verfügung. Dadurch konnte idemix als *Bibliothek* eingebunden werden. Die Integration als Bibliothek ermöglicht den flexiblen Einsatz von idemix für eigene Zwecke. Ferner kann die Komplexität des Credential-Systems so vor dem Benutzer verborgen werden. Dabei mussten kaum Änderungen an dem idemix-Quellcode vorgenommen werden. Es mussten lediglich an einigen wenigen Stellen leichte Modifikationen vorgenommen werden, um die Kompatibilität mit der auf den Handhelds eingesetzten eRCP (s. Abschnitt 2.1.4) zu gewährleisten. Außerdem wurde die Serialisierung vom SimpleRMI (s. Abschnitt 5.3.3) so erweitert, dass auch idemix-spezifische Objekte, beispielsweise Credentials, zwischen Handheld und eHomes ausgetauscht werden können.

Im Folgenden wird zunächst erläutert, welche Eingaben für die Initialisierung von idemix sowie für die Ausstellung und Verifikation von Credentials benötigt werden. Anschließend werden die Klassen beschrieben, die für die Anbindung von idemix an den eHome-Prototyp entwickelt wurden. Dabei wird nicht auf die theoretischen Details der idemix-Protokolle eingegangen, diese finden sich beispielsweise in [CH02].

6.5.2.1 Initialisieren von idemix

Bevor mit idemix Credentials und Beweise erstellt werden können, muss das System initialisiert werden. Hierfür müssen zunächst sogenannte *Gruppenparameter* bestimmt werden, die die algebraische Gruppe bestimmen, in der Operationen von idemix durchgeführt werden. Anschließend müssen die Schlüsselerstellung sowie das Ausstellen und Verifizieren der Credentials mit den gleichen Gruppenparametern durchgeführt werden. Daher müssen alle teilnehmenden Parteien die gleichen Parameter benutzen. Seine Gruppenparameter kann ein eHome mobilen Benutzern mitteilen, wenn er das eHome betritt. Allerdings müssen auch

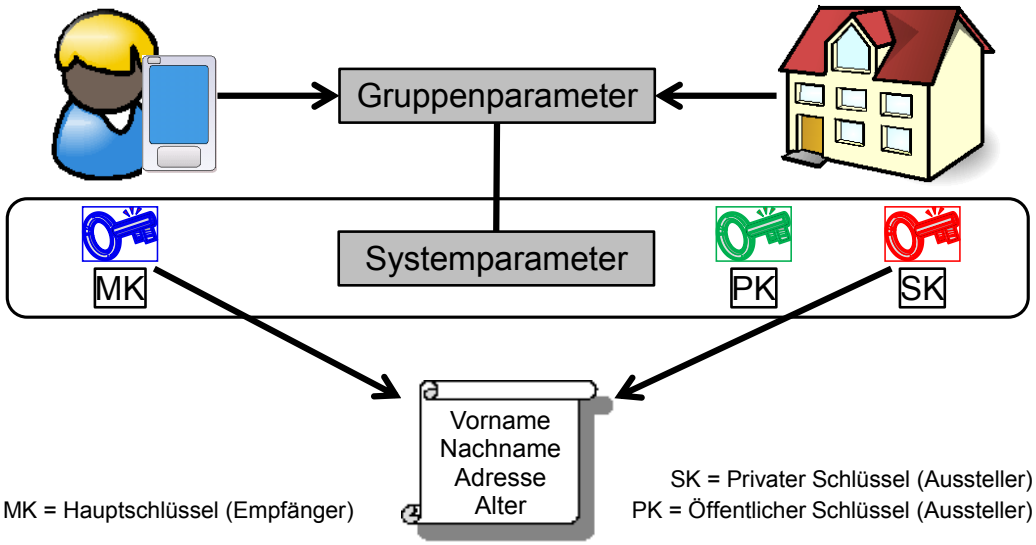


Abbildung 6.14: Zusammenhang von Parametern, Schlüsseln und Credentials.

die Parameter beim Ausstellen und Verifizieren eines TTP-Credentials gleich sein. Dazu ist es notwendig, dass der externe Aussteller und das eHome die Parameter synchronisieren, beispielsweise über öffentliche Listen.

Die algebraische Gruppe ist abhängig von bestimmten Rahmenbedingungen, welche die kryptografische Stärke von Credentials bestimmen. Diese Rahmenbedingungen werden durch die *Systemparameter* repräsentiert. Die Systemparameter legen die Schlüssellänge, die Wahrscheinlichkeitsverteilung des Primzahlengenerators und einige andere Eigenschaften fest. Auch die Systemparameter müssen bei allen Parteien übereinstimmen.

Sobald die Gruppenparameter erzeugt und verteilt sind, können Schlüsselpaare erstellt werden. Es werden verschiedene Schlüssel benötigt. Der *Aussteller* eines Credentials muss ein Schlüsselpaar aus privatem (engl. *Secret Key* (SK)) und öffentlichem (engl. *Public Key* (PB)) Schlüssel besitzen. Dieses Paar muss dann über die komplette Verwendungszeit der damit ausgestellten Credentials erhalten bleiben. Bei der Erstellung dieses Schlüsselpaares muss festgelegt werden, wie viele Attribute in einem Credential maximal enthalten seinen können. Das Zusammenspiel von Parametern, Schlüssel und Credentials wird in Abbildung 6.14 gezeigt.

Jeder Empfänger hingegen benötigt einen privaten *Hauptschlüssel*, mit dem alle seine Pseudonyme verknüpft werden. Der Hauptschlüssel wird benutzt, um Credentials zu beziehen und mit ihnen Nachweise zu führen.

6.5.2.2 Ausstellen von Credentials

Wie ein idemix-Credential konzeptionell ausgestellt wird, wurde schon in Abschnitt 2.8.3.3 beschrieben. Nun wird dieser Prozess aus der Sicht des Entwicklers erläutert. Ein Überblick über die dafür benötigten Daten ist in Abbildung 6.15 dargestellt. Auf der linken Seite ist der Empfänger und auf der rechten Seite der Aussteller des Credentials mit den zugehörigen Eingaben abgebildet.

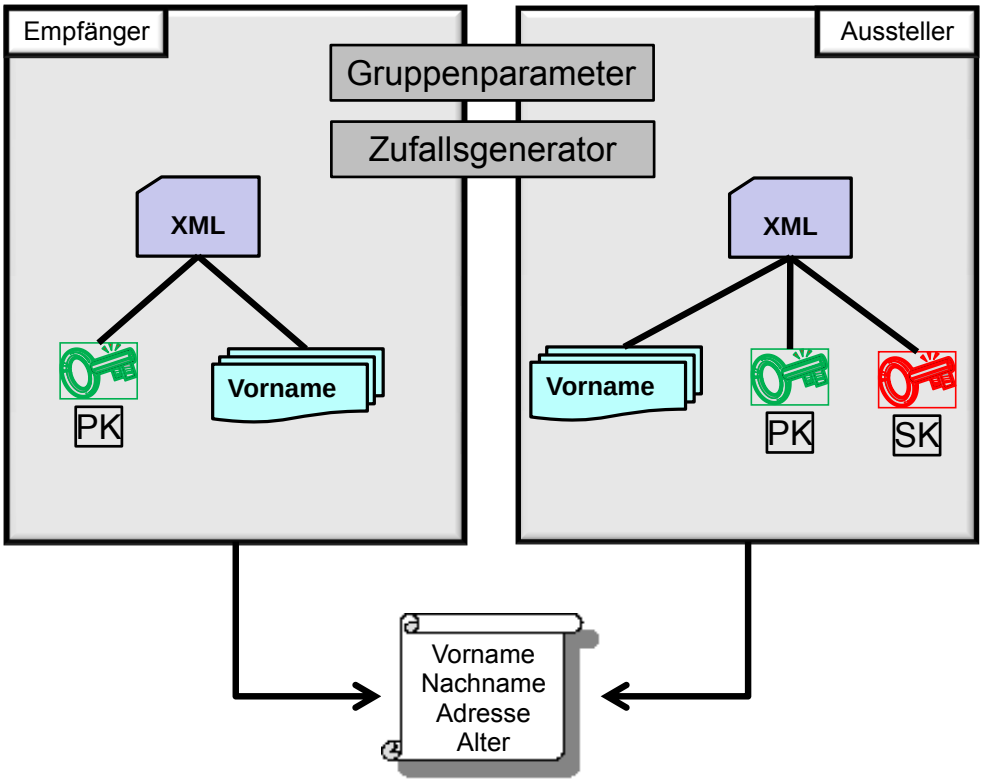


Abbildung 6.15: Eingaben von Empfänger und Aussteller eines Credentials.

Für die Repräsentation des Empfängers bzw. des Ausstellers sieht idemix die Klassen IssueCertificateRecipient bzw. IssueCertificateIssuer vor. Beide benötigen die Gruppenparameter und einen Zufallsgenerator. Außerdem brauchen sie ein XML-Dokument, das das auszustellende Credential beschreibt. Der IssueCertificateRecipient erhält als weitere Eingabe den privaten Hauptschlüssel. Von diesen Eingaben wird im Folgenden das XML-Dokument näher betrachtet, die übrigen Argumente ändern sich nach ihrer Erzeugung nicht mehr. Das XML-Dokument, dessen Spezifikation in [IBM08] eingesehen werden kann, ist nach einem festen Schema aufgebaut und besteht aus zwei Teilen.

Im *oberen Teil* wird festgelegt, welchen Inhalt das auszustellende Credential haben soll. Dazu dient das `Statements`-Element. Es beinhaltet beliebig viele `Predicate`-Elemente. Jedes Predicate-Element steht für ein Attribut, das im Credential verankert werden soll. Über die `PredicateID` wird der Typ des Attributs festgelegt. In Listing 6.1, das einen Auszug aus solch einer Date zeigt, hat der Nachname beispielsweise den Typ `String`. Der Eintrag `Ontology` legt den Namen des Attributs fest, hier `Nachname`. Der Wert des Attributs folgt in einem `Constant`-Element. Er wird in diesem Fall im Klartext in das XML-Dokument geschrieben, da beide Parteien den Wert kennen.

Im *unteren Teil* des XML-Dokumentes wird die Spezifikation des Credentials festgehalten. Die oben deklarierten Attribute werden noch einmal mit ihrem Typ, aber ohne Wert aufgeführt. Anschließend folgen Schlüssel und die Systemparameter. Natürlich ist nur der öffentliche Schlüssel des Ausstellers in der Spezifikation des Empfängers vorhanden. Daraus

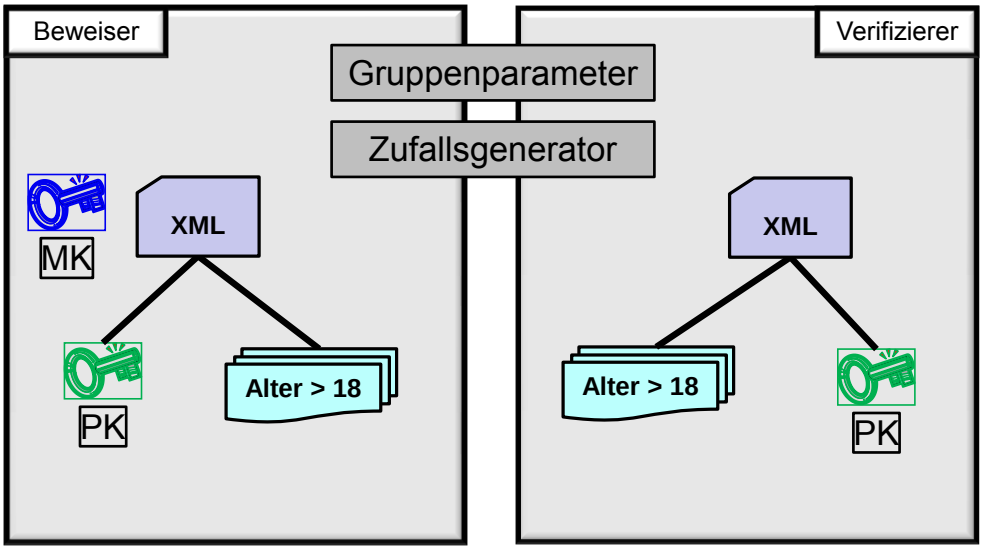


Abbildung 6.16: Eingaben von Beweiser und Verifizierer eines Credentials.

resultiert, dass zwei Versionen des Dokumentes existieren. Der Unterschied besteht aber lediglich darin, dass im XML-Dokument des Ausstellers sein privater Schlüssel enthalten ist.

Um den Anwender bei der Erstellung eines solchen XML-Dokuments zu unterstützen, wurde der SpecOutputter entwickelt, der automatisch ein spezifikationskonformes Dokument erzeugen kann. Dazu nimmt er eine Liste von Attributen entgegen. Für jedes dieser Attribute erzeugt er ein Predicate-Element und verankert es in dem XML-Dokument. Die Systemparameter und Schlüssel werden in die vorgesehenen Stellen eingefügt. Der SpecOutputter sorgt auch dafür, dass die privaten Schlüssel nur in den Dokumenten auftauchen, in denen sie enthalten sein sollen.

Das eigentliche Protokoll zur Ausstellung eines Credentials wird in drei Runden zwischen IssueCertificateIssuer und IssueCertificateRecipient ausgeführt. Dazu stehen Methoden bereit, die mit dem Ergebnis der jeweiligen Gegenseite aufgerufen werden. Am Ende des Ablaufes erhält der Empfänger ein weiteres XML-Dokument. Es stellt das Credential dar und kann gespeichert werden. Darin enthalten sind die vereinbarten Attribute in verschlüsselter Form und Werte, mit denen ein Beweis berechnet werden kann.

Ein solches Credential ist in Listing 6.3 dargestellt. Dabei ist erkennbar, dass das Credential keine Rückschlüsse auf die Typen der enthaltenen Attribute zulässt. Die Typen werden aber benötigt, um später einen Nachweis aus dem Credentials zu erstellen. Aus diesem Grund wird das Credential um eine Struktur erweitert, die die Typen identifiziert. Diese Aufgabe übernimmt der IdemixHandler. Dieser wird weiter unten beschrieben und bildet die Schnittstelle zum idemix-System. Durch das XML-Dokument zum Ausstellen des Credentials sind die Typen der Attribute noch verfügbar. Die Typen werden ausgelesen und in Form eines, in Listing 6.2 gezeigten, XML-Statements an das Credential angehängt. Die Typen werden durch Zahlen repräsentiert. Eine 0 steht beispielsweise für den Typ Integer, eine 1 für String und eine 2 für Date.

```
1 <Statements>
2   <Connective>
3     <Predicate PredicateId=
4       "http://se.rwth-aachen.de/idemix.xsdString-eq">
5        <Attribute Ontology="Nachname" IssuanceMode="userchosen"
6          CertificateSpecificationReference="#certSpec1" />
7        <Constant>Armac</Constant>
8     </Predicate>
9   </Connective>
10 </Statements>
```

Listing 6.1: Beispielattribut aus einem XML-Dokument für die Erstellung eines Credentials.

```
1   <AttributeTypes>11</AttributeTypes>
```

Listing 6.2: Angabe von Attributtypen für Credentials.

6.5.2.3 Verifizieren von Credentials

Nachdem der Empfänger sein Credential erhalten hat, können nun Nachweise über die enthaltenen Attribute berechnet werden (Vorzeigen des Credentials). Ein solcher Nachweis wird von dem Besitzer (ProveProver) berechnet und von dem Verifizierer (ProveVerifier) verifiziert. Aus Entwicklersicht läuft das Vorzeigen ähnlich zum Ausstellen ab, wie in Abbildung 6.16 dargestellt ist. Den Unterschied stellt lediglich das eingehende XML-Dokument dar. Es besitzt das gleiche Schema wie beim Ausstellen, der Inhalt ist jedoch anders. Der oben beschriebene Aufbau bleibt auch erhalten. Die Predicate-Elemente bestimmen nun, welche Attribute und in welcher Art sie bewiesen werden sollen. Es muss nicht immer die bloße Übereinstimmung nachgewiesen werden. Möglich sind auch Aussagen über ein Attribut. Wenn das Attribut aussagt, dass eine Person 25 Jahre alt ist, kann mit diesem Attribut auch bewiesen werden, dass die Person über 18 Jahre alt ist, ohne das tatsächliche Alter zu offenbaren.

Zusätzlich enthält das XML-Dokument Informationen über das verwendete Credential. Diese werden innerhalb des `Certificate`-Elements eingebettet. Zu erkennen ist dies im Listing 6.3. Zu den Informationen des Credentials gehören die verwendeten mathematischen Elemente, gekennzeichnet als `CertificateElement`. Außerdem gehören dazu alle enthaltenen Attribute. In Listing 6.3 sind die beiden Attribute *Nachname* und *Alter* zu erkennen. Sichtbar ist dort auch, dass deren Werte nicht mehr in Klartext, sondern verschlüsselt enthalten sind. Bei dem Dokument des Besitzers sind alle genannten Werte vorhanden. Im Dokument des Verifizierers sind diese Werte mit `0` angegeben. Dadurch werden die Werte vor dem Verifizierer geheim gehalten. Darin besteht der einzige Unterschied zwischen den Dokumenten der beiden Parteien.

Der Nachweis wird also vom Besitzer des Credentials berechnet. Die dazu notwendigen Schritte erfolgen innerhalb des idemix-Systems. Das notwendige Dokument wird von dem im Rahmen dieser Arbeit entwickelten AssertOutputter erstellt. Dieser ähnelt im Aufbau dem SpecOutputter. Er übernimmt nun allerdings auch das Credential und bettet es an der

```
1 <Certificate Id="26a3876e-6c02-4f29-953f-54ef1bfa9069"
2  CertificateSpecificationReference="certSpec1">
3   <CertificateElement Type="APInteger" Name="e"
4    Value="2593447..." />
5   <CertificateElement Type="APInteger" Name="capA"
6    Value="129694..." />
7   <CertificateElement Type="APInteger" Name="v"
8    Value="703039..." />
9   <Attribute Ontology="Nachname" CertifiedValue="149229..." />
10  <Attribute Ontology="Alter" CertifiedValue="417142..." />
11 </Certificate>
```

Listing 6.3: idemix-Credential als XML-Dokument.

vorgesehenen Position in das XML-Dokument ein. Der fertige Nachweis wird zusammen mit dem passenden XML-Dokument an den Verifizierer weitergereicht. Dieser ist damit in der Lage, den Nachweis zu verifizieren. Die Verifizierung findet ebenfalls vollständig innerhalb des idemix-Systems statt. Die für die Anbindung an idemix jeweils nötigen Klassen und Methoden werden in den folgenden Abschnitten beschrieben.

6.5.2.4 IdemixAuthenticator

Nachdem in den vorhergehenden Abschnitten erläutert wurde, wie mit idemix Credentials erstellt und vorgezeigt werden können, werden in diesem und im nächsten Abschnitt zwei Klassen beschrieben, die für die Anbindung von idemix entwickelt wurden. Beide Klassen sind in Abbildung 6.17 dargestellt. Während der IdemixHandler die idemix-bezogenen Aufgaben auf dem Handheld ausführt, ist der IdemixAuthenticator im eHome aktiv. Letzterer bildet das Kernstück der Authentifizierung. Er stellt alle Methoden für die Benutzung von Credentials zur Verfügung. Dazu gehören insbesondere das Ausstellen und Verifizieren von Credentials. Er kapselt die gesamte Anbindung von idemix an den eHome-Prototyp.

Im IdemixAuthenticator wird idemix initialisiert. Dazu werden die Gruppen- und Systemparameter erstellt und zum Abruf bereitgestellt. Jedes eHome ist bei der Gestaltung der Systemparameter für Session-Credentials frei. Es müssen auch nicht die gleichen Systemparameter wie bei den TTP-Credentials verwendet werden. Dies ist möglich, da jedes eHome mit den Session-Credentials ein geschlossenes System darstellt. Bei den TTP-Credentials müssen sich das eHome und der externe Aussteller allerdings auf die gleichen Systemparameter einigen. Neben den Parametern ist der öffentliche Schlüssel des eHomes ein wichtiger Bestandteil bei der Verwendung des idemix-Systems. Der Schlüssel wird benötigt, um Credentials auszustellen und zu verifizieren. Der Schlüssel wird im IdemixAuthenticator verwaltet und ist dort ebenfalls abrufbar.

Die im laufenden Betrieb wichtigsten Methoden sind solche zum Verifizieren und Ausstellen von Credentials. Zum Ausstellen eines Session-Credentials dient die Methode `prepareIssueCredential` (s. Listing 6.4). Als Argument nimmt die Methode eine Liste von Eigenschaften auf, die dem Benutzer attestiert werden sollen. Als Erstes wird die Liste der Eigenschaften in ein Format konvertiert, das vom SpecOutputter verarbeitet werden kann.

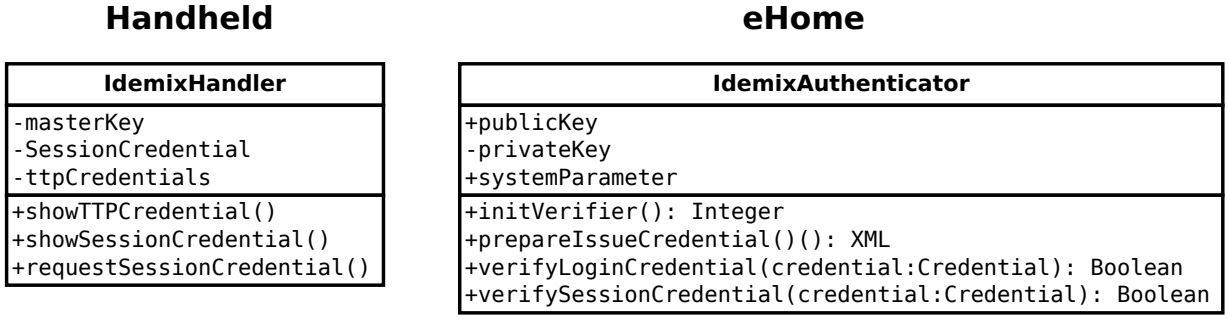


Abbildung 6.17: IdemixHandhler und IdemixAuthenticator realisieren die Anbindung von idemix an den eHome-Prototyp.

Der SpecOutputter erzeugt ein neues XML-Dokument für den Aussteller des Credentials. Im nächsten Schritt (Zeile 18) wird die Repräsentation des Ausstellers im idemix-System initialisiert. Schließlich erstellt der SpecOutputter ein neues XML-Dokument (Zeile 21ff.) ohne den privaten Schlüssel, da dieses an den Empfänger des Credentials zurück geliefert wird. Anhand der XML-Dokumente wird spezifiziert, wie und mit welchem Inhalt das Credential auszustellen ist.

In diesem Ablauf wird noch kein Credential erstellt. Der Rückgabewert dieser Methode erlaubt dem Aufrufer, also dem Empfänger, ebenfalls sein idemix-System zu initialisieren. Die eigentliche Berechnung des Credentials erfolgt *verteilt und interaktiv* mit dem Empfänger über drei Runden. Hierfür werden entsprechende Methoden des IdemixAuthenticators aufgerufen. Diese leiten die Übergabewerte an den IssueCertificateIssuer weiter und geben dessen Ergebnis zurück. Einzelheiten hierzu folgen weiter unten.

Der IdemixAuthenticator ist zwar teilweise an der Berechnung des Credentials beteiligt, er bekommt aber keine Kenntnis von dessen endgültiger Version. Auch wenn er das Credential verifiziert, überprüft er nur die Korrektheit eines auf Basis des Credentials ausgestellten Beweises. Verifiziert wird ein Beweis in der Methode `verifySessionCredential` (s. Listing 6.5). Das übergebene Argument ist vom allgemeinen Typ `Object` und wird zum Typ ProofContainer konvertiert. Dieser Datentyp wurde der idemix-Bibliothek hinzugefügt. Er fasst lediglich einen idemix-Beweis und ein XML-Dokument als String-Repräsentation zusammen. Dadurch lassen sich diese gemeinsam über SimpleRMI übertragen. Anschließend werden potenzielle Fehlerquellen ausgeschlossen. Ab Zeile 15 wird der Beweis extrahiert und an das idemix-System übergeben. Wenn die Berechnungen korrekt waren, antwortet die Methode mit einer positiven Antwort, andernfalls mit einer negativen. Sollte während des Vorgangs ein Fehler auftreten, wird natürlich eine negative Antwort zurückgegeben.

Auffällig in diesem Ablauf ist, dass die Initialisierung der idemix-Repräsentation des Verifizierers (ProveVerifier) in dieser Methode nicht vorgenommen wird. Wie der nächste Abschnitt beschreibt, benötigt der Inhaber des Credentials einen speziellen Wert vom ProveVerifier. Bei diesem Wert handelt es sich um den sogenannten *Nonce*-Wert. Dieser Wert dient dazu, Replay- und Man-in-the-Middle-Angriffe zu verhindern. Er variiert bei jeder Verifizierung eines Credentials und muss daher auch beim Beweis Berücksichtigung

```
1 private PublicParameters publicParams;
2 private SecureRandom random;
3 private IssueCertificateIssuer certIssuer;
4 public String prepareIssueCredential(
5  HashMap<String, String> predicates) {
6   Vector<Predicate> preds
7    = negotiateSpecification(predicates);
8   SpecOutputter outputter = new SpecOutputter();
9   // create xml including private key
10  String xml = outputter.createIssueSpec(
11              sessionKeyPair.getPublicKey(),
12              sessionKeyPair.getPrivateKey(),
13              publicParams, preds);
14  ByteArrayInputStream inStream
15   = new ByteArrayInputStream(xml.getBytes());
16  try {
17    // init idemix issuer
18    certIssuer = new IssueCertificateIssuer(
19              inStream, publicParams, random);
20    // create xml excluding private key for IssueCertificateRecipient
21    return outputter.createIssueSpec(
22              sessionKeyPair.getPublicKey(),
23              null, publicParams, preds);
24  } catch (Exception e) {
25    return null;
26  }
27 }
```

Listing 6.4: Ausstellen eines Credentials.

finden. Bevor also ein Beweis verifiziert werden kann, wird der Verifizierer initialisiert. Dazu dient die Methode `initVerifier`, gezeigt in Listing 6.6. Sie liefert als Rückgabewert den Nonce-Wert. Erst dann wird der Beweis berechnet und die Verifizierung angestoßen.

6.5.2.5 IdemixHandler

Der IdemixHandler ist für die Anbindung von idemix an den Prototyp für Handhelds zuständig (s. Abbildung 6.17). Über diese Klassen können Credentials angefordert und Nachweise berechnet werden. Diese Klasse ist auch für die Verwaltung der Parameter des idemix-Systems und der Schlüssel des Benutzers zuständig.

Zur Anforderung eines neuen Session-Credentials bietet der IdemixHandler die Methode `requestSessionCredential`. Wie in Listing 6.7 zu erkennen ist, bekommt die Methode eine Referenz auf den Authenticator und eine Liste von Dienstreferenzen. Diese Dienstreferenzen stehen für alle Dienste, die der Benutzer zuvor im Rahmen der Authentifizierung ausgehandelt hatte. Da das Session-Credential den Zugriff auf diese Dienste erlauben soll, werden diese Referenzen im Credential verankert (Zeilen 8ff.)[4].

[4]Es handelt sich hier also um ein *dienstbasiertes Session-Credential* (s. Abschnitt 6.1.2.3)

```
1 public boolean verifySessionCredential(Object proofC) {
2   // cast to ProofContainer
3   ProofContainer proofContainer = (ProofContainer) proofC;
4   if (proofContainer == null) {
5     return false;
6   }
7   try {
8     if (proofContainer.getProofBytes() == null
9       & proofContainer.getAssertion()
10        .equals("abording")) {
11      // there is no credential for service, do nothing
12      return false;
13    }
14    // extract assertion and proof
15    CryptographicProof cproof
16     = new CryptographicProof(
17        proofContainer.getProofBytes());
18    // perform check
19    verifier.round1(cproof);
20    if (verifier.isSuccess()) {
21      return true;
22    } else {
23      return false;
24    }
25  } catch (Exception e) {
26    return false;
27  }
28 }
```

Listing 6.5: Verifikation eines Credentials.

```
1 private ProveVerifier verifier;
2 public APInteger initVerifier(final String assertion) {
3   ByteArrayInputStream inStream
4    = new ByteArrayInputStream(assertion.getBytes());
5   try {
6     verifier = new ProveVerifier(
7             inStream, publicParams, random);
8     return verifier.getNonce();
9   } catch (Exception e) {
10    return null;
11  }
12 }
```

Listing 6.6: Initialisieren des Verifizierers

Der Authenticator selbst ist für mehrere Aufgaben verantwortlich und sollte nicht mit dem IdemixAuthenticator verwechselt werden. Beispielsweise ist der Authenticator für die Zugriffskontrolle auf Dienste zuständig, wie im nächsten Kapitel beschrieben wird. Auch die Authentifizierung von Benutzern findet über diesen statt. Erst wenn sich ein Benutzer anonym über idemix authentifizieren möchte, wird die Anfrage weiter an den IdemixAuthenticator weitergeleitet. Dies passiert beispielsweise in Zeile 17, wo der Authenticator über die Methode `prepareIssueCredential()` angesprochen wird. Diese Anfrage leitet der Authenticator dann direkt an den IdemixAuthenticator weiter. Als Rückgabewert erhält der IdemixHandler ein XML-Dokument (`spec`), das die Namen aller ausgehandelten Dienste als Attribute enthält. Der IdemixHandler kann nicht mehr Dienste in das Credential festschreiben lassen, als der IdemixAuthenticator bestätigt hat. Dieses Dokument ist zwar manipulierbar, aber der Aussteller würde eine Erweiterung der Liste um weitere Dienstnamen erkennen. Damit würde die Ausstellung des Session-Credentials fehlschlagen, was durch idemix garantiert wird. Die endgültige Entscheidung über die nutzbaren Dienste trifft demnach immer das eHome.

Mit diesem XML-Dokument wird der IssueCertificateRecipient initialisiert (Zeilen 21-23). Diese idemix-Klasse ist für das Anfordern eines Credentials verantwortlich. Nach ihrer Instanziierung kann das *rundenbasierte* Ausstellen des Credentials beginnen (Zeilen 25ff.). Am Ende dieses Vorgangs ist der Benutzer dann im Besitz eines neuen Session-Credentials.

Um einen Dienst benutzen zu können, muss der Benutzer einen Beweis auf Basis seines Session-Credentials berechnen. Dazu benötigt er die in Listing 6.8 gezeigte Methode `showSessionCredential` der Klasse IdemixHandler. Bei einem solchen Beweis wird immer nur eine Aussage über ein einzelnes Attribut, im Beispiel der Name des Dienstes, bewiesen. Möglich wäre es, mehrere Attribute gleichzeitig zu beweisen, dies ist in der aktuellen Version des eHome-Prototyps aber nicht notwendig. Daher verlangt die Methode als einziges Argument den Namen des Dienstes, auf den die Zugriffsberechtigung nachgewiesen werden soll (Zeile 3).

Der Ablauf sieht dabei vor, dass zunächst der öffentliche Schlüssel des Ausstellers als Byte-Strom vom IdemixAuthenticator abgerufen und in ein verwendbares Format konvertiert wird (Zeile 7ff.). Das vorliegende Session-Credential muss zudem in ein XML-Format konvertiert werden. Dabei geht die Typinformation des Attributs verloren. Diese Information ist aber nötig, um einen korrekten Nachweis berechnen zu können. Die Datentypen werden daher extrahiert und zusätzlich an das XML-Dokument angefügt (Zeile 15ff.). Aus dieser Repräsentation des Session-Credentials erstellt der AssertOutputter ein weiteres XML-Dokument, im welchem festgehalten ist, welche Attribute nachgewiesen werden sollen und wie der Nachweis berechnet werden muss (Zeile 24ff.). Dieses XML-Dokument wird in zwei Versionen erstellt. Der Beweiser (ProveProver) erhält ein Dokument mit allen Informationen. Die zweite Version enthält keine privaten Daten aus dem Session-Credential und wird für den Verifizierer im ProofContainer verankert.

Der Beweiser wird mit dem XML-Dokument, dem privaten Schlüssel des Benutzers sowie den idemix-Parametern initialisiert (Zeile 33ff.). Zum Berechnen des Beweises benötigt er noch den Nonce-Wert des Verifizierers. Dazu ruft er die Initialisierungsmethode des Authenticators auf und erhält den Wert als Ergebnis (Zeile 39ff.). Anschließend wird der

```
1 private HlNodeCertificate sessionCredential;
2 private UserMasterSecret userPrivateKey;
3 public void requestSessionCredential
4   (Authenticator authenticator,
5     ServiceReference[] services) {
6   HashMap predicates = new HashMap();
7   // create predicates for specification, fill up with dummys
8   for (int i = 0; i < ATTRIBUTE_SIZE; i++) {
9     if (i < services.length) {
10      // store all available services
11      predicates.put(services[i].getServiceName(),
12             services[i].getServiceName());
13    }
14  }
15  // request the credential
16  String spec
17   = authenticator.prepareIssueCredential(predicates);
18  ByteArrayInputStream inStream
19   = new ByteArrayInputStream(spec.getBytes());
20  try {
21    IssueCertificateRecipient recipient
22     = new IssueCertificateRecipient(inStream,
23          userPrivateKey, publicParams, random);
24    // compute credential
25    Message m = recipient.round0();
26    Message m1 = authenticator.issueRound0(m);
27    Message m2 = recipient.round1(m1);
28    Message m3 = authenticator.issueRound2(m2);
29    recipient.round3(m3);
30    sessionCredential = recipient.getCertificate();
31  } catch (Exception e) {
32  }
33 }
```

Listing 6.7: Anfordern eines Session-Credentials.

Nachweis berechnet und auch im ProofContainer verankert (Zeile 41ff.). Damit ist die Aufgabe des IdemixHandlers erfüllt.

6.5.3 Beteiligte Bausteine am Aushandlungsprozess

In diesem Abschnitt wird beschrieben, welche Bausteine für die Realisierung des in Abbildung 6.9 abgebildeten Aushandlungsprozesses entwickelt wurden und wie sie mit den zuvor eingeführten Bausteinen zusammenarbeiten.

Ein Überblick über die an dem Prozess beteiligten Bausteine ist in Abbildung 6.18 dargestellt. Dabei sind die wichtigsten Aspekte abgebildet, um eine möglichst einfache Darstellung zu bieten. Im ersten Schritt (1) kontaktiert die MobileGUI den ServiceManager und erhält die Liste der im eHome verfügbaren Dienste und der von ihnen benötigten

```
1 private IdemixAuthenticator authenticator;
2 public ProofContainer showSessionCredential(String service) {
3   HashMap services = new HashMap();
4   services.put(service, service);
5   Vector preds = generatePredicates(services);
6   // load public key from ehome
7   IssuerPublicKey pubKey = null;
8   try {
9     ByteArrayInputStream byteIn = new ByteArrayInputStream(
10        authenticator.getSessionPublicKey());
11    DataInputStream in = new DataInputStream(byteIn);
12    pubKey = new IssuerPublicKey(in);
13  } catch (IOException e) { }
14  // transform credential to xml and add types of attributes
15  String credentialXML = sessionCredential.outputXML()
16      + "\n<AttributeTypes>";
17  Iterator iter
18   = sessionCredential.getDatatypes().iterator();
19  while (iter.hasNext()) {
20    credentialXML += (iter.next().toString());
21  }
22  credentialXML += "<AttributeTypes>";
23  // create assertion
24  AssertOutputter outputter = new AssertOutputter();
25  String assertion = outputter.createAssertion(
26              preds, credentialXML, pubKey,
27              publicParams, true);
28  ByteArrayInputStream inStream
29   = new ByteArrayInputStream(assertion.getBytes());
30  ProveProver prover;
31  try {
32    // init and compute proof
33    prover = new ProveProver(inStream,
34      userPrivateKey, publicParams, random);
35    // assertion for verifier
36    String verifierAssertion = outputter.createAssertion(
37              preds, credentialXML, pubKey,
38              publicParams, false);
39    APInteger verifierNonce
40     = authenticator.initVerifier(verifierAssertion);
41    prover.computeProof(verifierNonce);
42    ProofContainer p = new ProofContainer(
43              prover.getProof().toBinary(),
44              verifierAssertion);
45    return p;
46  } catch (Exception e) {}
47  return null;
48 }
```

Listing 6.8: Berechnen eines Nachweises auf Basis eines Session-Credentials.

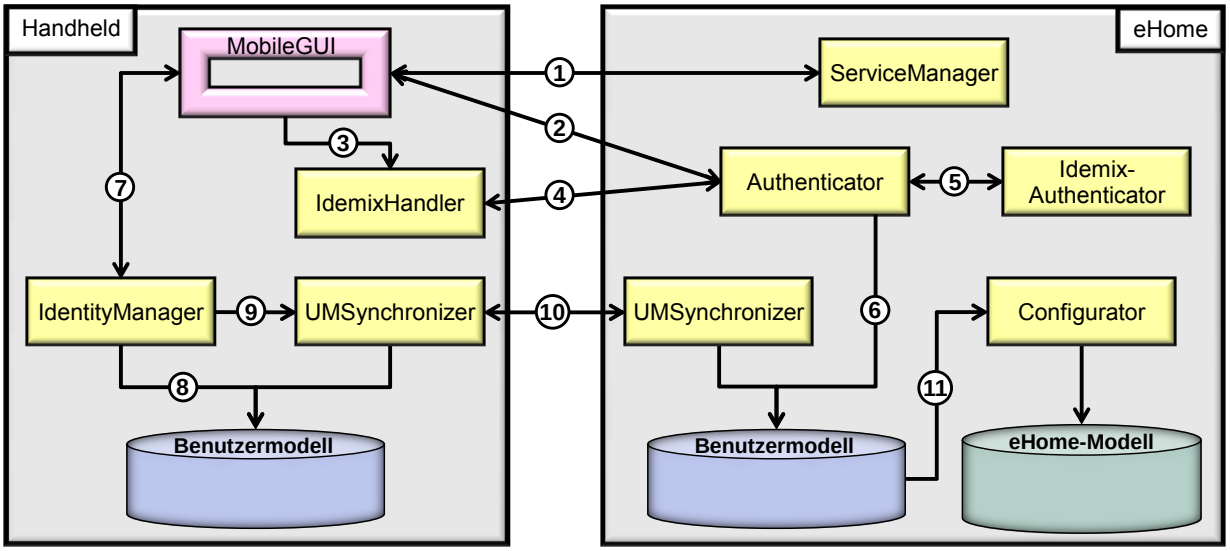


Abbildung 6.18: Beteiligte Bausteine an der Aushandlung von Diensten und Identitäten.

Benutzerdaten (s. Abschnitt 6.1.1). Implizit wird angenommen, dass sich der Benutzer in diesem Schritt für eine Menge von Diensten entscheidet.

Mit der Liste der ausgewählten Dienste wendet sich die MobileGUI an den Authenticator auf dem eHome-Gateway (2). Der Authenticator ist ein Baustein, der unterschiedliche Aufgaben im eHome wahrnimmt. Unter anderem ist er dafür zuständig, eHome-Dienste vor unbefugtem Zugriff zu schützen, wie im nächsten Kapitel genauer erläutert wird. Er ist aber auch für die Authentifizierung von Benutzern zuständig. Der Authenticator unterstützt zum einen die Authentifizierung über einen Anmeldenamen und ein Passwort. Er unterstützt aber auch die Authentifizierung mithilfe anonymer Credentials, indem er die Funktionalitäten des weiter oben eingeführten IdemixAuthenticators verwendet.

Nachdem nun der Authenticator die Liste der gewünschten Dienste erhalten hat, ermittelt er auf Basis dieser Liste die Eigenschaften, die der Benutzer nachweisen muss, um für die Nutzung der Dienste autorisiert zu werden. Der Authenticator teilt der MobileGUI daher mit, was für Eigenschaften der Benutzer nachweisen muss. Diese Information leitet die MobileGUI wiederum dem IdemixHandler weiter (3), der auf dem Handheld speziell für die Anforderung von Credentials und Berechnung von Nachweisen verantwortlich ist. Der IdemixHandler überprüft die auf dem Handheld verfügbaren TTP-Credentials und berechnet den geforderten Nachweis, falls ein geeignetes Credential vorhanden ist. Existieren mehrere passende Credentials, kann der Benutzer ein Credential interaktiv auswählen. Den berechneten Nachweis nimmt der Authenticator entgegen (4) und leitet es zur Verifizierung weiter an den IdemixAuthenticator (5). Die Berechnung und die Verifizierung des Nachweises laufen dabei nicht getrennt ab, sondern werden zwischen dem IdemixHandler und dem IdemixAuthenticator *interaktiv* durchgeführt (s. auch Listing 6.7 und Listing 6.8).

Falls die Authentifizierung erfolgreich war, erstellt der Authenticator zusammen mit dem IdemixAuthenticator ein *Session-Credential*, der als Attribute die Namen der ausgehandelten

Dienste enthält, und teilt diesen dem IdemixHandhler auf dem Handheld mit (Schritte (4) und (5)). Falls der Benutzer einen Dienst anonym nutzen möchte, kann er dieses Credential einsetzen, um seine Berechtigung nachzuweisen (s. Abschnitt 6.1.2.3).

Zusätzlich zum Session-Credential erstellt der Authenticator nach einer erfolgreichen Authentifizierung ein zufälliges *Hauptpseudonym*, das er dem Benutzermodell mitteilt (6). Infolgedessen ist der Benutzer dem eHome für die Dauer seiner Sitzung nun über ein Pseudonym bekannt und kann von anderen Benutzern unterschieden werden. Hierbei handelt es sich um ein *Beziehungspseudonym*, der nur für dieses eHome und für diese eine Sitzung gültig ist. Es kann daher nicht mit anderen Pseudonymen des Benutzers aus früheren Sitzungen mit diesem eHome oder anderen eHomes verkettet werden. Der Authenticator bietet auch die Möglichkeit der nicht-anonymen Authentifizierung über ein länger gültiges Pseudonym, falls der Benutzer dies wünscht (s. Kapitel 7). Dieses Hauptpseudonym wird auch der MobileGUI mitgeteilt und ist damit auch dem Handheld bekannt.

Im Anschluss an die erfolgreiche Authentifizierung gilt es nun, dem eHome die für die ausgewählten Dienste benötigten Benutzerdaten zur Verfügung zu stellen. Hierin wird der Benutzer durch den IdentityManager auf seinem Handheld unterstützt (s. auch Abschnitt 6.1.1.1). Dieser zeigt dem Benutzer die Liste von existierenden Teilidentitäten, die die benötigten Daten enthalten (7). Aus diesen Teilidentitäten kann der Benutzer anschließend eine auswählen. Falls keine passende Teilidentität existiert, kann der IdentityManager automatisch eine neue generieren (8). Für die Repräsentation unterschiedlicher Teilidentitäten auf dem Handheld wird wieder die PrivacyBox der SituationalStatements verwendet (s. Abschnitt 4.5.5). Ähnlich zur Realisierung der Zugriffskontrolllisten aus Abschnitt 6.5.1.1 durch die Verwendung der Variablen `access` wird diesmal die Variable `purpose` aus der PrivacyBox verwendet. Für jede Teilidentität, zu der ein SituationalStatement zugeordnet wurde, nimmt der IdentityManager einen Eintrag in dieser Variable der zugehörigen PrivacyBox im mobilen Benutzermodell vor.

Wie schon in Abschnitt 5.2.1 erläutert wurde, ist der UMSynchronizer für die Synchronisierung von Benutzerdaten zwischen Handheld und eHome zuständig. Der UMSynchronizer auf dem Handheld kann bei Bedarf von dem IdentityManager angewiesen werden, eine Synchronisierung für die aktuelle Teilidentität zu starten (9). Dies ist beispielsweise nach einer Authentifizierung oder nach einer Aktualisierung der aktiven Teilidentität der Fall. Der UMSynchronizer ermittelt daraufhin das aktuelle Hauptpseudonym, unter dem der Benutzer dem eHome bekannt ist, und startet die Synchronisation. Um sicherzustellen, dass nur Daten der aktuellen Teilidentität synchronisiert werden, überprüft der UMSynchronizer die Zugehörigkeit der SituationalStatements zur Teilidentität durch die Analyse der im letzten Absatz erwähnten Variable *purpose*. Für weitere Ereignisse, die eine Synchronisierung der Daten erfordern, und deren genauen Ablauf sei auf Abschnitt 5.2.1 verwiesen.

Somit hat das Benutzermodell auf dem eHome-Gateway die Daten des Benutzers erhalten, die er intern dem betreffenden Hauptpseudonym zuordnen kann. Bevor nun die Dienste (in der Abbildung nicht dargestellt) auf diese Daten zugreifen und dem Benutzer ihre personalisierbaren Funktionalitäten anbieten können, generiert das Benutzermodell zuvor die *Tokens* und teilt sie sowohl den Diensten als auch dem Configurator mit. Damit das Benutzermodell die Tokens generieren kann, teilt ihm der Authenticator zusätzlich zum

Hauptpseudonym in Schritt (6) auch die Liste der ausgehandelten Dienste mit. Die Dienste benötigen die Tokens, um beim Benutzermodell Benutzerdaten anzufragen. Der Configurator hingegen benötigt die Tokens, um *personengebundene* Dienste im Rahmen der Intra-eHome-Mobilität umkonfigurieren zu können. Die Details dieser Art der Umkonfigurierung wurden bereits in Abschnitt 4.3.2.3 erläutert. Dort wurde auch beschrieben, dass personengebundene Dienste im eHome-Modell an Person-Objekte gebunden sind, damit der Configurator einen Überblick darüber hat, welche Dienste umkonfiguriert werden müssen, wenn ein Benutzer seinen Aufenthaltsraum wechselt. Dies gilt gleichermaßen auch für den Sonderfall von persönlichen Diensten, die auf einem Handheld ausgeführt werden. Vor dieser Arbeit waren diese Objekte mit dem Hauptpseudonym des zugehörigen Benutzers gekennzeichnet. Dadurch konnte jeder Dienst, der Zugriff auf das eHome-Modell hatte, einfach das *Dienstnutzungsprofil* der Benutzer ermitteln. Daher werden jetzt auch im eHome-Modell Tokens eingesetzt, um das Dienstnutzungsprofil außerhalb des Benutzermodells geheim zu halten.

Alles in allem wurde der eHome-Prototyp um neue Bausteine erweitert, um die Konzepte zur Wahrung der Privatsphäre mobiler Benutzer zu gewährleisten. Im Hinblick auf die in diesem Kapitel vorgestellten Konzepte sind für das Handheld der IdentityManager und der IdemixHandler hinzugekommen. Für das eHome hingegen sind der Authenticator und der IdemixAuthenticator neu. Während diese Bausteine für die Datensparsamkeit und die Anonymität von Benutzern im Rahmen der Inter-eHome-Mobilität zuständig sind, spielt das Benutzermodell eine wichtige Rolle für die Unverkettbarkeit von Benutzerdaten und -aktionen während einer Sitzung.

6.6 Zusammenfassung

In diesem Kapitel wurden die in dieser Arbeit entwickelten Konzepte zum Schutz der Privatsphäre mobiler Benutzer im Rahmen der Inter-eHome-Mobilität beschrieben. Dabei wurde als besonderes Ziel die informationelle Selbstbestimmung der Benutzer verfolgt. Mit anderen Worten sollten die entwickelten Konzepte mobile Benutzer dabei unterstützen, selbst bestimmen und kontrollieren zu können, wer wann welche Informationen über ihn erhält. Um eine möglichst feingranulare Informationsflusskontrolle zu realisieren, wurde zwischen eHomes und eHome-Diensten als informationssammelnde Entitäten unterschieden.

Zunächst wurde diskutiert, wie die Privatsphäre auf der eHome-Ebene geschützt werden kann. Hier wurde als geeigneter Ansatz das Prinzip der Datensparsamkeit identifiziert, das nur die Herausgabe von Daten vorsieht, die für die vom Benutzer gewünschten Dienste notwendig sind. Für die Umsetzung dieses Prinzips wurde ein aushandlungsbasiertes Identitätsmanagementsystem entwickelt, das den Benutzer dabei unterstützt, mit eHomes aushandeln zu können, welche Dienste er verwenden und welche Daten er preisgeben möchte.

Dieses Konzept für sich allein ist jedoch nur dann ausreichend, wenn keine eHomes zusammenarbeiten, um ihre Daten zu verketten. Um solch eine Verkettbarkeit zu verhindern, wurde daher ein Authentifizierungsmechanismus auf Basis *anonymer Credentials* entwickelt. Dadurch sind mobile Benutzer in der Lage, sich bei fremden eHomes anonym anzumelden und Dienste zu nutzen.

Als Nächstes wurde diskutiert, wie verhindert werden kann, dass einzelne oder zusammenarbeitende eHome-Dienste mehr Benutzerdaten erlangen, als ihnen zusteht. Hierfür wurden die Daten genauer untersucht, die der Benutzer dem eHome im Rahmen seiner aktuellen Sitzung bereitgestellt hat. Weil diese Daten durch das Benutzermodell im eHome verwaltet werden, wurden Konzepte entwickelt, die an dieser Stelle ansetzen. Zunächst wurde das Benutzermodell um eine selektive Zugriffskontrolle auf Basis von Zugriffskontrolllisten erweitert, die sicherstellt, dass nur solche Dienste auf ein Attribut im Benutzermodell zugreifen dürfen, das sie für ihre Funktionalität benötigen. Um ferner die Verkettbarkeit von Benutzerdaten durch Dienste zu verhindern, wurden spezielle, als Token bezeichnete, Pseudonyme eingeführt. Diese Tokens werden von Benutzermodell generiert und von den Diensten für den Datenzugriff verwendet.

Schließlich wurde erläutert, wie Benutzerdaten sowohl bei der persistenten Speicherung als auch während der drahtlosen Übertragung zwischen Handhelds und eHomes gesichert wurden. Hierfür wurden verschiedene Verschlüsselungsverfahren eingesetzt.

Zum Schluss dieses Kapitels wurden einige Details zur Realisierung der entwickelten Konzepte beschrieben. Eine besondere Rolle nahmen dabei die Bausteine IdentityManager und Authenticator an.

Kapitel 7

Zugriffskontrolle

Im vorangegangenen Kapitel wurde der Lösungsansatz zum Schutz der Privatsphäre vorgestellt, der die Schutzinteressen von Benutzern berücksichtigt. Das Prinzip der mehrseitigen Sicherheit erfordert jedoch die Berücksichtigung der Schutzinteressen aller an einer Kommunikationsbeziehung Beteiligten. Im Kontext dieser Arbeit bedeutet dies, dass auch die Interessen von eHome-Betreibern verfolgt werden sollten. Ein besonderes Schutzziel in diesem Zusammenhang ist die *Zugriffskontrolle für eHome-Dienste*, die in diesem Kapitel behandelt wird.

Es können zwei Arten von Zugriffen auf eHome-Dienste unterschieden werden. Erstens können Benutzer auf eHome-Dienste zugreifen. Zweitens können eHome-Dienste auf andere Dienste zugreifen. Für beide Arten wurden jeweils Konzepte zur Vermeidung unbefugter Zugriffe entwickelt, wie im Folgenden beschrieben wird. Anschließend wird diskutiert, wie die Zugriffskontrolle nicht-invasiv durchgeführt werden kann, also ohne die Implementierung von den Diensten selbst anpassen zu müssen. Danach werden wichtige Realisierungsdetails erörtert, bevor dieses Kapitel mit einer Zusammenfassung abgeschlossen wird.

7.1 Schutz vor Zugriff unbefugter Benutzer

Der Bedarf für den Schutz von eHome-Diensten vor dem Zugriff unbefugter Benutzer wurde schon in Abschnitt 1.2.2 und Abschnitt 1.3.5 motiviert. Ohne eine angemessene *Zugriffskontrolle* könnte beispielsweise nicht verhindert werden, dass ein Hotelgast die Sauna benutzt, ohne dafür bezahlt zu haben. Ferner könnten Studenten in einem Lehrstuhl das Faxgerät nutzen, obwohl dieser nur für die Mitarbeiter bestimmt ist. In einem pessimistischeren Szenario könnten Kriminelle Sensortreiber nutzen, um Informationen über den Zustand eines privaten Haushalts zu sammeln. Mithilfe dieser Information könnten Einbrüche besser geplant werden. Umgekehrt kann auch die unerlaubte Nutzung von Aktortreibern die Sicherheit eines eHomes gefährden, etwa wenn ein Außenstehender die Haustür mit seinem PDA öffnen könnte.

Solche Beispiele der unbefugten Dienstnutzung können durch Zugriffskontrolle unterbunden werden. Eine wichtige Voraussetzung hierfür ist ein geeigneter *Authentifizierungsmechanismus*. Im vorigen Kapitel wurde ein Mechanismus vorgestellt, der die Authentifizierung

von Benutzern anhand anonymer Credentials ermöglicht. Anonyme Credentials haben den Vorteil, dass sie eine zuverlässige, fälschungssichere Authentifizierung von Benutzern ermöglichen und sich gleichzeitig zum Schutz der Privatsphäre eignen. Der Einsatz anonymer Credentials ist jedoch nicht überall sinnvoll. In vertraulichen Umgebungen, wie etwa zu Hause, ist die eindeutige Identifizierung der Benutzer unter Umständen eine bessere Alternative, zumal anonyme Credentials mit einer rechenintensiven Authentifizierung einhergehen, die sich auf Handhelds komfortmindernd auswirken können.

Diese Ausführung macht deutlich, dass abhängig davon, welchen Grad an Schutz der Privatsphäre ein Benutzer wünscht und welche Anforderungen ein eHome an die Identifizierbarkeit von Benutzern stellt, unterschiedliche Arten der Authentifizierung anwendbar sein sollten. Im Rahmen dieser Arbeit wurden die im Folgenden erläuterten Mechanismen realisiert, die den Benutzern unterschiedliche Anonymitätsstufen gewähren. Diese Mechanismen sind dabei in den Aushandlungsprozess eingebettet, der in Abschnitt 6.1.1 beschrieben wurde.

7.1.1 Authentifizierung über Beziehungspseudonyme

Die einfachste Möglichkeit der Authentifizierung verlangt vom Benutzer, dass er sich über eindeutige Merkmale anmeldet. Dies kann beispielsweise über einen Benutzernamen und einem zugehörigen Passwort geschehen. Dieses Verfahren ist weit verbreitet und wird insbesondere bei der Anmeldung an Computern verwendet. Insbesondere für eHomes könnten auch weitere Authentifizierungsmechanismen anstelle von Benutzernamen und Passwort eingesetzt werden. Beispiele hierfür sind der Einsatz von Chipkarten oder von biometrischen Merkmalen wie der Fingerabdruck oder die Irisstruktur. Die Erweiterung des im Rahmen dieser Arbeit entwickelten Ansatzes um solche zusätzlichen Verfahren ist problemlos möglich.

Diese Verfahren setzen voraus, dass der Benutzer dem eHome zuvor, beispielsweise über ein *Beziehungspseudonym*, bekannt ist. Ferner müssen sich das eHome und der Benutzer zuvor darüber geeinigt haben, anhand welcher Merkmale sich der Benutzer anmelden muss, um die Dienste des eHomes zu nutzen. Beispielsweise müssen der Benutzername und das Passwort bestimmt werden. Üblicherweise ist der eHome-Betreiber bzw. ein Administrator dafür zuständig, für solche Benutzer jeweils einen Account anzulegen. Bei der Anlegung dieses Accounts wird auch festgelegt, welche Dienste der Benutzer nach seiner Anmeldung nutzen darf. Im Rahmen dieser Arbeit wird für diesen Fall die *rollenbasierte Zugriffskontrolle* verwendet. Der Benutzer wird daher gemäß seiner Befugnisse oder Rechte einer Rolle zugeordnet, die ihm den Zugriff auf eine Menge von Diensten im eHome erlaubt. Meldet sich der Benutzer in Zukunft dann mit diesem Benutzernamen und dem Passwort (oder anderen Merkmalen) an, kann er diese Dienste nutzen. Der Zugriff auf weitere Dienste wird ihm verweigert.

Besonders in kleinen und privaten eHomes hat dieser Ansatz Vorteile. Einerseits existiert hier keine starke Benutzerfluktuation, sodass der Administrator nicht zu sehr mit der Verwaltung von Accounts belastet wird. Andererseits haben Benutzer geringere Anonymitätsanforderungen an solche Umgebungen als an fremde eHomes. Diese Verfahren

ermöglichen dem eHome nämlich, Benutzeraktionen über mehrere Sitzungen hinweg miteinander zu verketten. Während sich solch eine Verkettbarkeit in privaten Umgebungen komfortsteigernd auswirken kann, birgt sie Gefahren bzgl. der Privatsphäre. Falls der Benutzer sich zudem mit dem gleichen Merkmal, wie etwa mit dem Fingerabdruck, bei mehreren eHomes authentifiziert, besteht auch die Möglichkeit der Verkettbarkeit zwischen den eHomes.

7.1.2 Authentifizierung über anonyme Credentials

Um den Anonymitätsanforderungen von mobilen Benutzern gegenüber eHomes wie Hotels oder Bürogebäude gerecht zu werden, die ein weniger privates Umfeld und eine starke Benutzerfluktuation aufweisen, wird der im letzten Kapitel vorgestellte Ansatz verwendet. Dieser ermöglicht die Authentifizierung von Benutzern auf Basis anonymer Credentials. Während der Fokus im vorangegangenen Kapitel auf dem Schutz der Privatsphäre lag, wird hier die Realisierung der Zugriffskontrolle anhand anonymer Credentials diskutiert.

Anonyme Credentials eignen sich besonders gut für die Zugriffskontrolle, weil sie fälschungssicher und nicht übertragbar sind. Darüber hinaus bieten sie den Vorteil, dass die Benutzer den eHomes zuvor nicht bekannt sein müssen. Stattdessen können sie sich mit Credentials authentifizieren, die von vertrauenswürdigen Stellen bezogen wurden (TTP-Credentials). Damit sinkt der Verwaltungsaufwand für eHome-Betreiber. Ein weiterer Vorteil liegt darin, dass das eHome Benutzeraktionen nun nicht mehr über mehrere Sitzungen hinweg miteinander verketten kann und somit die Privatsphäre besser geschützt wird.

Des Weiteren kann ein Benutzer im Rahmen des hier entwickelten Ansatzes auswählen, seine Anonymität auch während der Sitzung beizubehalten. Dabei kann er sich wie folgt entscheiden:

- **Ohne Sitzungsanonymität**: Falls der Benutzer kein Session-Credential nutzen möchte, wird ihm nach der erfolgreichen Authentifizierung mit dem TTP-Credential ein zufällig generiertes Pseudonym zugeordnet. Dieses Pseudonym ist mit einer Rolle assoziiert, die Zugriffsrechte für die ausgehandelten Dienste enthält[1]. Anschließend wird bei Dienstzugriffen mithilfe dieser Rolle entscheiden, ob die Zugriffe dieses Benutzers gewährt werden oder nicht. Auf diese Weise kann die Zugriffskontrolle effizient durchgeführt werden, weil keine rechenintensiven Nachweise auf Basis von Credentials berechnet und verifiziert werden müssen. Dafür sind die Benutzeraktionen innerhalb einer Sitzung jedoch miteinander verkettbar, es besteht also keine Sitzungsanonymität.

- **Mit Sitzungsanonymität**: Möchte der Benutzer nicht, dass seine Aktionen innerhalb einer Sitzung miteinander verkettet werden, kann er sich dafür entscheiden, ein Session-Credentials zu erhalten. Diese Möglichkeit wurde in Abschnitt 6.1.2.3 vorgeschlagen und bereits ausführlich diskutiert. Diese Variante bietet den bestmöglichen Schutz der Privatsphäre, weil bei *jedem* Zugriff ein neuer Nachweis der Zugriffsberechtigung auf

[1] Falls noch keine Rolle existiert, die Zugriff auf die ausgehandelten Dienste erlaubt, wird automatisch eine neue Rolle mit den passenden Rechten generiert.

Basis von Zero-Knowledge-Beweisen berechnet und verifiziert wird. Da das eHome somit nicht auf das Session-Credential zurückschließen kann, das für die Berechnung des Nachweises eingesetzt wurde, kann es auch kein detailliertes Dienstnutzungsprofil des Benutzers erstellen. Der Nachteil dieser Variante liegt darin, dass die Berechnung von Nachweisen auf dem Handheld zeitintensiv ist und somit der Dienstzugriff insgesamt länger dauert.

Insgesamt bietet der in dieser Arbeit verfolgte Ansatz mobilen Benutzern vielfältige Möglichkeiten der Authentifizierung und Zugriffskontrolle, die unterschiedliche Grade der Anonymität bieten. Sie können sich somit aktiv an der Findung eines Kompromisses zwischen dem Schutz der Privatsphäre und der komfortablen Nutzung von eHome-Diensten beteiligen.

7.1.3 Sonderfall: Basisdienste

Basisdienste stellen für die Kontrolle von Zugriffen durch Benutzer einen Sonderfall dar. Ein Benutzer kann nämlich auf unterschiedliche Arten auf einen Dienst zugreifen.

Die erste Möglichkeit ist die Nutzung des Dienstes von seinem Handheld aus. In diesem Fall findet der Zugriff in Form eines direkten Methodenaufrufs des Top-Level-Dienstes statt. Es muss also nur überprüft werden, ob der Benutzer Zugriffsrechte für diesen Top-Level-Dienst besitzt. Dies ist nicht weiter kompliziert, weil die Rolle oder das Session-Credential des Benutzers genau die Top-Level-Dienste enthalten, die der Benutzer mit dem eHome ausgehandelt hat.

Es kann jedoch auch vorkommen, dass der Benutzer einen Schalter bzw. ein anderes Gerät bedient, das von einem Basisdienst kontrolliert wird und dessen Bedienung die Funktionalität eines Top-Level-Dienstes beeinflusst. Am Beispiel eines `Lichtdienstes` kann die Beleuchtungsintensität etwa auch über einen Dimmschalter an der Wand geregelt werden. Nun stellt sich die Frage, wie hier die Zugriffsberechtigung überprüft wird[2].

Eine Möglichkeit besteht darin, bei der Erzeugung der Rolle oder des Session-Credentials zusätzlich zu den Top-Level-Diensten auch alle Unterdienste bis auf die Ebene der Basisdienste in die Rechtevergabe einzubeziehen. Dann müsste nur noch überprüft werden, ob die Rolle bzw. das Session-Credential eine Zugriffsberechtigung für den Schalter beinhaltet. Der `Lichtdienst` in seiner Funktion als Top-Level-Dienst müsste dann nicht mehr betrachtet werden. Diese Möglichkeit wurde im Rahmen dieser Arbeit jedoch nicht verfolgt, weil die Rollen bzw. Session-Credentials auf diese Weise zu viele Informationen erhalten würden.

Stattdessen wird bei Zugriffen von Benutzern auf Geräte, die von einem Basisdienst kontrolliert werden, zunächst der zugehörige Top-Level-Dienst ermittelt. Hierfür wird die Dienstkomposition aus dem eHome-Modell Bottom-up durchlaufen. Anschließend wird überprüft, ob der Benutzer die Berechtigung für die Nutzung dieses Top-Level-Dienstes

[2]Für diese Diskussion wird angenommen, dass das eHome anhand einer besonderen Technik genau ermitteln kann, welcher Benutzer den Schalter betätigt hat. Die Entwicklung solch einer Technik ist im Moment noch Gegenstand der Forschung. In Abschnitt 8.3 werden die Ergebnisse dieser Arbeit anhand von Softwaredemonstratoren evaluiert. Diese Demonstratoren wurden so entwickelt, dass jede Bedienung eines Gerätes eindeutig einem Benutzer zugeordnet werden kann.

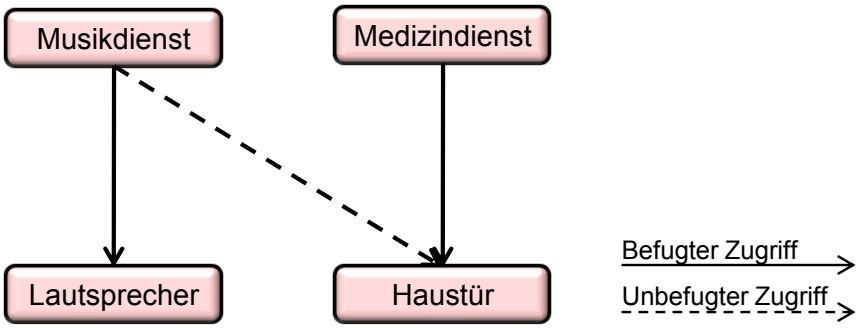


Abbildung 7.1: Beispiel für befugte und unbefugte Dienstzugriffe.

besitzt. Sollte also ein Benutzer versuchen, die Beleuchtung über einen Schalter zu bedienen, wird die Zugriffskontrolle für den zugehörigen `Lichtdienst` durchgeführt und nicht für den Schalter selbst.

7.2 Schutz vor Zugriff unbefugter Dienste

In eHomes sollten nicht nur Zugriffe von Benutzern einem Kontrollmechanismus unterliegen, sondern auch die *Interaktionen von Diensten untereinander*. Auch hier kann es vorkommen, dass ein Dienst auf einen anderen Dienst zugreift, den er aber nicht für seine eigene Funktionalität benötigt. Bereits in Abschnitt 1.2.2 wurde ein Beispiel für solch einen Fall gegeben. Dieses Beispiel ist in Abbildung 7.1 visualisiert. Dem `Medizindienst` ist beispielsweise erlaubt, in Notfällen die `Haustür` automatisch zu öffnen, um dem Rettungsdienst Zutritt zum Haus zu ermöglichen. Der `Musikdienst` darf hingegen die `Lautsprecher` verwenden, um Musik abzuspielen.

Offensichtlich sollte der `Musikdienst` jedoch nicht die `Haustür` öffnen dürfen. Falls dieser beispielsweise von kriminellen Dritten kontrolliert wird, könnten sie versuchen, sich über diesen Dienst Zugang zum Haus zu verschaffen, wenn niemand anwesend ist. Neben der unbefugten Nutzung von Aktoren wie im vorigen Beispiel, birgt auch die unbefugte Nutzung von Sensoren Gefahren. Es könnten dadurch Informationen über den Zustand des eHomes oder der im eHome aktiven Benutzer gesammelt werden. Beispielsweise sollte nur der `Medizindienst` Daten von Sensoren erhalten, die die Vitalwerte von Benutzern erfassen. Der `Musikdienst` hingegen sollte dies nicht dürfen. Andernfalls würden sich Möglichkeiten zum Ausspähen der Privatsphäre der Benutzer ergeben. Der in dieser Arbeit verfolgte Ansatz sieht daher vor, dass solche *persönlichen* Sensoren ihre Daten ausschließlich im Benutzermodell ablegen und nicht von anderen Diensten benutzt werden.

Wie die obige Diskussion verdeutlicht, muss die Zugriffskontrolle zusätzlich zu Benutzerzugriffen auf Dienste auch die Zugriffe von Diensten untereinander berücksichtigen. Nun stellt sich die Frage, wie diese Zugriffskontrolle für Dienstzugriffe umgesetzt werden kann. In dieser Arbeit wird auch hierfür ein *rollenbasierter Ansatz* verfolgt. Jedem Dienst wird dabei eine Rolle zugeordnet, die ihm Zugriffsrechte auf genau die Dienste erlaubt, die er nutzen darf.

Dienstname	**Dienstrolle**	**Zugriff auf Dienste**
Personalisierbarer Temperaturdienst	Personalisierbarer Temperaturdienst	Temperaturdienst
Temperaturdienst	Temperaturdienst	Thermometer, Heizung
Musikdienst	Musikdienst	Lautsprecher
Medizindienst	Medizindienst	Haustür

Tabelle 7.1: Beispiele für Dienstrollen.

Als Nächstes stellt sich die Frage, wie die Rollen für Dienste verwaltet werden und wie bestimmt werden soll, welcher Dienst welcher Rolle zugeordnet wird. Eine Möglichkeit besteht darin, dass ein Administrator des eHomes die Rollen manuell anlegt. Dieser Ansatz ist zwar für die Verwaltung von Benutzerrollen geeignet, nicht aber für Dienstrollen, weil ein eHome unter Umständen sehr viele Dienste betreiben kann. Außerdem ist die Dienstelandschaft in eHomes üblicherweise nicht statisch, da zur Laufzeit Dienste hinzukommen oder entfernt werden können. Dies hätte zur Folge, dass der Administrator sich aufgrund geänderter Umstände ständig mit der Verwaltung von Dienstrollen beschäftigen muss.

Ein praktikablerer Ansatz, der in dieser Arbeit verfolgt wird, sieht die *automatische* Generierung der Rollen und ihre Anpassung an geänderte Umstände zur Laufzeit vor. Die hierfür nötigen Informationen können dem eHome-Modell, oder genauer gesagt der *Dienstkomposition* aus dem eHome-Modell, entnommen werden. In der Dienstkomposition sind nämlich alle Abhängigkeiten zwischen Diensten durch eine Graphdatenstruktur beschrieben, die explizit vom eHomeConfigurator im Rahmen der Konfigurierung aufgebaut wird. Dieser Ansatz sieht vor, dass alle Zugriffe zwischen Diensten erlaubt werden, die den Abhängigkeiten in der Dienstkomposition entsprechen. Daher wird für jeden Dienst genau eine Rolle erzeugt, die ihm den Zugriff auf die Dienste erlaubt, mit denen er in der Dienstkomposition verbunden ist.

Tabelle 7.1 zeigt einige Beispiele für Dienstrollen, die aus einer Dienstkomposition abgeleitet wurden. Dabei beziehen sich die ersten zwei Zeilen auf die Dienstkomposition aus Abbildung 2.5 und die letzten beiden Zeilen auf die Dienstkomposition aus Abbildung 7.1. Weil die Dienstrollen automatisch generiert werden und jeder Dienst üblicherweise eine eigene Rolle mit spezifischen Zugriffsrechten erhält, bekommt jede Dienstrolle den gleichen Namen wie der zugehörige Dienst[3]. Beispielsweise sagt die zweite Zeile der Tabelle 7.1 aus, dass der `Temperaturdienst` die Rolle `Temperaturdienst` erhalten hat, die ihm Zugriff auf die beiden Unterdienste `Thermometer` und `Heizung` gewährt. Die dritte Zeile hingegen sagt aus, dass der `Musikdienst` mit der Rolle `Musikdienst` ausschließlich auf den `Lautsprecher` zugreifen darf. Er darf beispielsweise nicht auf die `Haustür` zugreifen, der `Medizindienst` hingegen schon, wie in der vierten Zeile angegeben ist.

Diesem Ansatz liegt die Annahme zugrunde, dass Dienste, die der eHome-Betreiber in seinem eHome deployt, zunächst vertrauenswürdig sind. Diese Annahme kann dadurch

[3]Aus diesem Grund kann es vorkommen, dass zwei generierte Rollen dieselben Zugriffsrechte enthalten, wenn die zugehörigen Dienste die gleichen Unterdienste besitzen. Diese Redundanz stellt aber keinen Mehraufwand dar, da Dienstrollen automatisch verwaltet werden.

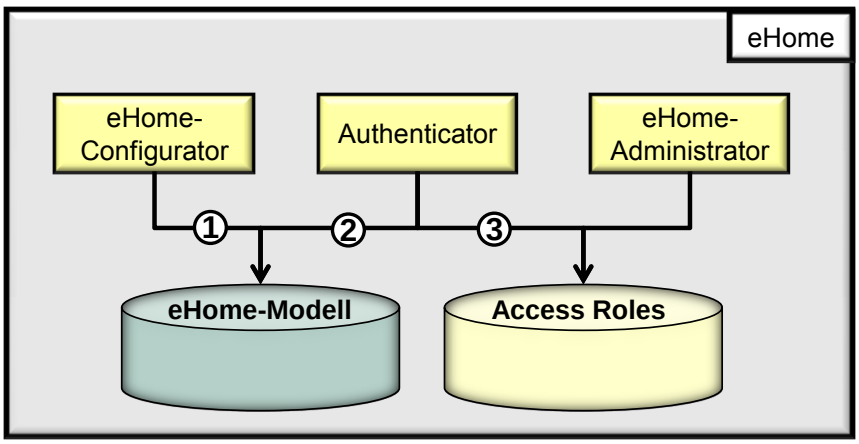


Abbildung 7.2: Der Authenticator ist für die Verwaltung von Zugriffsrollen zuständig.

untermauert werden, dass in einem eHome nur solche Dienste installiert werden, die von einem vertrauenswürdigen Anbieter stammen. Um die Vertrauenswürdigkeit der Dienste überprüfen zu können, wird in [PF06] beispielsweise vorgeschlagen, nur signierte OSGi-Bundles zu installieren. Nach der Signierung vorgenommene (bösartige) Änderungen der Bundles, und damit der Dienste, können dann erkannt werden. Die Signierung von Bundles stellt jedoch nur sicher, dass der eHome-Betreiber vor dem Deployment eines Bundles sicher sein kann, dass der Dienst nachweislich von dem angegebenen Hersteller stammt. Es eignet sich jedoch nicht für die Zugriffskontrolle zur Laufzeit. Falls der Hersteller doch bösartig ist oder ein Dienst zur Laufzeit unter die Kontrolle eines bösartigen Dritten gerät, kann der Dienst versuchen, auf Dienste zuzugreifen, für die er keine Zugriffsrechte hat. Daher reicht eine Signierung von Bundles allein nicht aus, weil eine Lücke zur Laufzeit entstehen kann. Die Zugriffskontrolle, wie sie in dieser Arbeit vorgeschlagen wird, schließt diese Lücke.

Zuständig für die automatische Erstellung der Dienstrollen ist der Authenticator, der bereits in Abschnitt 6.5.3 eingeführt wurde. Dieser ist nicht nur für die Authentifizierung von Benutzern, sondern auch für die Verwaltung von Benutzer- und Dienstrollen (`Access Roles`) zuständig. Während die Verwaltung der Benutzerrollen in Abschnitt 8.1 beschrieben wird, ist in Abbildung 7.2 skizziert, wie der Authenticator Dienstrollen erstellt und pflegt. Zunächst registriert sich der Authenticator beim eHome-Modell als Beobachter, um stets den aktuellen Zustand der Dienstkomposition überwachen zu können, aus der die Dienstrollen generiert werden.

Die Dienstkomposition wird ausschließlich durch den eHomeConfigurator verwaltet (1). Zum ersten Mal erstellt der Authenticator die Dienstrollen, wenn das eHome in Betrieb genommen wird. Dafür durchläuft er die Dienstkomposition Top-down (2) und generiert für jeden Dienst eine eigene Rolle, die ihm Zugriffsrechte entsprechend seiner Abhängigkeiten gewährt (3). Sollten sich die Abhängigkeiten aufgrund der strukturellen Adaption zur Laufzeit ändern, aktualisiert auch der Authenticator die entsprechenden Dienstrollen. Auf diese Weise berücksichtigt er die *Dynamik* von Dienstkompositionen in eHomes und stellt sicher, dass die Zugriffsrechte jederzeit mit der Dienstkomposition konsistent sind.

Ferner ist in Abbildung 7.2 der eHomeAdministrator abgebildet. Hierbei handelt es sich um ein im Rahmen dieser Arbeit entwickeltes Werkzeug, das eine Benutzeroberfläche für die Verwaltung von Benutzerdaten sowie Benutzer- und Dienstrollen (*Access Roles*) anbietet. Dieses Werkzeug wird in Abschnitt 8.1 detailliert beschrieben.

7.2.1 Sonderfall: mobil ausgeführte Dienste

Auch Dienste, die auf einem Handheld ausgeführt werden, greifen unter Umständen auf Dienste im eHome zu. Daher stellt sich hier die Frage, wie ein Zugriff dieser Art kontrolliert werden kann.

In diesem Fall kann das gleiche Verfahren angewendet werden wie für Dienste, die im eHome ausgeführt werden. Die Dienstkomposition im eHome-Modell enthält nämlich auch die mobil ausgeführten Dienste, die speziell markiert sind. Daher unterscheidet sich dieser Sonderfall in Bezug auf die Zugriffskontrolle nicht von der normalen Situation. Aus diesem Grund wird die Ausführung eines Dienstes auf Handhelds nicht anders gehandhabt.

7.3 Durchführung der Zugriffskontrolle

Die Zugriffskontrolle für eHome-Dienste gehört zu den *nichtfunktionalen* Anforderungen an eHomes. Weil ihre Realisierung mehrere Komponenten betrifft, wird diese Anforderung auch als *querschneidend* bezeichnet. Für eine detaillierte Beschreibung von querschneidenden Anforderungen sei auf [Nus04, KLM+97] verwiesen.

In diesem Abschnitt wird diskutiert, wer die Zugriffskontrolle durchführen sollte. Dabei werden zwei alternative Mechanismen diskutiert. Der Erste sieht vor, dass die Dienste selbst die Zugriffskontrolle durchführen. Der Zweite sieht den Einsatz eines Interceptors vor, der die Zugriffskontrolle durchführt. Für die Diskussion wird beispielhaft der Zugriff eines Benutzers über sein Handheld auf den `Faxdienst` in einem eHome herangezogen. Dabei wird die Zugriffsberechtigung auf Basis eines Session-Credentials nachgewiesen. Die geführte Diskussion lässt sich jedoch auch analog auf die übrigen Arten von Zugriffen übertragen, die weiter oben diskutiert wurden.

7.3.1 Durchführung durch Dienste

Eine Möglichkeit, die Zugriffskontrolle durchzuführen, besteht darin, die *Dienste in den Entscheidungsprozess einzubeziehen.* Ein Beispiel hierfür ist in Abbildung 7.3 dargestellt. Dort greift der Benutzer mit seinem Handheld auf den `Faxdienst` zu. Der Zugriff wird abgebildet auf den Aufruf der Methode `doAction()`, der von der Benutzeroberfläche des Handhelds aufgerufen wird.

Die Besonderheit dieser Möglichkeit liegt darin, dass der Dienst eine aktive Rolle bei der Durchführung der Zugriffskontrolle spielt. Wie in Abbildung 7.3 gezeigt, weist der Benutzer seine Zugriffsberechtigung auf Basis seines Session-Credentials nach. Hierfür fügt er dem Aufruf der Methode auch den Nachweis bei (1). Bevor der Dienst die geforderte Aktion jedoch ausführt, leitet er den Nachweis zunächst an den Authenticator weiter (2), weil er

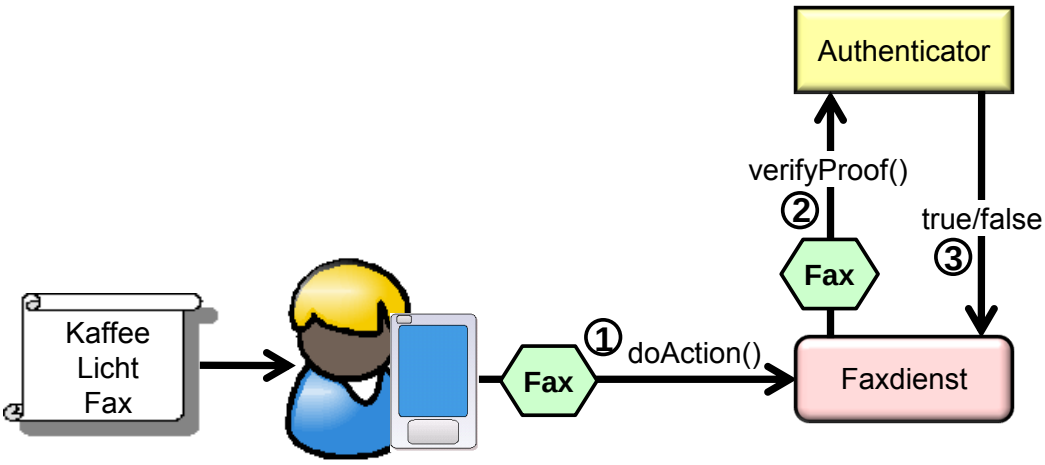


Abbildung 7.3: Zugriffskontrolle durch die Dienste am Beispiel des `Faxdienstes`.

den Nachweis nicht selbst verifizieren kann. Dieser überprüft anhand des Nachweises, ob der Benutzer befugt ist, auf den `Faxdienst` zuzugreifen. Abhängig von dem Ergebnis der Verifikation erhält der Dienst als Antwort `true` oder `false` zurück. Im ersten Fall führt er die gewünschte Aktion aus, im zweiten Fall jedoch nicht.

Anhand dieses Beispiels wird eine weitere Funktionalität des Authenticators deutlich. Zusätzlich zur Authentifizierung von Benutzern und der automatischen Verwaltung von Dienstrollen ist der Authenticator auch für die Überprüfung bzw. Verifizierung von Zugriffsberechtigungen verantwortlich. Während in diesem Beispiel der Nachweis auf Basis eines Session-Credentials verifiziert wurde, wird bei der rollenbasierten Zugriffskontrolle die Berechtigung anhand der Rolle des Aufrufenden überprüft.

7.3.2 Durchführung durch einen Interceptor

Eine weitere Möglichkeit, die Zugriffskontrolle durchzuführen, besteht darin, einen *Interceptor* einzusetzen. Auf diese Weise müssen sich nicht mehr die Dienste um die Zugriffskontrolle kümmern. Diese Aufgabe nimmt ihnen der Interceptor ab.

Der Interceptor ist in Abbildung 7.4 in Form eines Halbkreises dargestellt. Er fängt jeden Zugriff auf einen eHome-Dienst ab und überprüft gemeinsam mit dem Authenticator die Berechtigung des Aufrufenden. In dem Beispiel aus Abbildung 7.4 landet der Aufruf zusammen mit dem zugehörigen Nachweis zunächst beim Interceptor (1). Der `Faxdienst` erfährt noch nichts von dem Aufruf. Zuerst leitet der Interceptor den Nachweis an den Authenticator weiter (2). Der Authenticator wiederum verifiziert den Nachweis, wie zuvor beschrieben. Das Ergebnis der Verifizierung teilt er dem Interceptor mit (3). Falls Antwort des Authenticators positiv war, führt der Interceptor den Aufruf auf dem `Faxdienst` aus. Falls nicht, wird der Aufruf nicht durchgeführt.

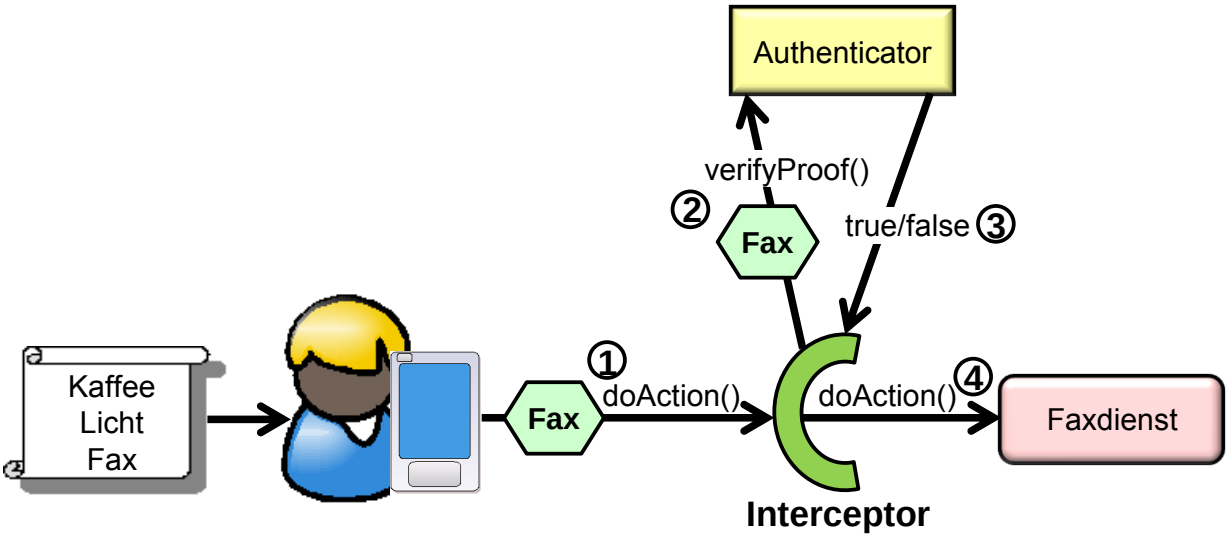


Abbildung 7.4: Zugriffskontrolle durch den Interceptor am Beispiel des `Faxdienstes`.

7.3.3 Diskussion

In diesem Abschnitt wird diskutiert, welche der beiden Möglichkeiten besser für die Durchführung der Zugriffskontrolle geeignet ist.

Ein Vorteil der Durchführung der Zugriffskontrolle durch die Dienste ist der, dass jeder Dienst selbst dafür sorgen kann, dass die Zugriffskontrolle tatsächlich durchgeführt wird. Sollte im Alternativen Fall der Interceptor nämlich umgangen werden, könnten alle Dienste im eHome ohne Zugriffskontrolle genutzt werden.

Dies ist jedoch der einzige Nachteil des Interceptor-Ansatzes. Es sprechen viele Gründe gegen die Durchführung der Zugriffskontrolle durch die Dienste (s. [GHL08, Hur08]), wie im Folgenden beschrieben wird:

- Zunächst führt der Ansatz dazu, dass jeder Dienst Aufgaben wahrnehmen muss, die nicht zu seinen Kernaufgaben zählen. Dadurch steigt einerseits der Entwicklungsaufwand für die Dienste und andererseits leidet darunter die Wartbarkeit (Erweiterbarkeit und Änderbarkeit) des Systems. Außerdem kann die Schnittstelle des Dienstes davon betroffen sein, sodass auch die Verwender der Schnittstelle beeinflusst werden. Beispielsweise muss der Dienst in Abbildung 7.3 für jede Methode einen zusätzlichen Parameter für den Nachweis vorsehen, der mit dem Aufruf des Dienstes vorgezeigt werden muss.

- Wenn jeder Dienst selbst die Zugriffskontrolle implementiert, können Insellösungen entstehen. Auf der einen Seite kann es vorkommen, dass einige Dienste gar keine Zugriffskontrolle realisieren, insbesondere dann, wenn die Dienste von unterschiedlichen Entwicklern stammen. Auf der anderen Seite können unnötige Redundanzen entstehen. Daher ist es besser, einheitliche Verfahren für alle Dienste anzuwenden.

- Falls die Zugriffskontrolle auch auf Dienste angewendet werden soll, die bereits existieren, müssten diese angepasst werden. Dieser Vorgang ist oft mit erheblichem Aufwand verbunden. Es sollte daher möglich sein, die Mechanismen für die Zugriffskontrolle sowohl auf alte als auch auf neue Dienste anwenden zu können.

- Eine besondere Errungenschaft der eHome-Gruppe besteht darin, dass ein Dienst in mehreren eHomes wiederverwendet werden kann. Es kann aber vorkommen, dass jeder eHome-Betreiber unterschiedliche Formen der Zugriffskontrolle umsetzen möchte. Würde die Zugriffskontrolle im Dienst durchgeführt, müsste der Dienst immer angepasst werden, wenn ein anderes Verfahren gewünscht ist. Stattdessen sollte es einfach möglich sein, die Zugriffskontrolle einem Dienst einfach hinzuzufügen oder auszutauschen. Somit erhält der eHome-Betreiber auch die Möglichkeit, eigene Verfahren unabhängig von den betriebenen Diensten anzuwenden. Damit wird die Flexibilität erhöht. Außerdem können mehrere Verfahren gleichzeitig auf einen Dienst angewendet werden. Beispielsweise hängt die Art der Überprüfung der Zugriffsberechtigung davon ab, ob der Zugriff von einem Benutzer oder von einem Dienst stammt. Der Interceptor-Ansatz erlaubt es, mehrere solcher Verfahren einfach zu kombinieren, ohne dafür die Dienste anpassen zu müssen.

- Ferner sollten die Verfahren für die Zugriffskontrolle leicht wiederverwendbar sein. Hierfür eignet sich die Durchführung im Dienst nicht.

Zusammenfassend kann festgehalten werden, dass die Durchführung der Zugriffskontrolle durch einen Interceptor der Durchführung durch die Dienste vorzuziehen ist.

7.4 Realisierung

In diesem Abschnitt werden Realisierungsdetails der Zugriffskontrolle für eHome-Dienste beschrieben. Dabei wird der Schwerpunkt auf die aspektorientierte Umsetzung des Interceptors gelegt. Zunächst wird jedoch eine Einführung in das Paradigma der aspektorientierten Programmierung gegeben.

7.4.1 Aspektorientierte Programmierung und AspectJ

Die aspektorientierte Programmierung (AOP) ist ein relativ neues Programmierparadigma, das Kiczales et al. erstmals 1997 auf der Konferenz für objektorientierte Programmierung vorgestellt haben [KLM+97]. Ausgangspunkt war die Erkenntnis, dass die objektorientierte Programmierung sich nicht dafür eignet, einige wichtige Entwurfsentscheidungen zu erfassen. Speziell *querschneidende Sachverhalte* (engl. *Crosscutting Concerns*) müssen in jedem Modul untergebracht werden, obwohl sie dort logisch nicht hingehören. Das läuft der Idee der Objektorientierung zuwider [KHH+01]. Ein klassisches Beispiel für eine querschneidende Aufgabe ist das Logging. Aber auch die Zugriffskontrolle, wie sie in diesem Kapitel beschrieben wurde, gehört hierzu. AOP separiert diese querschneidenden Aufgaben von den Modulen und kapselt sie in eigenen Modulen, sodass bei der Programmierung

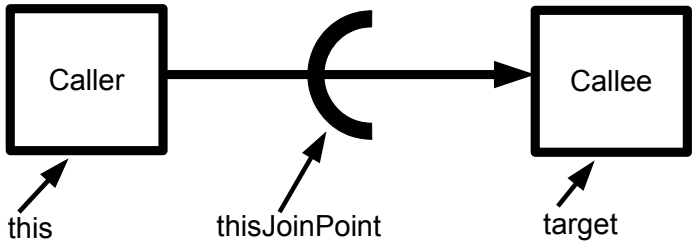


Abbildung 7.5: Ein Joinpoint mit Kontextinformationen.

kein verstreuter Code entsteht. Ausgelagerte querschneidende Aufgaben werden *Aspekte* genannt.

Erweiterungen, welche die Aspektorientierung umsetzen, wurden bereits für viele Programmiersprachen entwickelt. Dazu gehören zum Beispiel AspectJ für Java, AspectC für C/C++ und AspectXML für XML. Im folgenden Abschnitt werden die grundlegenden Konzepte der aspektorientierten Programmierung anhand von AspectJ vorgestellt.

AspectJ wurde von der Firma Xerox entwickelt und setzt die theoretischen Konzepte für die Programmiersprache Java um. AspectJ ermöglicht, existierendem Java-Code an vordefinierten Stellen zusätzlichen Code einzuweben. Eingewobener Code wird unbemerkt von dem Trägersystem automatisch ausgeführt. Um zusätzlichen Code einzuweben, werden folgende drei Konstrukte benötigt:

- **Joinpoint**: Ein Joinpoint ist ein (Zeit-)Punkt im Programm, der über Advices erweitert oder modifiziert werden kann. Mögliche Joinpoints können sein: Methodenaufruf, Methodenausführung oder Variablenzugriff.

- **Pointcut**: Ein Pointcut repräsentiert einen oder mehrere Joinpoints, die auch durch logische Operatoren miteinander verknüpft werden können. Pointcuts ermöglichen den Zugriff auf den Kontext der Joinpoints, etwa die Methodenparameter bei einem Methodenaufruf.

- **Advice**: Advices enthalten den Code, der an einem Pointcut ausgeführt werden soll. Advices treten, abhängig davon, zu welchem Zeitpunkt sie ausgeführt werden sollen, in drei verschiedenen Arten auf: `before`, `after` und `around`. Der `before`-Advice wird vor dem Eintritt in einen Joinpoint, etwa einem Methodenaufruf, ausgeführt. Der `after`-Advice wird entsprechend nach und der `around`-Advice statt des Joinpoints ausgeführt.

An dieser Stelle werden einige grundlegende Syntaxelemente von AspectJ vorgestellt, weil im nächsten Abschnitt konkrete Pointcuts und Advices erläutert werden.

Von Interesse sind hier vor allem die Joinpoints bei Methodenaufrufen und bei der Ausführung von Methodenrümpfen. Die dazugehörigen Schlüsselwörter sind `call` und `execution`. Jeder Joinpoint stellt normalerweise Kontextinformationen zur Verfügung, die

```
1 pointcut selectOne():
2   call(String EhomeRemoteService.getServiceName());
3 pointcut selectAll():
4   call(* EhomeRemoteService.*(..));
5
6 before() : selectAll() {
7   System.out.println(thisJoinPoint.getThis());
8 }
```

Listing 7.1: Beispiele für Pointcuts und Advices.

dazu verwendet werden können, um das weitere Vorgehen zu bestimmen. Abbildung 7.5 zeigt das Beispiel eines Joinpoints mit den zugehörigen Kontextinformationen. Mögliche Kontextinformationen bei Methodenaufrufen (`call`) sind:

- **this**: Der Aufrufer einer Methode kann über die Kontextinformation `this` ermittelt werden.
- **target**: Die aufgerufene Methode hingegen kann über `target` ausgemacht werden.
- **thisJoinPoint**: Schließlich wird mit `thisJoinPoint` der Pointcut mit seinen Argumenten referenziert.

Auch bei JoinPoints für die Ausführung von Methodenrümpfen (`execution`) sind diese Informationen verfügbar. In diesem Fall sind `this` und `target` aber identisch, weil die Ausführung der Methode bereits begonnen wurde.

Beispiele für Pointcuts mit unterschiedlichen Joinpoints sind in Listing 7.1 wiedergegeben. Pointcuts werden durch das Schlüsselwort `pointcut` eingeleitet, gefolgt von einem Namen und optional den Argumenten. Die Signatur ähnelt also der einer Methode in Java. Im Rumpf des Pointcuts werden Joinpoints ausgewählt. Der erste Pointcut `selectOne()` besitzt im Rumpf den Joinpoint `call(String EhomeRemoteService.getServiceName())`, der nur beim Aufruf der Methode `getServiceName()` der Klasse EhomeRemoteService zum Tragen kommt.

Bei der Definition von Joinpoints können aber auch Wildcards verwendet werden. Der Stern (`*`) steht für ein beliebiges Argument und zwei Punkte hintereinander (`..`) für beliebig viele Argumente. Die Wildcards greifen auch bei Klassen- oder Methodendeklarationen. Der zweite Pointcut `selectAll()` in Listing 7.1 kommt daher bei allen Aufrufen von Methoden der Klasse EhomeRemoteService zum Tragen, somit auch beim Aufruf von `getServiceName()`.

In einem Pointcut können auch mehrere Joinpoints miteinander über logische Operatoren verknüpft werden. Zur Verfügung stehen die bekannten Operatoren && (und), || (oder) und ! (nicht).

Ein Advice beginnt mit einem der drei Schlüsselworte `before`, `after` oder `around`, optional gefolgt von Argumenten. Nach einem Doppelpunkt folgt der Bezeichner des Pointcuts, auf den der Advice angewendet werden soll. Advices entsprechen einer Methode und

```
1 pointcut PDAcalls(RequestHandler handler) :
2   // interrupt method calls from PDA
3   execution(* ehome.services.rmi.*.*(..))
4   // notice RMI Controllflow
5   && cflow(execRequestHandler(handler))
6   // don't interrupt calls from service to another
7   && !cflow(callServiceInterfaceMethod()));
8
9 pointcut execRequestHandler(RequestHandler handler) :
10   // notice rmi method invocation
11   execution(* RequestHandler.executeRequest(RMIRequest))
12   // make context available
13   && this(handler);
```

Listing 7.2: Pointcut aus dem InterceptorAspect für Benutzerzugriffe vom Handheld.

werden über die übliche Java-Syntax spezifiziert. Ein besonderer Schlüssel ist `thisJoinPoint`, mit dem auf den Kontext des Pointcuts zugegriffen werden kann (s. Abbildung 7.5). Nur durch diesen Schlüssel und über die Argumente des Advices kann auf Objekte des Programmablaufs zugegriffen werden. Der in Listing 7.1 dargestellte `before`-Advice wird auf den Pointcut `selectAll()` angewendet und greift daher bei allen Methodenaufrufen der Klasse EhomeRemoteService. Vor der Ausführung der jeweiligen Methode ermittelt sie den Aufrufenden (`thisJoinPoint.getThis()`) der Methode und gibt ihn aus.

Für eine ausführliche Darstellung der Sprache AspectJ sei zum Beispiel auf [Boe06] verwiesen. Im nächsten Abschnitt wird erläutert, wie AspectJ für die Implementierung des Interceptors eingesetzt wurde.

7.4.2 Aspektorientierte Realisierung des Interceptors

Dieser Abschnitt befasst sich mit der aspektorientierten Realisierung des Interceptors. Für die Implementierung wurde das *AspectJ Project* [KHH+01] mit der Einbettung *AspectJ Development Tools (AJDT)* [Ecl10b] in Eclipse verwendet. Dadurch war eine einfache Anwendung der aspektorientierten Programmierung innerhalb der Entwicklungsumgebung des eHome-Prototyps möglich.

Für die Realisierung des Interceptors wurde im Rahmen dieser Arbeit ein Aspekt mit dem Namen InterceptorAspect entwickelt[4]. Dieser ist für die Durchführung der Zugriffskontrolle auf eHome-Dienste zuständig. Es können jederzeit neue Aspekte eingeführt werden und als Klassenverband ein- und ausgeschaltet werden. Die Konfiguration der Aspekte wird in einer separaten XML-Datei, der *aop.xml* geregelt. Eine Besonderheit besteht darin, dass Aspekte ausschließlich vor der Inbetriebnahme der Dienste aktiviert oder deaktiviert werden können. Dies liegt daran, dass AspectJ momentan nur *statisches Weben*, also das Einweben des Aspekt-Codes vor der Kompilierung, unterstützt. Es gibt jedoch erste Ansätze zur Erweiterung von AspectJ um die Möglichkeit des *dynamischen Webens* (s. [AN08, Bon04]).

[4]Die Implementierung eines Aspekts mit AspectJ gleicht der einer Java-Klasse.

```
1 before(RequestHandler handler) : PDAcalls(handler) {
2   EHomeRemoteService service
3    = (EHomeRemoteService) thisJoinPoint.getTarget();
4   Object credential
5   = handler.prepareShowProof(service.getRemoteServiceName());
6   boolean result
7    = Authenticator.getInstance()
8     .verifySessionCredential(credential)
9   if (!result) {
10     throw new ServiceAuthenticationException(thisJoinPoint
11      .getThis(), thisJoinPoint.getTarget());
12   } else {
13     System.out.println("PDA Service "
14      + thisJoinPoint.getTarget()
15      + " authentication passed");
16   }
17 }
```

Listing 7.3: Advice für Aufrufe vom Handheld.

Der InterceptorAspect befindet sich in einem OSGi-Bundle (kurz Aspects-Bundle), das wie die übrigen Bundles von der zugrunde liegenden OSGi-Dienstplattform ausgeführt wird. Auch die Kommunikation zwischen den Bundles wird von dieser Plattform geregelt. Das Aspects-Bundle muss aus zuvor genannten Gründen gestartet werden, bevor das eHome-Gateway eingerichtet wird. Eine Konfiguration stellt den Start des Aspects-Bundles sicher. Sollte dieses Bundle nicht gestartet werden, kann keine Zugriffskontrolle durchgeführt werden. Nach dem Start ist dann ein Abschalten des Aspects-Bundles und somit das Ausschalten der Zugriffkontrolle nicht mehr möglich. Der eHome-Betreiber muss beim Start darauf achten, dass die Aspekte gestartet wurden. Er kann somit darauf vertrauen, dass der Interceptor zur Laufzeit nur noch die Dienstzugriffe erlaubt, die von berechtigen Benutzern oder Diensten erfolgen.

Ein Aspekt setzt sich aus einem Pointcut und einem oder mehreren Advices zusammen. Im vorliegenden Fall existieren drei Pointcuts. Sie fangen die Aufrufe auf Dienste von verschiedenen Ausgangspunkten ab. Ein Pointcut ist für die Kontrolle von Zugriffen zwischen Diensten verantwortlich. Die übrigen Beiden sind für Zugriffe zuständig, die von Benutzern getätigt werden. Dabei wird zwischen solchen Zugriffen unterschieden, die von einem Handheld getätigt wurden, und solchen, die über das Bedienen eines Geräts im eHome erfolgen.

Listing 7.2 zeigt den Pointcut `PDAcalls` mit einem RequestHandler als Parameter. An diesem Beispiel wird die Funktionsweise des Aspektes erklärt. Als Erstes sei angemerkt, dass der RequestHandler zur Ausführungszeit verfügbar ist (Zeile 1). Wie in Abschnitt 5.3.4 erläutert wurde, nimmt der RequestHandler Anfragen über SimpleRMI (diese kommen von Handhelds) entgegen und leitet sie weiter an das Zielobjekt. Umgekehrt können über den RequestHandler auch Nachrichten an das Handheld gesendet werden. Weil dieser Pointcut für die Zugriffskontrolle anhand eines Session-Credentials eingesetzt wird, muss der Interceptor

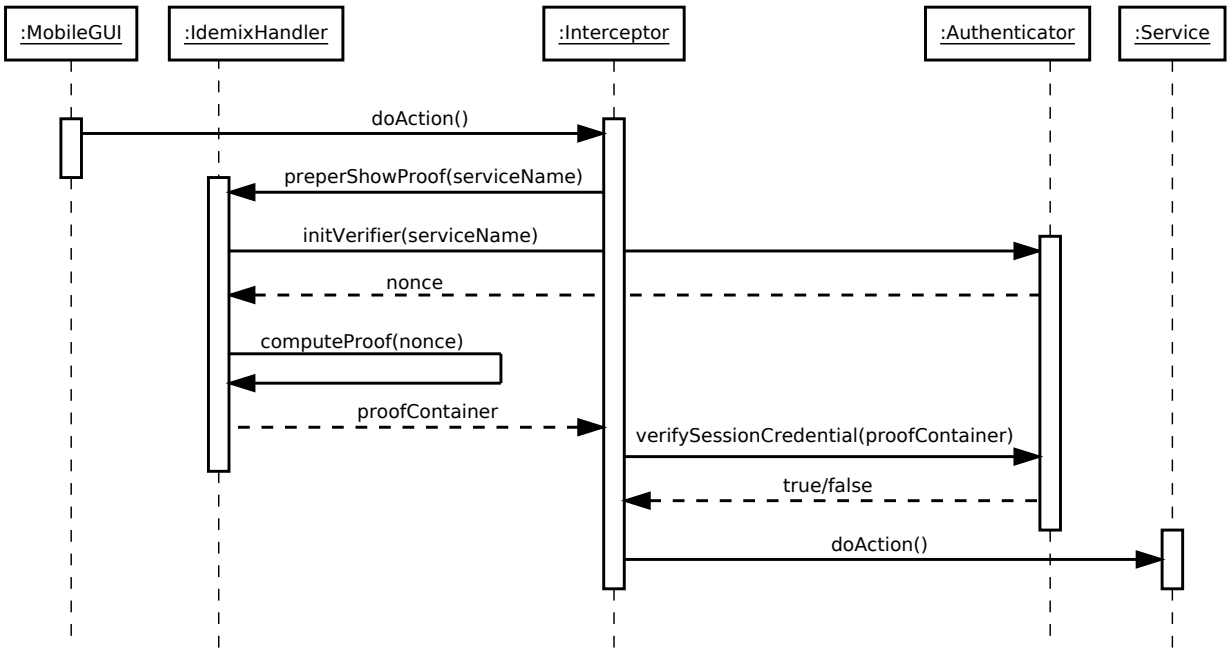


Abbildung 7.6: Ablauf der Zugriffskontrolle für einen Benutzerzugriff auf Basis eines Session-Credentials.

den entsprechenden Berechtigungsnachweis von dem Handheld anfordern. Genau hierfür wird er den RequestHandler benutzen, wie später anhand von Listing 7.3 beschrieben wird.

In Zeile 3 von Listing 7.2 werden die Methoden bestimmt, deren Ausführung der Interceptor überwachen soll. Konkret handelt es sich hier um die Methoden aller Klassen aus dem Paket ehome.services.rmi. Dabei steht das erste '.*' für „alle Klassen", die in diesem Paket liegen. Dieses Paket enthält alle Schnittstellen für Dienste, die über SimpleRMI, also von Handhelds, aufrufbar sein sollen. Ein Beispiel für solch eine Schnittstelle ist das Interface LightFollowsPerson aus Abbildung 5.24, das die Funktionalität des `Lichtdienstes` spezifiziert. Das anschließende '.*(..)' steht für alle möglichen Methoden aus diesen Schnittstellen, dabei kann die Methodensignatur beliebig sein. Somit greift dieses Pointcut bei jeder Ausführung einer Methode, die von einem Dienst realisiert wird, die eine Schnittstelle aus dem Paket ehome.services.rmi implementiert.

In Zeile 5 wird ein weiterer Pointcut referenziert, der in Zeile 9 mit dem Namen `execRequestHandler` definiert wird und genau dann greift, wenn die Ausführung der abgefangenen Methode von dem RequestHandler, genauer gesagt von der Methode `executeRequest` des RequestHandlers, angestoßen wurde (Zeile 11). Dieser Pointcut greift in Zeile 13 über das `this`-Schlüsselwort auf den Kontext der RequestHandler-Instanz zu. Da er innerhalb des ersten Pointcuts eingebettet ist, leitet dieser die Instanz an den zugehörigen Advice weiter, der in Kürze beschrieben wird. Dort besteht dann die Möglichkeit, die Instanz des RequestHandlers zu nutzen.

Die letzte Zeile des Pointcuts PDAcalls stellt sicher, dass Aufrufe zwischen Diensten von diesem Pointcut nicht berücksichtigt werden. Das ist deshalb notwendig, weil unterschieden

```
1 pointcut callServiceInterfaceMethod() :
2   // interrupt all methods from ehome.services.*
3   (call(* ehome.services.*.*(..))
4   // or from .rmi and subpackages
5   || call(* ehome.services.rmi.*.*(..)))
6   // when the call comes from another service only
7  && within(ehome.services..*);
```

Listing 7.4: Pointcut für Zugriffe von Diensten untereinander.

werden muss, ob der Aufruf vom Handheld direkt von dem Benutzer über die MobileGUI oder von einem auf dem Handheld ausgeführten Dienst stammt. Für den zweiten Fall ist das Pointcut `callServiceInterfaceMethod` zuständig. Genauer gesagt ist es speziell für die Kontrolle von Dienstzugriffen untereinander zuständig, unabhängig davon, ob sich der aufrufende Dienst auf dem Handheld oder im eHome befindet.

Zu jedem der Pointcuts wurde auch ein Advice entwickelt. Der zum obigen Pointcut gehörende Advice ist in Listing 7.3 dargestellt. Weil es sich hier um ein `before`-Advice handelt, greift dieser vor der Ausführung der im Pointcut spezifizierten Methoden ein. Der Ablauf innerhalb des Adivces ist zum besseren Verständnis in Abbildung 6.13 visualisiert, anhand der auch die folgende Erläuterung durchgeführt wird. Dort ist der InterceptorAspect kurz als Interceptor gekennzeichnet.

Im vorliegenden Advice wird zunächst das Ziel des Aufrufes, also der aufgerufene Dienst ermittelt. Das geschieht über das Schlüsselwort `thisJoinPoint.getTarget()` (Zeile 3). Anschließend fordert der Interceptor den IdemixHandler auf dem Handheld auf, den Nachweis für die Zugriffsberechtigung auf diesen Dienst zu liefern. Hierfür wird dem RequestHandler der Name des aufgerufenen Dienstes übergeben (Zeile 4). Wie im Sequenzdiagramm dargestellt, tauschen der IdemixHandler auf dem Handheld und der Authenticator im eHome die nötigen Informationen aus, damit der IdemixHandler den Nachweis berechnen kann. Ist der Nachweis (`proofContainer`) beim Interceptor angekommen, übermittelt dieser ihn an den Authenticator zur Verifizierung weiter (Zeilen 7-8). Bei einer positiven Antwort wird eine Bestätigung ausgegeben (Zeilen 13-15) und der Advice beendet. Dadurch wird der Kontrollfluss an der angehaltenen Stelle wie üblich fortgeführt.

Im Sequenzdiagramm ist dies so dargestellt, dass der Aufruf von der MobileGUI nicht direkt beim Dienst landet, sondern zunächst von dem Interceptor abgefangen wird. Dieser leitet ihn bei einer positiven Antwort an den Dienst weiter. Das entspricht nicht der Realität, sondern ist für das bessere Verständnis auf diese Art modelliert. Zu beachten ist hier, dass weder die MobileGUI noch der Dienst eine Kenntnis vom Interceptor haben. Die Zugriffskontrolle findet sowohl für den Aufrufenden als auch für den Aufgerufenen verdeckt statt, sie bemerken die Unterbrechung also nicht.

Falls die Überprüfung des Nachweises eine negative Antwort liefert, wird eine Ausnahme ausgelöst (Zeilen 9-11). Im Rahmen dieser Ausnahme wird eine Fehlermeldung ausgegeben und der Kontrollfluss an dieser Stelle beendet. Der Aufruf erreicht somit den Dienst nicht mehr. In diesem Fall erfährt der Aufgerufene nichts von dem Aufruf.

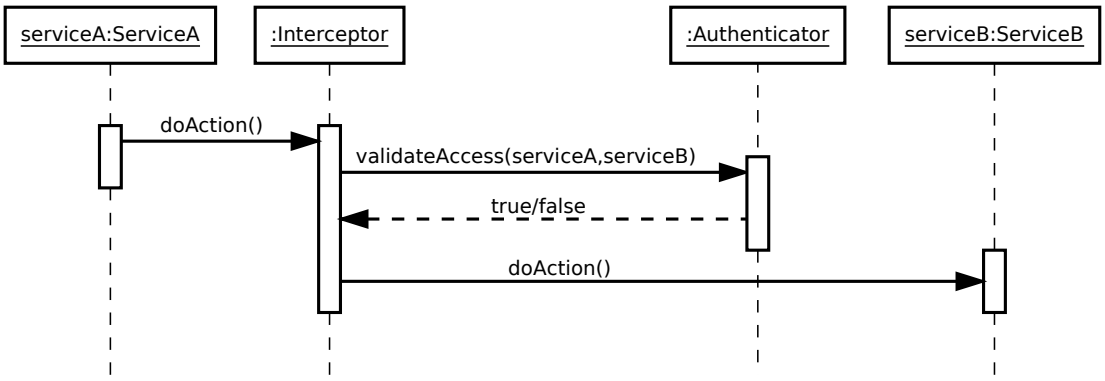


Abbildung 7.7: Ablauf der Zugriffskontrolle für einen Zugriff zwischen zwei Diensten.

Der Pointcut, der für die Überprüfung von Zugriffen von Diensten auf Dienste auf Basis von Dienstrollen zuständig ist, ist in Listing 7.4 dargestellt. Dort werden alle Methoden der Pakete ehome.services und ehome.services.rmi abgefangen. Dabei stellt der Ausdruck `within (ehome.services..*)` sicher, dass der Aufrufende ein Dienst sein muss, der in dem Paket ehome.services enthalten ist.

Der schematische Ablauf der dort greifenden Zugriffskontrolle ist in Abbildung 7.7 dargestellt. In diesem Beispiel fängt der Interceptor den Aufruf der Methode `doAction()` von serviceB ab, den der Dienst serviceA aufruft. Zunächst lässt der Interceptor den Authenticator überprüfen, ob dieser Zugriff erlaubt werden soll. Falls die Antwort positiv ist, wird der Aufruf durchgelassen. Falls nicht, wird der Aufruf abgebrochen und es wird eine Fehlermeldung ausgegeben.

Hierdurch wird die Flexibilität dieses Ansatzes deutlich. Für die Aufrufe derselben Methode können, abhängig von dem Aufrufenden, unterschiedliche Verfahren zur Zugriffskontrolle zum Einsatz kommen. Greift nämlich der Benutzer über die MobileGUI auf seinem Handheld auf einen Dienst im eHome zu, wird die Überprüfung anhand eines Session-Credentials durchgeführt. Greift jedoch ein Dienst im eHome auf einen anderen Dienst zu, der möglicherweise mit dem von dem Benutzer benutzten Dienst identisch ist, wird die Überprüfung anhand von Dienstrollen durchgeführt. Der eHome-Betreiber kann durch Anpassung des InterceptorAspects zudem eigene Überprüfungsverfahren hinzufügen, um auf sich ändernde Umstände zu reagieren.

7.5 Zusammenfassung

In diesem Kapitel wurde ein Ansatz beschrieben, der für den Schutz von eHome-Diensten vor unbefugtem Zugriff entwickelt wurde. Dabei wurden zwei Arten von Zugriffen unterschieden, nämlich von Benutzern und von Diensten.

Für den Zugriff von Benutzern wurden verschiedene Mechanismen entwickelt. Abhängig davon, welchen Grad der Anonymisierung sich die Benutzer wünschen, können sie zwischen diesen Mechanismen wählen. Diese Mechanismen lassen sich grob in rollenbasierte Zugriffskontrolle und in Zugriffskontrolle auf Basis anonymer Credentials unterteilen.

Für den Zugriff von Diensten untereinander wurde auch die rollenbasierte Zugriffskontrolle angewendet. Hierbei wurde jedem Dienst automatisch eine Dienstrolle zugeordnet, die Zugriffsrechte auf die Dienste enthält, mit denen dieser in der Dienstkomposition im eHome-Modell verbunden ist.

Anschließend wurde der Interceptor beschrieben, der für die Durchführung der Zugriffskontrolle in Zusammenarbeit mit dem Authenticator zuständig ist. Aufgrund seiner besseren Unterstützung der Wartbarkeit und Änderbarkeit wurde dieser Ansatz der Durchführung der Zugriffskontrolle in den Diensten vorgezogen. Abgeschlossen wurde dieses Kapitel mit den Einzelheiten zur aspektorientierten Realisierung des Interceptors.

Kapitel 8

Werkzeugunterstützung und Demonstratoren

In den vorangegangenen Kapiteln wurden bereits Details zur Realisierung der einzelnen Konzepte erläutert, die im Rahmen dieser Arbeit entwickelt wurden. Dabei wurden auch die Erweiterungen beschrieben, die an dem bestehenden eHome-Prototyp vorgenommen wurden. In diesem Kapitel werden die im Rahmen dieser Arbeit entwickelten Werkzeuge beschrieben. Dabei handelt es sich jeweils um ein Werkzeug für eHomes und eins für Handhelds. Ferner werden mehrere Demonstratoren vorgestellt, die für die Evaluierung der entwickelten Konzepte Werkzeuge herangezogen wurden.

8.1 Werkzeugunterstützung im eHome

Eines der Werkzeuge, das im Rahmen dieser Arbeit erstellt wurde, ist der eHomeAdministrator. Es erlaubt die Administration von eHomes bzgl. der Verwaltung von Benutzern, deren Daten und Identitäten sowie der Verwaltung von Zugriffsrollen. Das Werkzeug ist vollständig in Java implementiert. Es kann sowohl selbstständig (engl. *Stand-Alone*) ausgeführt als auch als Eclipse-Plug-In in den eHome-Prototyp integriert werden.

Der eHomeAdministrator basiert dabei auf dem Grundmodell, das in Abbildung 8.1 dargestellt ist. Dieses Modell stellt die Zusammenhänge zwischen Diensten, Rollen, Benutzern, Identitäten und Attributen dar. Abgeleitet aus dem rollenbasierten Zugriffskontrollmodell können Benutzern und Diensten Rollen zugewiesen werden. Während jeder Dienst genau eine Rolle haben kann (`has`-Beziehung), können Benutzern mehreren Rollen zugeordnet sein. In jeder Rolle ist eine Menge von Diensten enthalten, für die der Besitzer der Rolle ein Zugriffsrecht hat (`accessTo`-Beziehung). Ein Benutzer hat ferner eine Hauptidentität und mehrere Teilidentitäten. Während die Hauptidentität mindestens ein Attribut enthält, können Teilidentitäten beliebig viele Attribute enthalten. Dabei stellen die Attribute von Teilidentitäten eine Teilmenge der Attribute der Hauptidentität dar. Die Attribute enthalten einzelne Aussagen bzw. Informationen über den Benutzer.

Im Folgenden wird die Funktionalität dieses Werkzeugs anhand seiner Benutzeroberfläche näher beschrieben.

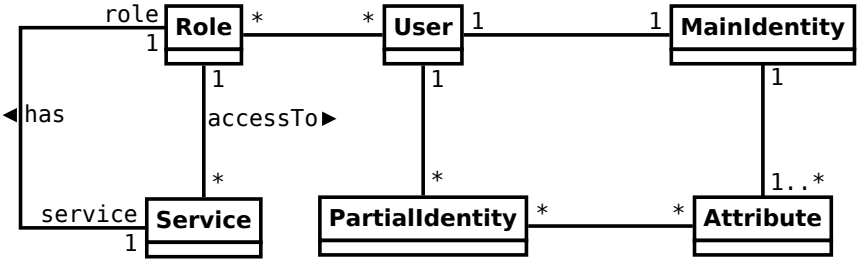


Abbildung 8.1: Zusammenhang zwischen Diensten, Rollen, Benutzern, Identitäten und Attributen.

8.1.1 Benutzer

In diesem Abschnitt wird der Teil des Werkzeugs beschrieben, der die Verwaltung der Benutzer einschließlich der Rechtevergabe an diese ermöglicht. Abbildung 8.2 zeigt den eHomeAdministrator, nachdem der Menüpunkt `Users` gewählt wurde. Beispielhaft wurde die Benutzerstruktur eines eHomes angelegt, wie sie in einem Einfamilienhaus aussehen könnte. Zu sehen ist eine Baumstruktur, die die Benutzer des eHomes auflistet. Die Ordnersymbole beschreiben eine Kategorie und die anderen jeweils eine Person. Kategorien sind rein für die Strukturierung vorhanden, um dem Anwender die Übersicht zu erleichtern. Benutzer und Kategorien können innerhalb des Baumes beliebig verschoben werden. Mittels der Toolbar über dem Baum können Änderungen über das Diskettensymbol gespeichert werden, oder es können Kategorien wie auch Benutzer neu angelegt oder gelöscht werden.

Wird nun beispielsweise der Benutzer *Rosanne* in der Baumstruktur ausgewählt, erscheint rechts neben dem Baum eine Detailansicht des Benutzers, die zusätzliche Informationen bereithält. Der Name des Benutzers kann hier geändert werden. Das Werkzeug prüft bei einer Namensänderung diesen auf Eindeutigkeit im System. Darunter befindet sich die Übersicht der Rollenzugehörigkeit des Benutzers. Hierfür ist eine neue Komponente implementiert worden, die ermöglicht, auf einfache Weise einem Benutzer Rollen zuzuordnen. Unter dem Punkt `Available Roles` werden alle Rollen aufgelistet, die dem Benutzer zugeordnet werden können und noch nicht zugeordnet wurden. Rechts davon werden unter dem Punkt `User has Roles` die Rollen aufgelistet, die dem Benutzer bereits zugeordnet wurden. Um den Anwender bei der Auswahl der Rollen hinsichtlich der enthaltenen Dienste zu unterstützen, kann mit einem Tooltip eine Liste mit den Diensten angezeigt werden, deren Zugriffsrechte in der Rolle enthalten sind. In Abbildung 8.2 wurde beispielsweise das Tooltip der Rolle *Children* ausgelöst, der die Dienste `Lighting` und `Music` anzeigt. Abgeleitet aus allen dem Benutzer zugeordneten Rollen werden in der unteren Tabelle alle Dienste aufgelistet, auf die der Benutzer *Rosanne* Zugriff hat. In diesem Beispiel hat der Benutzer nur die Rolle *Guests* zugewiesen bekommen, welche die Zugriffsrechte auf die Dienste *Lighting*, *Television* und *Temperature* enthält. In der Tabelle wird neben jedem Dienst noch zusätzlich eine Liste der Rollen aufgeführt, die die Zugriffsrechte auf diesen enthalten.

Die Rollenzugehörigkeit der Benutzer wird vom eHomeAdministrator in den Baustein `Access Roles` übertragen, der in Abbildung 7.2 dargestellt ist. Nach seiner Anmeldung

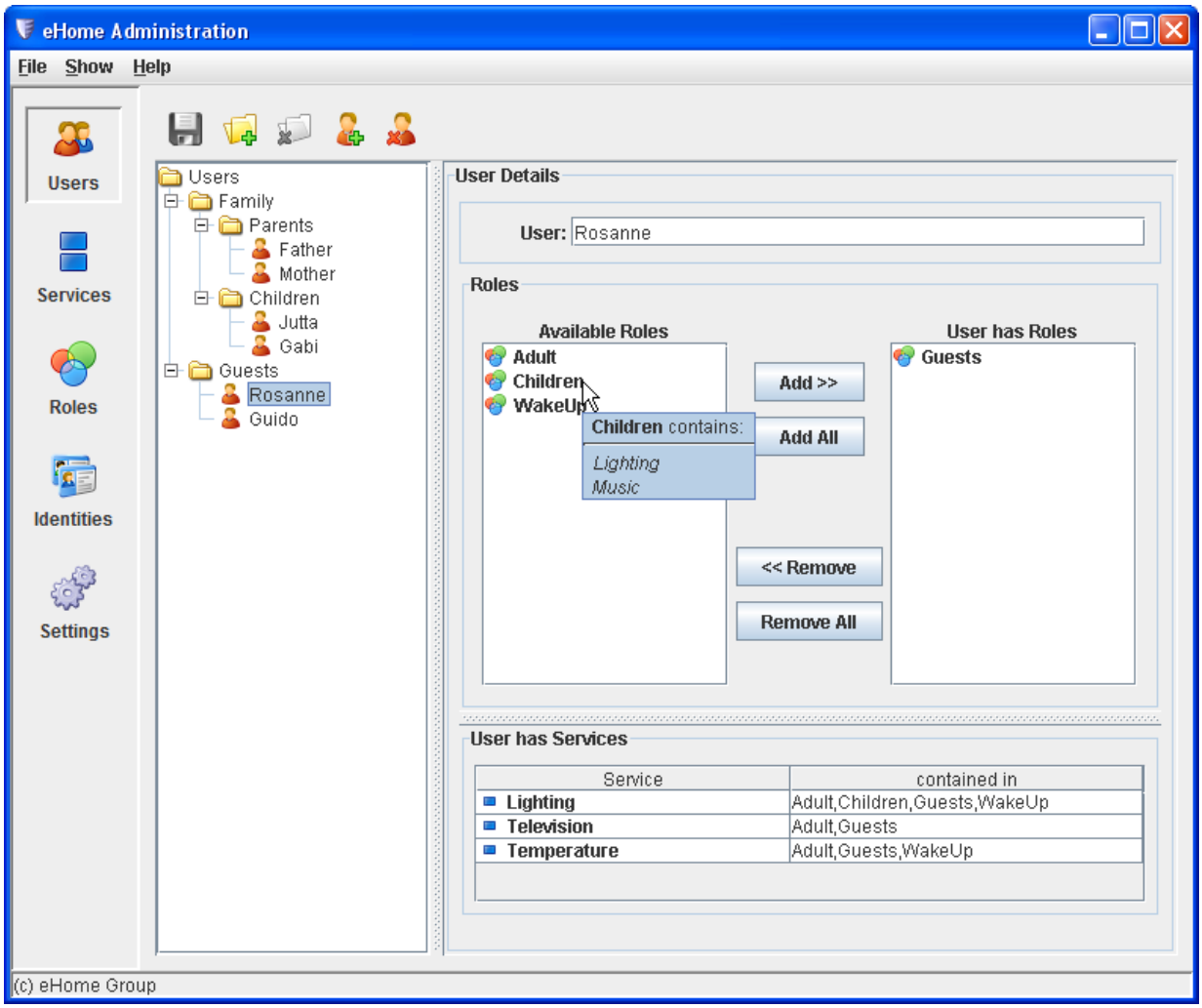


Abbildung 8.2: eHomeAdministrator: Benutzerverwaltung.

beim eHome wird dem Benutzer eine dieser Rollen zugeordnet. Er kann entsprechend der Zugriffsrechte in dieser Rolle auf die eHome-Dienste zugreifen. Dabei kann ein Benutzer zur selben Zeit immer nur mit einer Rolle aktiv sein.

8.1.2 Identitäten

Die Verwaltung von Identitäten kann nach der Auswahl des Menüpunktes `Identities` durchgeführt werden. Hier können Identitätsattribute und komplette Identitäten eines Benutzers angelegt, bearbeitet und gelöscht werden. In Abbildung 8.4 wurde erneut der Benutzer *Rosanne* ausgewählt. In der Detailansicht auf der rechten Seite ist die Identität `Sunny` ausgewählt. In der unteren Tabelle `Attributes` sind die Identitätsattribute aufgelistet. Neben dem Attributnamen steht der Attributwert. Dabei sind alle Attribute angezeigt, die in der Hauptidentität enthalten sind. Die Attribute der ausgewählten Identität sind mit einem Häkchen gekennzeichnet.

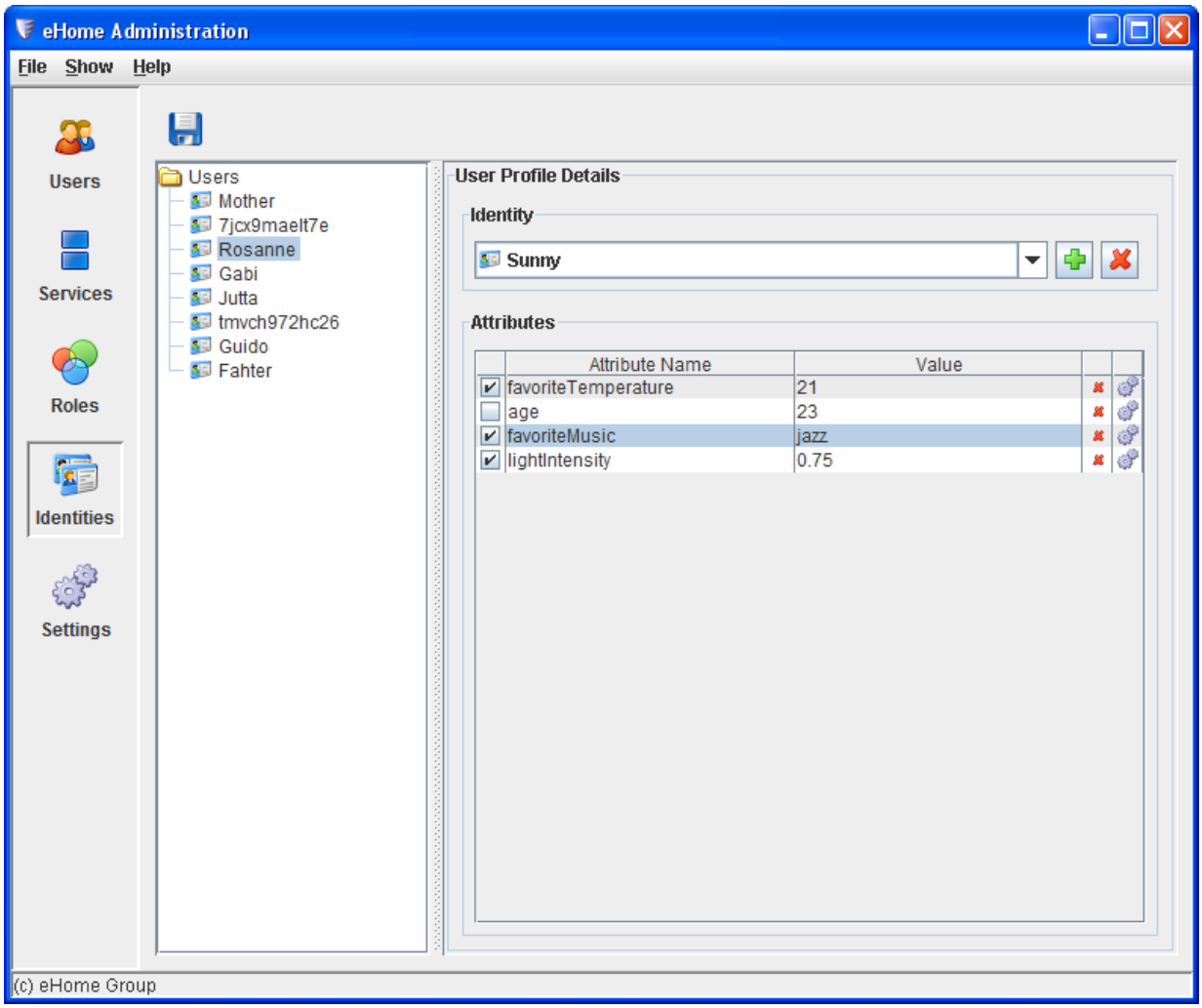


Abbildung 8.3: eHomeAdministrator: Identitäten.

Der Anwender kann einfach neue Attribute anlegen oder löschen und diese zur gewählten Identität hinzufügen bzw. aus dieser entfernen. Falls der Anwender ein Attribut anlegt oder bearbeitet, erscheint das in Abbildung 8.4 gezeigte Dialogfenster, der zwei Reiter bietet. Im ersten Reiter können der Name und der Wert des Attributs bearbeitet werden. Im ausgewählten zweiten Reiter werden die Dienste aufgelistet, die auf dieses Attribut zugreifen dürfen. Im dargestellten Beispiel handelt es sich um das Attribut `favoriteMusic`, auf den der Dienst `Hifi Service` zugreifen darf.

Wie in Abschnitt 4.1 bereits erwähnt wurde, interagiert der eHomeAdministrator mit dem Benutzermodell. Die mit diesem Werkzeug bearbeiteten Daten werden im Benutzermodell abgelegt. Umgekehrt werden die Daten aus dem Benutzermodell mit dem eHomeAdministrator visualisiert. Dabei wird unterschieden zwischen Benutzern, deren Daten im eHome persistent gespeichert werden und dem eHome über eine längere Zeit bekannt sind, und solchen, deren Daten nach einer beendeten Sitzung gelöscht werden. In der Benutzerliste tauchen beispielsweise zwei Benutzernamen auf, die zufällig generiert wurden. Diese Benutzernamen

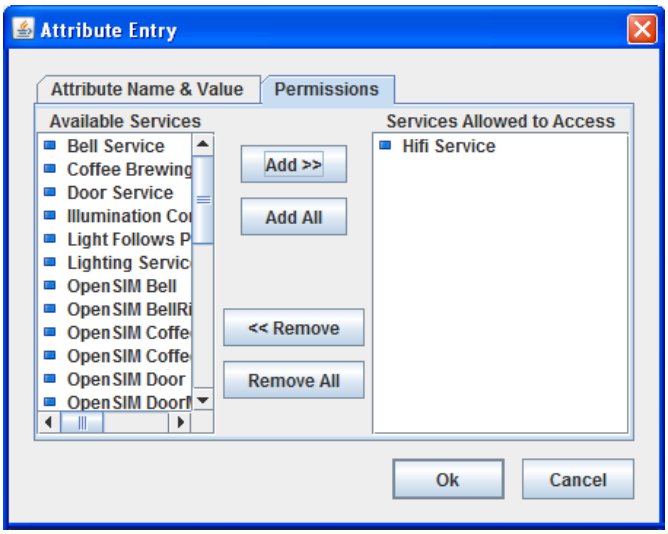


Abbildung 8.4: eHomeAdministrator: Bearbeiten eines Identitätsattributs.

waren in Abbildung 8.2 nicht dargestellt, weil sie nicht zu den Benutzern gehören, deren Daten in diesem eHome persistent gespeichert werden. Hierbei handelt es sich um Benutzer, die sich anonym authentifiziert haben und ein zufälliges Pseudonym erhalten haben. Die Identitäten dieser Benutzer können zwar eingesehen, aber nicht bearbeitet werden. Sie bearbeiten ihre Identitäten ausschließlich über ihre Handhelds. Der eHomeAdministrator ist vielmehr für die komfortable Verwaltung der eigenen Benutzerdaten zu Hause vorgesehen. Diese Daten können dann mit dem eigenen Handheld synchronisiert werden und im Rahmen der Inter-eHome-Mobilität weiter verwendet werden (s. Abschnitt 5.2.1). Weiter unten wird beschrieben, wie die Werkzeugunterstützung für die Verwaltung der Daten auf den Handhelds aussieht.

8.1.3 Dienste

Der eHomeAdministrator präsentiert sich nach Auswahl des Menüpunktes `Services` ähnlich wie unter dem Menüpunkt `Users`. In einer Baumstruktur werden alle verfügbaren Dienste des eHomes aufgelistet. Hier können jedoch nur Kategorien zur besseren Übersicht eingefügt und wieder gelöscht werden, jedoch nicht neue Dienste angelegt werden. Für die Konfigurierung und das Deployment von eHome-Diensten ist wie bereits erläutert der eHomeConfigurator zuständig. Der eHomeAdministrator hingegen visualisiert die Informationen in Bezug auf die Zugriffskontrolle. Für jeden Dienst kann sich der Anwender anzeigen lassen, welche Rolle dem ausgewählten Dienst zugeordnet ist und auf welche anderen Dienste er zugreifen darf. Die Rollen werden, wie in Abschnitt 7.2 beschrieben, automatisch aus der Dienstkomposition abgeleitet, die im eHome-Modell enthalten ist.

In Abbildung 8.5 wurde in der Baumansicht der Dienst `Wakeup Service` ausgewählt. Auf der rechten Seite ist dargestellt, dass dieser Dienst die gleichnamige Rolle innehat. Dies ist in der Liste `Service has Roles` dargestellt. Weiter unten in der Liste `Service has access to` sind die Dienste aufgelistet, die in der zugehörigen Rolle enthalten sind

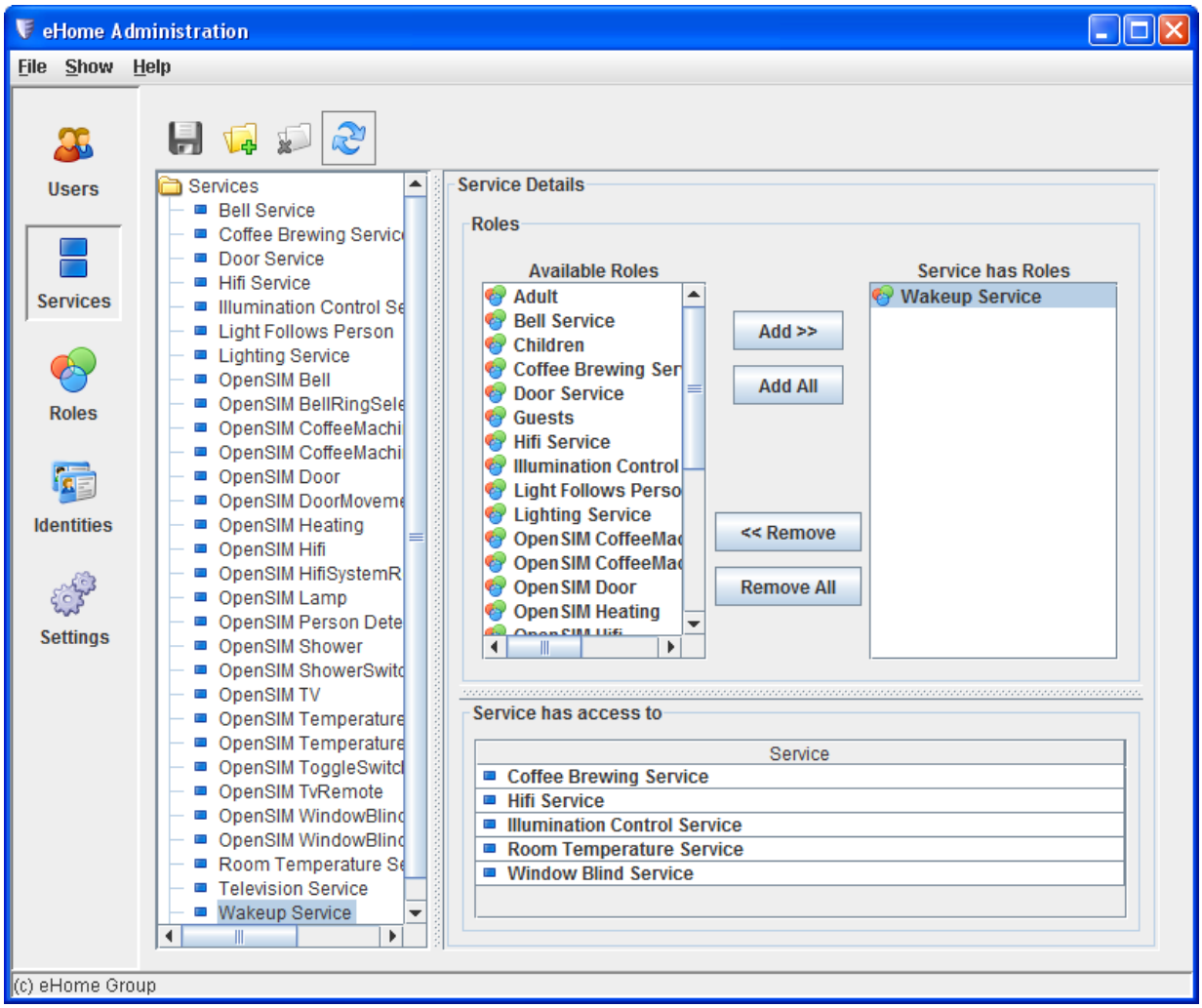


Abbildung 8.5: eHomeAdministrator: Dienste.

und auf die der `Wakeup Siervice` zugreifen darf. Es handelt sich hierbei genau um die Unterdienste, die für die Realisierung der Funktionalitäten des gewählten Dienstes nötig sind, wie etwa das Aufwecken mit Musik und Licht oder das Vorheizen des Badezimmers vor dem Aufwecken.

8.1.4 Rollen

Durch die Wahl des Menüpunktes `Roles` wird die Rollenverwaltung aufgerufen. Hier kann festgelegt werden, welche Zugriffsrechte auf Dienste in den Rollen enthalten sein sollen. Abbildung 8.6 zeigt, wie sich der eHomeAdministrator bei der Wahl einer bestimmten Rolle verhält. Der Baum listet die vorhanden Rollen auf, wobei, wie schon bei der Benutzerverwaltung, die Rollen in Kategorien geordnet werden können. Darüber hinaus können mittels der Menüpunkte auf der Toolbar oberhalb des Baumes neue Rollen angelegt oder bestehende gelöscht werden.

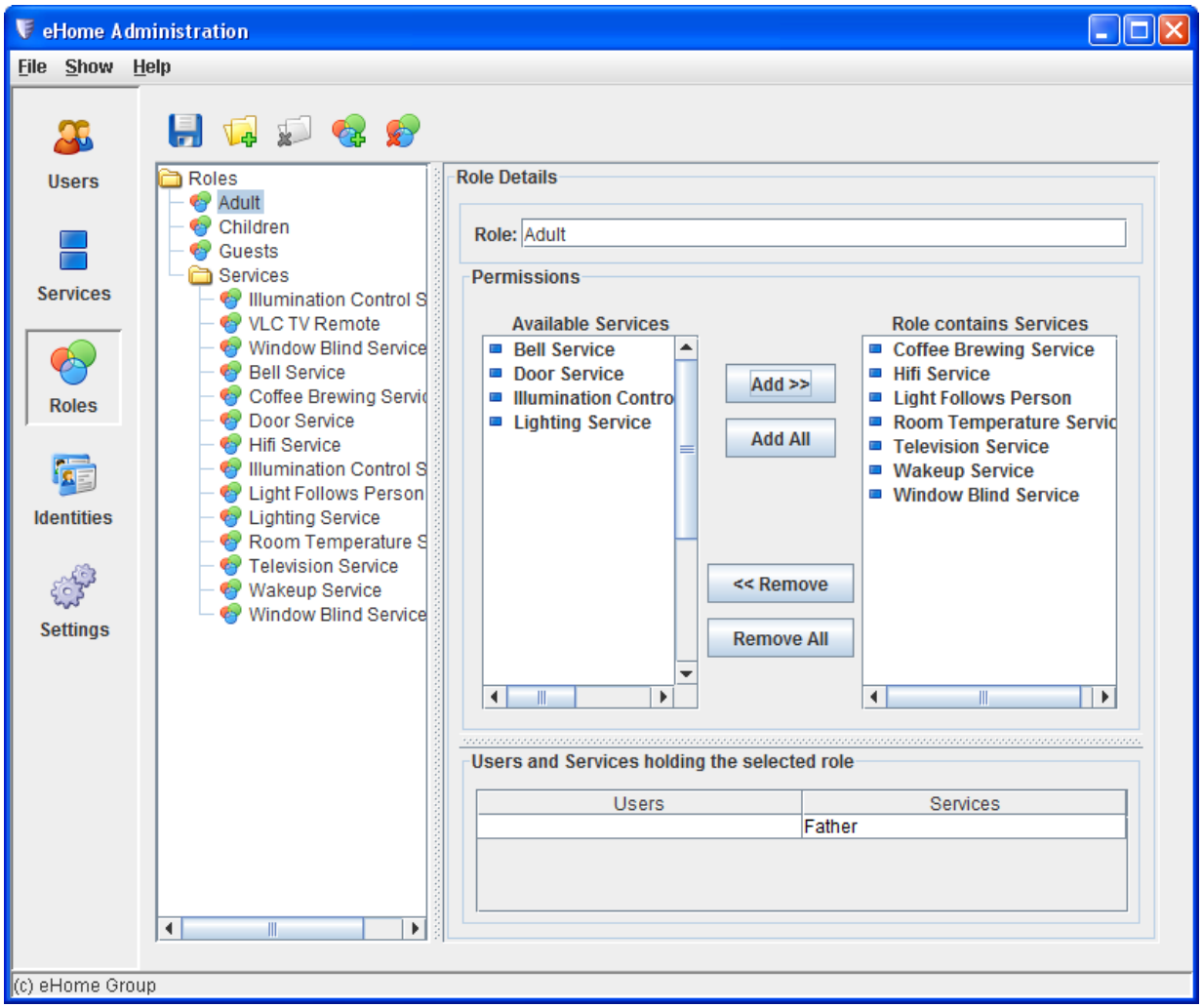
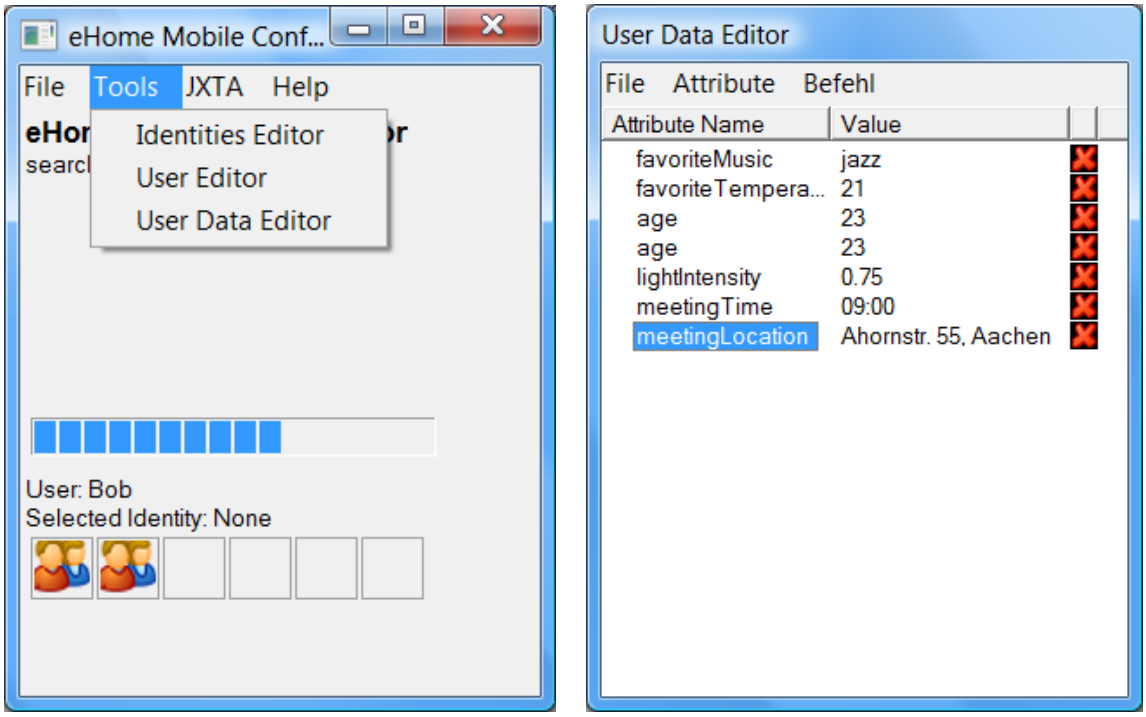


Abbildung 8.6: eHomeAdministrator: Rollen.

In Abbildung 8.6 sind Rollen sowohl für Benutzer als auch für Dienste aufgelistet. Letztere sind dabei in einer eigenen Kategorie zusammengefasst. Ausgewählt wurde die Rolle `Adult`. In der Detailansicht zu dieser Rolle kann der Name geändert und durch Auswahl von Diensten bestimmt werden, auf welche Dienste ein Benutzer oder ein Dienst zugreifen darf, der diese Rolle innehat. Die linke Liste im Bereich `Permissions` zeigt alle Dienste an, die zur Verfügung stehen und noch nicht in der Rolle enthalten sind. Die rechte Liste hingegen enthält die Dienste, deren Zugriffsrechte mit der Rolle verknüpft sind. Dabei sind nur die Top-Level-Dienste dargestellt, die Einbeziehung der Unterdienste bis auf die Ebene der Basisdienste geschieht dann implizit durch den Authenticator, wie in Abschnitt 7.1.3 erläutert wurde. Die untere Tabelle zeigt an, welche Benutzer oder Dienste diese Rolle annehmen können. Im gewählten Beispiel ist dies nur der Benutzer `Father`.



(a) Startbild. (b) Identitätsattribute.

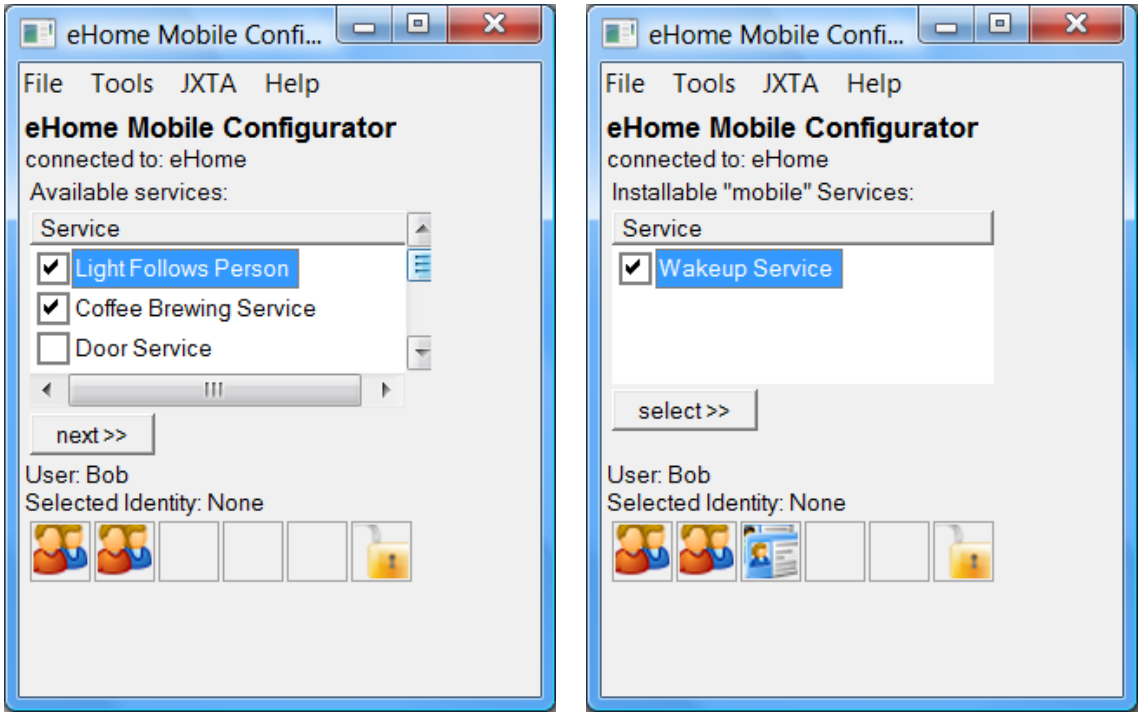
Abbildung 8.7: MobileGUI: Startbild und Identitätsattribute.

8.2 Werkzeugunterstützung auf dem Handheld

Bereits in Abschnitt 5.3.2.2 wurden die Komponenten des Prototyps für Handhelds beschrieben, der im Rahmen dieser Arbeit entwickelt wurde. In diesem Abschnitt wird die Benutzeroberfläche dieses Prototyps, die MobileGUI, beschrieben. Zunächst werden alle wichtigen Screenshots anhand eines Emulators gezeigt. Zum Abschluss dieses Abschnitts werden dann auch einige Bilder des Prototyps auf den eingesetzten PDAs gezeigt.

Nach dem Start des Prototyps auf dem Handheld wird ein Discovery-Mechanismus angestoßen, der automatisch in regelmäßigen Intervallen nach eHomes in Reichweite sucht. Abbildung 8.7(a) zeigt das Startfenster der MobileGUI. Im unteren Bereich der Abbildung ist ein Balken zu erkennen, der auf den Suchmodus hinweist, es ist also noch kein eHome gefunden worden. Ferner ist angezeigt, dass dieses Handheld dem Benutzer Bob gehört und aktuell noch keine Identität gewählt wurde. Weiter oben sind die Unterpunkte des Menüpunkts `Tools` angezeigt. Diese Unterpunkte erlauben die Bearbeitung von Identitäten, Benutzern und Benutzerdaten.

Abbildung 8.7(b) zeigt das Fenster an, das erscheint, wenn der Unterpunkt `User Data Editor` aufgerufen wird. Es zeigt die Liste aller Identitätsattribute des Benutzers an. Die Attribute können zuvor von dem Heimrechner kopiert worden sein. Alternativ kann der Benutzer seine Attribute aber auch mit der MobileGUI auf dem Handheld anlegen, bearbeiten oder löschen. Das Löschen eines Attributs kann durch das Anklicken des Kreuzsymbols auf der rechten Seite des Attributs erzielt werden. Die übrigen Funktionalitäten zur Verwaltung der Attribute werden unter dem Menüpunkt `Attribute` zur Verfügung gestellt. Dabei bietet



(a) Auswählen von Diensten, die das eHome anbietet.

(b) Auswählen eines Dienstes zur Ausführung auf dem Handheld.

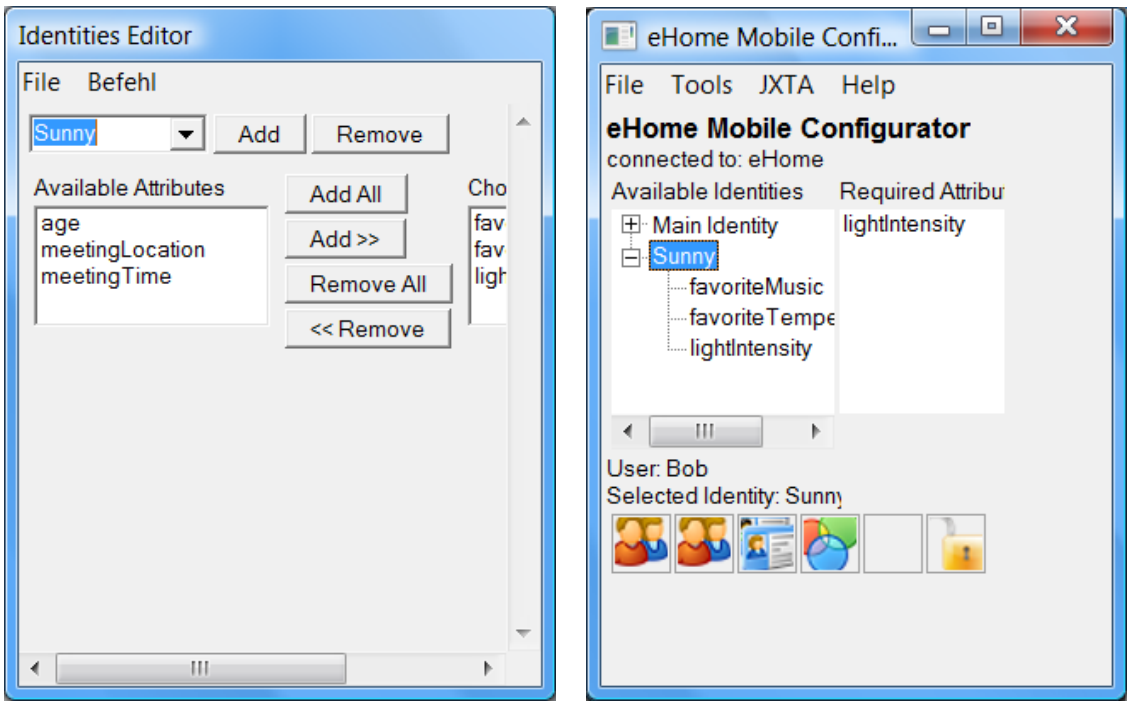
Abbildung 8.8: MobileGUI: Auswählen von Diensten.

die MobileGUI alle Funktionalitäten, die auch der eHomeAdministrator diesbezüglich anbietet. Hierzu gehört auch die Möglichkeit, über das gleiche Fenster wie in Abbildung 8.4 bestimmen zu können, welche Dienste auf ein Attribut zugreifen können (Zugriffskontrollliste).

Falls die Suche nach einem eHome erfolgreich verlaufen ist, kann sich der Benutzer die Liste der im eHome verfügbaren Dienste anzeigen lassen. Abbildung 8.8(a) zeigt solch eine Situation. Im oberen Teil des Fensters ist angegeben, dass das Handheld mit einem eHome namens „`eHome`“ verbunden ist (`connected to:`). Gleich darunter sind die Dienste aufgelistet, die der Benutzer auswählen kann (`Available services:`). Im aktuellen Beispiel hat der Benutzer die Dienste `Light Follows Person` und `Coffee Brewing Service` ausgewählt. Er kann die gesamte Liste einsehen, indem er die Scrollingfunktionalität nutzt.

Als Nächstes kann er, wie in Abbildung 8.8(b) gezeigt, sich die Liste seiner persönlichen Dienste anzeigen lassen, die er mobil ausführen kann (`Installable mobile Services:`). In diesem Beispiel ist dies nur der `Wakeup Service`, von dem angenommen wird, dass der Benutzer diesen auf seinem Handheld ausführen möchte.

Abbildung 8.9(a) zeigt das Fenster an, in dem der Benutzer seine Identitäten bearbeiten kann. Im angezeigten Beispiel wurde die Identität `Sunny` ausgewählt. Im unteren Bereich ist das Fenster zweigeteilt. Auf der linken Seite ist die Liste `Available Attributes` dargestellt, die alle Attribute der Hauptidentität enthält, die der ausgewählten Identität noch nicht zugewiesen worden sind. Rechts neben der Liste befinden sich mehrere Buttons, mit denen Attribute der Identität hinzugefügt oder von dieser entfernt werden können. Weiter rechts kann die Liste der bereits in der gewählten Identität enthaltenen Attribute eingesehen



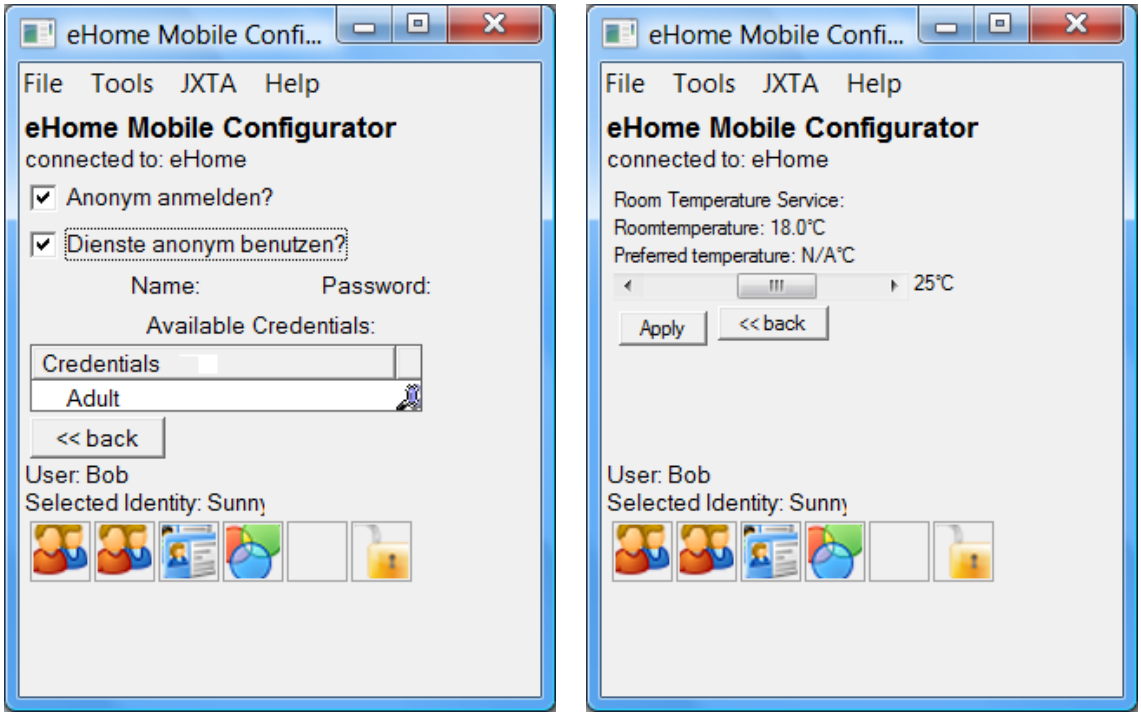
(a) Bearbeiten einer Identität. (b) Auswählen einer Identität.

Abbildung 8.9: MobileGUI: Bearbeiten und Auswählen einer Identität.

werden (`Chosen Attributes`). Hierfür müsste erneut der Scrollbalken verwendet werden. Hinter dieser Funktionalität steckt der in Abschnitt 6.1.1.1 und Abschnitt 6.5.3 beschriebene IdentityManager. Dieser ist dafür zuständig, dass die Identitätsinformationen im mobilen Benutzermodell abgelegt werden.

Die Bearbeitung seiner Identitäten kann der Benutzer auf seinem Handheld jederzeit vornehmen. Bevor er sich an einem eHome jedoch anmeldet, muss er eine konkrete Identität auswählen, mit der er dem eHome entgegentreten möchte. Hierfür dient das in Abbildung 8.9(b) gezeigte Fenster. Dieses Fenster ist auch zweigeteilt. Auf der rechten Seite ist die Liste der vom eHome benötigten Attribute angezeigt (`Required Attributes`). Dabei handelt es sich genau um die Attribute, die von den Diensten benötigt werden, die in Abbildung 8.8(a) ausgewählt wurden. Im gegebenen Beispiel wurden nur zwei Dienste ausgewählt, von denen `Light Follows Person` personalisierbar ist und daher das Attribut `lightIntensity` benötigt. Auf der linken Seite ist die Liste der Identitäten des Benutzers angezeigt, die die geforderten Attribute (in diesem Fall nur `lightIntensity`) enthalten. Der Benutzer kann sich zu jeder Identität anzeigen lassen, welche Attribute insgesamt enthalten sind. Im Beispiel sind die Attribute der Identität `Sunny` aufgelistet. Diese Identität enthält mehr Attribute als nötig sind. Nun kann sich der Benutzer entscheiden, diese Identität trotzdem freizugeben oder eine neue generieren zu lassen, die nur dieses eine geforderte Attribut `lightIntensity` enthält. Im gezeigten Beispiel wurde Identität `Sunny` ausgewählt, wie weiter unten im Fenster angezeigt ist (`Selected Identity`).

Der Benutzer kann an dieser Stelle auch zurückgehen und einige Dienste abwählen, falls sie zu viele oder zu sensible Daten erfordern. Die MobileGUI ist diesbezüglich flexibel gehalten.



(a) Auswählen der Authentifizierungsart.

(b) Interaktion mit einem Dienst über das Handheld.

Abbildung 8.10: MobileGUI: Auswählen der Authentifizierungsart und Interaktion mit einem Dienst.

Die Buttons ganz unten im Fenster erlauben dem Benutzer zwischen den unterschiedlichen Schritten hin und her zu wechseln. Um außerdem den wiederholten Durchlauf dieses Prozesses beim nächsten Betreten des gleichen eHomes zu vermeiden, werden die Ergebnisse dieser Aushandlung auf dem Handheld gespeichert und können beim nächsten Mal automatisch ausgewählt werden.

Nachdem sich der Benutzer nun für die Dienste und die Identität entschieden hat, kann er sich beim eHome anmelden. Das in Abbildung 8.10(a) dargestellte Fenster erlaubt dem Benutzer, sich auf unterschiedliche Arten authentifizieren zu können (s. Abschnitt 7.1). Konkret kann er zwischen drei Alternativen auswählen. Erstens kann er sich mit seinem Benutzernamen und dem zugehörigen Passwort anmelden, falls er dem eHome unter dem Benutzernamen bekannt ist. Dafür sind die Felder `Name` und `Password` vorgesehen. Alternativ kann er sich anonym anhand eines TTP-Credentials anmelden. Hierfür muss er das Häkchen vor der Auswahl `Anonym anmelden?` setzen. Falls er dies tut, wird ihm weiter unten im Fenster angezeigt, welche seiner TTP-Credentials die vom eHome geforderten Eigenschaften enthalten (`Available Credentials:`). Im gezeigten Beispiel besitzt der Benutzer ein Credential, mit dem mit dem er seine Volljährigkeit (`Adult`) nachweisen kann. Hier wurde angenommen, dass dieser Nachweis ausreicht, um dem Benutzer den Zugriff auf die zuvor ausgewählten Dienste zu gewähren. Die MobileGUI erfüllt somit auch die Aufgaben eines *Credential-Managers*, sodass der Benutzer beim Umgang mit seinen Credentials unterstützt wird. Falls der Benutzer zudem ein Session-Credential erhalten

(a) Startbild auf Dell-PDA. (b) Dienstinteraktion auf Dell-PDA.

Abbildung 8.11: Die MobileGUIauf einen Dell-Axim-PDA.

möchte, um auch Sitzungsanonymität zu genießen, muss er zusätzlich das Häkchen vor der Auswahl `Dienste anonym benutzen?` setzen, wie es in der gezeigten Situation geschehen ist. Damit kann er anonym auf die ausgewählten Dienste über die MobileGUI zugreifen.

Falls der Benutzer nach seiner Authentifizierung mit einem der Dienste interagieren möchte, kann er sich zuerst die Liste seiner Dienste anzeigen lassen. Die Liste enthält alle zuvor ausgehandelten Top-Level-Dienste. Nachdem er einen dieser Dienste ausgewählt hat, wird ihm die Benutzeroberfläche des ausgewählten Dienstes angezeigt (s. Abschnitt 5.3.7). In Abbildung 8.10(b) ist beispielsweise die Benutzeroberfläche des `Temperaturdienstes` auf dem Handheld angezeigt. Auf diesem kann der Benutzer die aktuelle Temperatur im Raum einsehen (`18.0°C`) und diesen über einen Schieberegler ändern. Um den gewünschten Wert dem Dienst im eHome mitzuteilen, muss der Button `apply` betätigt werden. Falls der Benutzer wie oben erwähnt ein Session-Credential erhalten hat, berechnen das Handheld und das eHome gemeinsam den benötigten Berechtigungsnachweis. Diese Berechnung findet automatisch und ohne expliziten Eingriff des Benutzers statt.

Zur Evaluierung des Prototyps für Handhelds wurden PDAs der Marke Dell Axim X51v mit Intel PXA270 Prozessoren (624 MHz) und 3.7"Touchscreens mit VGA-Auflösung (640x480 Pixel) verwendet. Sie verfügen über WLAN und Bluetooth zur drahtlosen Kommunikation und Windows Mobile 5.0 Professional als Betriebssystem. Wie bereits in Abschnitt 5.3.2.2 erwähnt, wird eRCP als OSGi-Dienstplattform eingesetzt. Abbildung 8.11

zeigt zwei Screenshots des Prototyps auf den Dell-PDAs. Auf der linken Seite wird das Startbild gezeigt, das in Abbildung 8.7(a) bereits erläutert wurde. Die dort dargestellte Information ist unverändert, nur die Bedienelemente sind optisch anders geordnet. Auf der rechten Seite wird die Benutzeroberfläche des `Lichtdienstes` dargestellt, der, ähnlich zum `Temperaturdienst`, einen Schieberegler zur Einstellung der Beleuchtung anbietet.

8.3 Demonstratoren

Um die im Rahmen dieser Arbeit entwickelten Konzepte testen und auswerten zu können, wurden verschiedene Szenarien realisiert. Dabei wurden alle der in Abschnitt 2.3 beschriebenen Dienste in Form von Testszenarien implementiert, die exemplarisch typische eHome-Funktionalitäten realisieren.

Im Rahmen des Projekts standen für die Evaluation der Dienste und Werkzeuge keine realen Gebäude zur Verfügung. Daher wurden einige Testumgebungen zur Simulation von eHomes herangezogen. Dadurch, dass der eHome-Prototyp anhand der Basisdienste von den konkreten Techniken in den einzelnen eHomes abstrahiert, ergibt sich im Vergleich zu realen Geräten in realen Umgebungen kein qualitativer Unterschied. Im Verlauf dieser Arbeit wurden mehrere solcher Testumgebungen verwendet, die auch als Demonstratoren bezeichnet werden. Für jeden der eingesetzten Demonstratoren mussten für die darin verwendeten Testgeräte eigene Treiberdienste entwickelt werden, die jeweils von der speziellen Kommunikationstechnik abstrahieren. Darüber hinaus konnten die Ergebnisse ohne Änderung wiederverwendet werden, insbesondere auch die integrierenden bzw. Top-Level-Dienste. Im Folgenden werden diese Demonstratoren vorgestellt.

8.3.1 Der X10-Demonstrator

Der erste Demonstrator, der in der eHome-Gruppe eingesetzt wurde, ist der von Norbisrath [Nor07] entwickelte *X10-Demonstrator*, der in Abbildung 8.12 dargestellt ist. Er wurde aus Holz gebaut und besteht aus mehreren Räumen. Das eHome-Gateway wurde hinter dem Demonstrator montiert. Dabei handelt es sich um einen handelsüblichen PC, der sowohl für die Ausführung der Dienste als auch für die Anbindung der Geräte des Demonstrators verwendet wurde. Es wurden verschiedene Geräte verwendet, die über die X10-Technik (s. [Sma10]) angesteuert werden können. Unter anderem wurden X10-Schalter, X10-Lampenfassungen und X10-Bewegungsmelder verwendet, die über ein X10-Gateway mit dem Gateway verbunden wurden. Zusätzlich wurden Touchscreens, Webcams, Aktivboxen und Kopfhörer angeschlossen, die direkt von dem Gateway aus angesteuert werden.

8.3.2 Der Lego-Demonstrator

Weil es sich jedoch herausstellte, dass der X10-Demonstrator schlecht transportierbar ist, hat Norbisrath den kleineren *Lego-Demonstrator* erstellt, der in Abbildung 8.13 dargestellt ist. Dieser wurde aus Legosteinen zusammengesetzt und mit LED-Lampen, Tastern und ebenfalls Webcams ausgestattet. Für die Audioausgabe wurden zusätzlich Kopfhörer verwendet.

Abbildung 8.12: Der X10-eHome-Demonstrator (Quelle: [Nor07]).

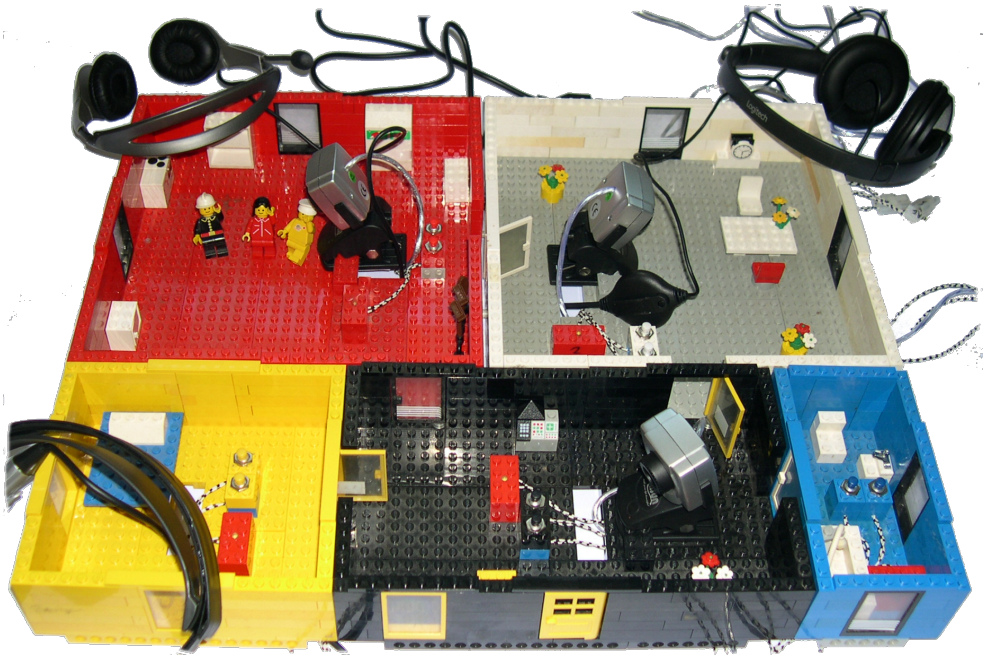


Abbildung 8.13: Der Lego-eHome-Demonstrator (Quelle: [Nor07]).

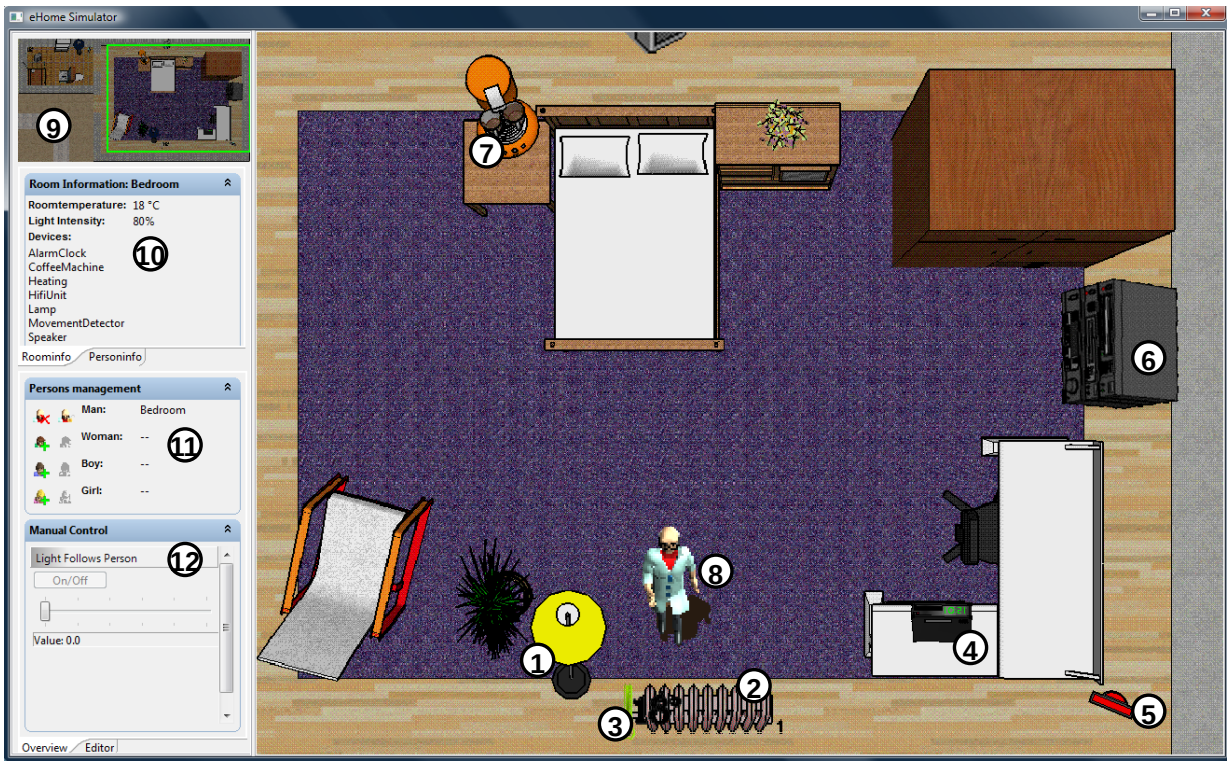


Abbildung 8.14: Der eHomeSimulator.

Aufgrund seiner Größe war es nicht möglich, diesen Demonstrator mit einem festen Gateway auszustatten. Stattdessen wurde hierfür ein Laptop eingesetzt. Die verwendeten Legogeräte wurden über ein spezielles Steuermodul angesteuert, das über die USB-Schnittstelle mit dem Laptop verbunden wurde. Ebenso wurden die Webcams und Kopfhörer über USB an den Laptop angeschlossen. Dieser Demonstrator konnte leicht transportiert und somit beispielsweise auf Konferenzen vorgestellt werden.

8.3.3 Der eHomeSimulator

Im Rahmen der durchgeführten Tests stellte sich heraus, dass weder der X10- noch der Lego-Demonstrator die nötige Flexibilität geboten haben, die für eine angemessene Evaluierung der Konzepte zur Unterstützung der Inter-eHome-Mobilität nötig war. Mit dem Ziel, mehrere eHomes gleichzeitig und damit die Mobilität von Benutzern zwischen diesen simulieren zu können, wurde in Zusammenarbeit mit Retkowitz der eHomeSimulator entwickelt [AR07]. Diese ist vollständig im Software realisiert und erlaubt die Simulation von Geräten wie etwa Heizkörper, Rollläden, Türen und Kaffeemaschinen sowie einer vollautomatischen Personenerkennung. Dadurch können weitergehende Szenarien simuliert werden, die mit den Hardwaredemonstratoren nicht umsetzbar waren. Außerdem bietet der eHomeSimulator die Möglichkeit, mehrere Umgebungen modellieren zu können. Somit ist es möglich, mehrere eHomes zur gleichen Zeit zu instanziieren und die Inter-eHome-Mobilität zu simulieren.

Die 2D-Oberfläche des Simulators besteht aus zwei Teilen, wie in Abbildung 8.14 zu erkennen ist. Im Hauptteil auf der rechten Seite ist die simulierte Umgebung dargestellt. Es handelt sich hier um den Schlafraum eines Hotelzimmers. Der Raum ist mit unterschiedlichen Geräten und dekorativen Gegenständen ausgestattet. Dazu gehören beispielsweise eine Lampe (1), ein Heizkörper (2) mit einem Thermometer (3), ein Wecker (4), ein Bewegungsmelder (5), eine Hi-Fi-Anlage (6) und eine Kaffeemaschine (7). Ferner befindet sich eine Person (8) im Raum, die mithilfe der Maus oder Tastatur bewegt werden kann. Die Geräte können bedient werden, indem sie mit der Maus angeklickt werden. Daraufhin bewegt sich die Person zu dem Gerät, an dem anschließend eine gerätespezifische Bedienoberfläche erscheint. Diese Oberfläche ist ähnlich zu der, die auch auf dem Handheld gezeigt wird. Im Falle einer Lampe würde beispielsweise ein Schieberegler angezeigt, mit der die Helligkeit eingestellt werden kann.

Auf der linken Seite der Oberfläche befindet sich das sogenannte Bedienpanel. Im oberen Teil des Panels befindet sich eine Gesamtübersicht der gesamten Umgebung (9). Unter dieser Übersicht befindet sich eine Anzeige, die Informationen zum aktuellen Zustand der Umgebung bereithält (10). Beispielsweise werden hier die aktuelle Temperatur (18°C) und die Beleuchtungsintensität (80%) im Raum eingeblendet. Ferner ist eine Liste der im Raum verfügbaren Geräte angezeigt. Unter dieser Ansicht befindet sich das Werkzeug zur Verwaltung von Personen (11). Mit diesem Werkzeug können Personen zur Laufzeit dynamisch in die Umgebung hinzugefügt oder wieder entfernt werden. Hier ist es auch möglich, die Personen an das eHome anzumelden, d. h. die Authentifizierung anzustoßen. Dadurch lässt sich die Inter-eHome-Mobilität simulieren. In der Abbildung ist zwar nur eine Person angezeigt, es könnten aber bis zu drei weitere hinzukommen. Im unteren Teil des Kontrolpanels befinden sich die Schnittstellen zu den persönlichen Diensten der Personen (12). Da diese keine Benutzeroberflächen innerhalb der Umgebung besitzen, sind sie separat untergebracht.

8.3.4 Der OpenSim/Second-Life-Demonstrator

Der eHomeSimulator bietet eine Reihe von Vorteilen im Vergleich zu den Hardwaredemonstratoren. Er weist jedoch auch einige Schwächen auf. Erstens wirkt die 2D-Oberfläche nicht besonders natürlich. Zweitens musste der eHomeSimulator auf dem gleichen Rechner ausgeführt werden wie der eHome-Prototyp, eine Ausführung auf unterschiedlichen Rechnern ist nicht möglich. Drittens konnte mit dem eHomeSimulator nur eine Person zur gleichen Zeit aktiv sein.

Um diese Schwächen auszugleichen, wurde im Rahmen dieser Arbeit ein weiterer Demonstrator auf Basis von *Second Life* [RAW+08] bzw. *OpenSim* [ope10] entwickelt. Second Life ist eine Online-3D-Infrastruktur, die ihren Benutzern ermöglicht, virtuelle Welten zu errichten und in diesen Welten durch Avatare mit anderen Teilnehmern zu interagieren, zu spielen oder anderweitig zu kommunizieren. Das Bebauen von eigenem Land in Second Life ist nur gegen Entgelt möglich, während OpenSim eine kostenfreie Alternative darstellt, die größtenteils die gleichen Funktionalitäten zur Verfügung stellt wie Second Life. Die Benutzer können sich mithilfe des *Second Life Viewer* über das Internet an dem Server

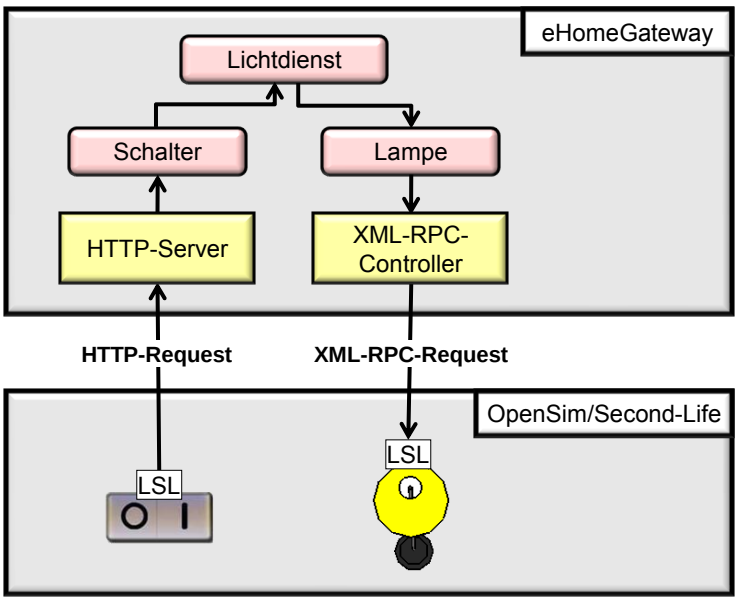


Abbildung 8.15: Anbindung des OpenSim/Second-Life-Demonstrators an ein eHome-Gateway über XML-RPC und HTTP.

anmelden und die gewünschten Funktionalitäten in Anspruch nehmen. Darin enthalten ist auch ein Werkzeug, mit dem 3D-Modelle von beliebigen Objekten erstellt werden können. Dieser Viewer kann sowohl für Second Life als auch für OpenSim verwendet werden.

Jedes Objekt kann zudem mit ein oder mehreren Skripten angereichert werden, die sein Verhalten bestimmen. Solche Skripte können mit der Sprache *Linden Scripting Language* (LSL) erstellt werden, die Ähnlichkeiten zur Programmiersprache C besitzt. Zusätzlich zur Spezifizierung des Verhaltens der Objekte in der virtuellen Welt unterstützt LSL auch Kommunikation dieser mit Entitäten außerhalb der virtuellen Welt.

Abbildung 8.15 stellt am Beispiel des `Lichtdienstes` dar, wie der OpenSim/Second-Life-Demonstrator an ein eHome-Gateway angebunden wurde. Im unteren Bereich der Abbildung ist der Demonstrator mit einem Schalter und einer Lampe dargestellt. Jedes dieser Geräte besitzt ferner ein LSL-Skript. Im oberen Bereich ist der eHome-Prototyp auf einem eHome-Gateway dargestellt. Dort ist der `Lichtdienst` mit zwei Basisdiensten (Treiberdienste) verbunden, nämlich mit `Schalter` und `Lampe`. Der Demonstrator und das eHome-Gateway müssen nicht auf dem gleichen Rechner ausgeführt werden, wie es beim eHomeSimulator der Fall ist, sondern können sich auf unterschiedlichen Rechnern befinden.

Nun wird erläutert, wie Ereignisse aus dem Demonstrator dem eHome-Gateway mitgeteilt werden und wie umgekehrt das eHome-Gateway Nachrichten an den Demonstrator schicken kann. LSL-Skripte können HTTP-Anfragen an einen externen Server senden. Diese Funktionalität wird genutzt, um Ereignisse wie „`Schalter wurde gedrückt`“ an das eHome-Gateway zu senden. Dafür wurde der eHome-Prototyp um einen HTTP-Server erweitert, der dafür zuständig ist, Sensorinformationen aus dem OpenSim/Second-Life-Demonstrator entgegenzunehmen und dem zugehörigen Treiberdienst mitzuteilen. Der Treiberdienst `Schalter` selbst leitet diese Information wiederum an den `Lampendienst` weiter. Falls

Abbildung 8.16: Der OpenSim/Second-Life-Demonstrator.

Letzterer aus dieser Information ableitet, dass die Beleuchtung angepasst werden muss, teilt er dies dem Treiberdienst `Lampe` mit. Um den Objekten im Demonstrator wie der Lampe Nachrichten senden zu können, wird die Möglichkeit genutzt, dass OpenSim und Second Life XML-RPC-Nachrichten empfangen können (s. Abschnitt 5.3.3). Hierfür wurde der XML-RPC-Controller entwickelt, der die Anfragen der Treiberdienste entgegennimmt und über einen entsprechenden XML-RPC-Aufruf an das entsprechende Objekt im Demonstrator weiterleitet.

Damit ist der Kreis geschlossen. Wenn nun ein Avatar im Demonstrator einen Lichtschalter bedient, kann der `Lichtdienst` auf dem eHome-Gateway – nach Überprüfung der Zugriffsrechte des zugehörigen Benutzers – die Beleuchtung der entsprechenden Lampe anpassen. Für weiterführende Information zur Realisierung des Demonstrators und seiner Anbindung an das eHome-Gateway sei an dieser Stelle auf [Mei10] verwiesen.

Ein Screenshot des OpenSim/Second-Life-Demonstrators ist in Abbildung 8.16 dargestellt. Dort ist ein Raum zu erkennen, der wieder dem Schlafraum eines Hotelzimmers nachgeahmt ist. Zusätzlich zum üblichen Mobiliar wie einem Bett, einem Esstisch oder einem Schrank ist der Raum mit Geräten ausgestattet, die von den eHome-Diensten verwendet werden können. Beispielsweise sind eine Lampe (1), eine Kaffeemaschine (2), ein

Lichtschalter an der Wand (3), mehrere Lautsprecher (4) und ein Fernseher (5) dargestellt. Rollläden, Heizkörper, Türen und Deckenlampen sind einige der weiteren Geräte, die für diesen Demonstrator modelliert wurden, aber in dieser Abbildung nicht sichtbar sind. Ferner ist im Raum eine Person in Form eines Avatars (6) dargestellt, der vom Anwender frei bewegt werden kann. Dieser Avatar hat einen eindeutigen Namen in der virtuellen Welt, der auch für die Realisierung der Personenerkennung verwendet wurde.

Ein besonderer Vorteil dieses Demonstrators liegt darin, dass nun gleichzeitig mehrere Benutzer in der virtuellen Welt aktiv sein können. Jeder der Benutzer kann sich von einem anderen Rechner aus mit der Umgebung verbinden und die Welt aus eigener Perspektive betrachten. Die im Rahmen dieser Arbeit modellierte Welt enthält eine Straße mit zwei weiteren Häusern. Jede dieser Häuser kann als ein eigenes eHome aufgefasst werden, ohne dass zusätzliche Instanzen des Demonstrators gestartet werden müssen. Jedes simulierte eHome kann ferner an ein eigenes eHome-Gateway angebunden werden. Dadurch konnte die Simulation der Inter-eHome-Mobilität vereinfacht und realitätsnah durchgeführt werden. Der Benutzer kann sich nun einfach innerhalb einer Instanz des Demonstrators von einem eHome in ein anderes begeben. Der eHomeSimulator hingegen musste mehrfach instanziiert werden, um das gleiche Verhalten simulieren zu können.

8.3.5 Implementierte Anwendungsbeispiele

Auf Basis der beiden Softwaredemonstratoren (eHomeSimulator und OpenSim/Second-Life-Demonstrator) wurden einige der Testszenarien aus Abschnitt 2.3 umgesetzt. Dabei konnte die Anwendbarkeit der im Rahmen dieser Arbeit entwickelten Konzepte gezeigt werden.

Als einfache Dienste wurden ein `Lichtdienst` und ein `Temperaturdienst` entwickelt. In einem weiteren Schritt wurden diese Dienste personalisiert, sodass die Raumbeleuchtung bzw. die Raumtemperatur automatisch den Präferenzen der Benutzer angepasst werden konnten. Auch der `Musikdienst`, der dem Benutzer die Musik durch das eHome folgen lässt, konnte erfolgreich realisiert werden.

Ferner wurde ein `Medizindienst` entwickelt, der abhängig von den Vitalwerten Puls, Blutdruck und Körpertemperatur Notfälle erkennen und Maßnahmen ergreifen kann. Unter anderem kann dieser Dienst eine Nachricht an eine vorgegebene Adresse senden und für das Rettungspersonal automatisch die Türen des eHomes öffnen. Außerdem kann er den Weg markieren, der das Rettungspersonal am schnellsten von der Außentür bis zur erkrankten Person führt. Hierfür wurden exemplarisch die Türen farblich gefärbt, durch die der schnellste Weg führt. Diese Funktionalität ist insbesondere in größeren Gebäuden nützlich.

Außerdem wurde der `Weckdienst` mit all seinen in Abschnitt 2.3.4 beschriebenen Funktionalitäten erstellt. Alle hierfür benötigten Geräte wie Lampen, Lautsprecher, Rollläden, Kaffeemaschine usw. waren im OpenSim/Second-Life-Demonstrator verfügbar. Für die Berechnung der Fahrtzeit wurde ein Unterdienst entwickelt, der zufällig Wetter- und Stauinformationen generiert und dadurch die Fahrtzeit beeinflusst hat. Dadurch konnte überprüft werden, ob der Dienst die Weckzeit auch dynamisch berechnen kann.

Schließlich wurden der `Weckdienst` und der personalisierbare `Lichtdienst` exemplarisch auf den eingesetzten PDAs ausgeführt. Es konnte hierdurch gezeigt werden, dass die mobile

Ausführung und die verteilte Kommunikation von eHome-Diensten über Gateway-Grenzen hinweg durchaus funktioniert.

8.4 Zusammenfassung

In diesem Kapitel wurden die im Rahmen der vorliegenden Arbeit entwickelten Werkzeuge vorgestellt. Zunächst wurde der eHomeAdministrator als Werkzeug für das eHome beschrieben, der den Anwender in einem eHome dabei unterstützt, die Rechte für Zugriffe auf eHome-Dienste zu verwalten. Zudem kann dieses Werkzeug für die Verwaltung von Benutzerdaten und Identitäten verwendet werden.

Als Nächstes wurden die Funktionalitäten des Werkzeugs für Handhelds beschrieben, die über die Benutzeroberfläche MobileGUI bedienbar sind. Mobile Benutzer können dieses Werkzeug auch dazu verwenden, um ihre Daten und Identitäten handzuhaben. Außerdem unterstützt dieses Werkzeug die Benutzer bei der Aushandlung von Diensten und Identitäten sowie bei der Wahl eines passenden Authentifizierungsmechanismus, und ermöglicht somit eine komfortable Anmeldung an einem eHome.

Des Weiteren wurden die Demonstratoren beschrieben, die zur Evaluierung der im Rahmen dieser Arbeit entwickelten Konzepte, Werkzeuge und Dienste herangezogen wurden. Für diese Arbeit wurde dabei in erster Linie auf die beiden neu entwickelten Softwaredemonstratoren zurückgegriffen, die jeweils eine virtuelle eHome-Umgebung zur Verfügung stellen. Damit konnten die in Abschnitt 2.3 beschriebenen Dienste in verschiedenen Szenarien getestet werden.

Kapitel 9

Verwandte Ansätze

Im letzten Jahrzehnt haben sich viele Forschungsprojekte mit Fragestellungen rund um intelligente Umgebungen wie eHomes beschäftigt. Im Folgenden werden einige Arbeiten vorgestellt, die sich mit ähnlichen Themen wie in der vorliegenden Arbeit beschäftigt haben. Zunächst werden Ansätze beschrieben, die sich mit der Mobilität von Benutzern befasst haben. Anschließend werden Ansätze zur Benutzermodellierung und Personalisierung wiedergegeben. Den Abschluss dieses Kapitels bildet eine Übersicht von Ansätzen zur Privatsphäre und Sicherheit.

9.1 Ansätze zur Mobilität

In diesen Abschnitt werden Ansätze beschrieben, die sich unter anderem mit dem Aspekt der Mobilität in und zwischen intelligenten Umgebungen beschäftigt haben.

9.1.1 Gaia

Im Rahmen des Gaia-Projekts [Uni10] der University of Illinois wurde ein Middleware-Betriebssystem entwickelt, das für die Verwaltung der Ressourcen eines sogenannten *Active Space* verantwortlich ist. Die Idee eines Active Space ähnelt dem eines eHomes: Ein Active Space ist ein physikalischer Ort, der sowohl physikalische als auch virtuelle Komponenten, also Hardware und Software, enthält. Diese Komponenten werden allgemein als Ressourcen betrachtet, für deren Verwaltung das Betriebssystem *Gaia OS* zuständig ist. Es übernimmt diese Aufgabe und bietet eine Abstraktionsschicht für spezifische Eigenschaften von Active Spaces, sodass für die Entwicklung von Anwendungen eine einheitliche Softwareinfrastruktur zur Verfügung steht. Gaia OS ist ein verteiltes, objektorientiertes System, das auf CORBA [GT00] aufbaut.

Die Architektur des Gaia-Systems enthält drei große Bestandteile: den Gaia Kernel, das Gaia Application Framework und die Anwendungen. Der *Gaia Kernel* besteht aus dem Component Management Core, der die Verwaltung und Installation von Diensten übernimmt, sowie verschiedenen Basisdiensten. Diese bieten generelle Funktionen, wie z. B. das Ereignismanagement oder das Bereitstellen von Kontextinformationen, die von den

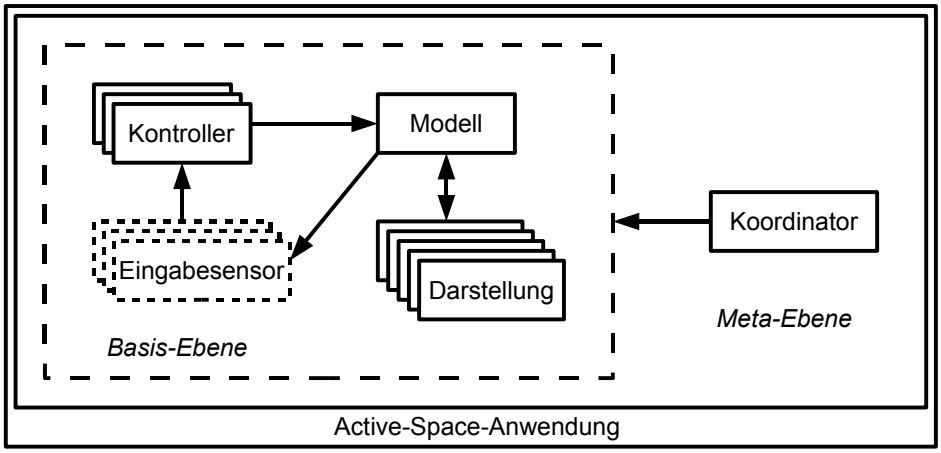


Abbildung 9.1: Struktur einer Anwendung in Gaia (nach [RHC02]).

Anwendungen und dem Framework benötigt werden. Das *Gaia Application Framework* unterstützt die Entwickler von Anwendungen durch Mechanismen zum Erzeugen, Ausführen und Anpassen dieser an bestehende Active Spaces.

Eine Anwendung wird in Anlehnung an das Architekturmuster Model-View-Controller (MVC) in folgende vier Komponenten aufgeteilt (s. Abbildung 9.1):

- Das *Modell* implementiert die Anwendungslogik und stellt eine Schnittstelle bereit, um den Zustand der Anwendung abzufragen und zu verändern. Es ist die einzige Komponente, die über einen Zustand verfügt. Andere Objekte können sich beim Modell registrieren, um über Änderungen des Zustands benachrichtigt zu werden.

- Eine *Darstellung* bildet den Zustand der Anwendung in eine vom Benutzer wahrnehmbare Form ab. Diese kann beliebig sein, z. B. visuell, hörbar oder eine Änderung der Temperatur. Zu einer Anwendung bzw. dem enthaltenen Modell kann es mehrere Darstellungen geben.

- Ein *Eingabesensor* kann jedes Artefakt sein, das eine Zustandsänderung des Modells bewirken kann. Beispiele dafür sind eine grafische Oberfläche, Eingabegeräte oder Sensoren, d. h., es kommen sowohl Hardware- als auch Softwareeinheiten in Frage. Wie bei den Darstellungen kann es auch mehrere Eingabesensoren pro Anwendung geben.

- Ein *Kontroller* bildet die Ereignisse der Eingabesensoren auf das Modell ab, indem er Methoden des Modells aufruft. So können Eingabesensoren von konkreten Anwendungen entkoppelt werden, weil der Kontroller als Adapter fungiert und die konkrete Schnittstelle des Modells vor den Eingabesensoren verbirgt. Da eine Anwendung mehrere Eingabesensoren benutzen kann, kann sie dementsprechend auch mehrere Kontroller beinhalten.

Der *Koordinator* ist nicht Teil des Anwendungskerns. Er verwaltet die Komposition der vier Basiskomponenten und arbeitet damit auf der Meta-Ebene der Anwendung.

In Gaia werden verschiedene Aspekte der Mobilität berücksichtigt [CUH+01], die in den folgenden Unterabschnitten erläutert werden.

9.1.1.1 Mobilität von Benutzern

Die Unterstützung mobiler Benutzer erfolgt mithilfe globaler Dienste, die nicht in einem speziellen Active Space ausgeführt werden, aber von allen Active Spaces benutzt werden können. Der *User Proxy* ist ein solcher Dienst, der speziell an einen Benutzer gebunden ist. Er kennt dessen Aufenthaltsort, seine Präferenzen und andere Daten, die er anderen Anwendungen bei Bedarf zur Verfügung stellt. Der *Provider Service* ist ebenfalls global, aber unabhängig von einem speziellen Benutzer. Er besitzt Referenzen auf die User Proxies aller bekannten Benutzer. Betritt der Benutzer einen Active Space, wird eine Login-Komponente benachrichtigt, die den Provider Service kontaktiert und über ihn auf den User Proxy des Benutzers zugreift, um Konfigurationen und Präferenzen abzurufen. Die Verwendung globaler Dienste erleichtert somit den Austausch von Daten, setzt aber ein zentrales System voraus, auf dem diese Dienste ausgeführt und allen Active Spaces zur Verfügung gestellt werden.

9.1.1.2 Mobilität von Geräten

Bei der Unterstützung für mobile Geräte besteht das Anwendungsszenario darin, dass der Benutzer ein mobiles Gerät bei sich trägt, auf dem Anwendungen ausgeführt werden, die dem aktuellen Active Space angepasst werden sollen. Zur Unterstützung dieses Vorgangs befindet sich in jedem Active Space ein sogenannter *Location Notifier*, der den User Proxy eines Benutzers informiert, sobald dieser den Active Space betritt oder verlässt. Damit besitzt der User Proxy immer aktuelle Informationen über den Aufenthaltsort des Benutzers, auf dessen Änderung die Anwendungen auf dem mobilen Gerät reagieren können.

9.1.1.3 Mobilität von Anwendungen

Unter der Mobilität von Anwendungen in Gaia [RHC02] wird die Migration einer Anwendung von einem Active Space in einen anderen verstanden. Dieser Vorgang erfordert eine Rekonfigurierung der Anwendung und somit eine Anpassung an den neuen Kontext.

Dabei werden zwei Arten der Mobilität unterschieden: Die *Intra-Space Mobility* betrachtet eine Anwendung, die innerhalb eines Active Space ein benutztes Gerät gegen ein anderes Gerät austauschen soll. Dieser Vorgang sowie die Auswahl des Geräts erfolgt manuell durch den Benutzer. Von besonderem Interesse ist die *Inter-Space Mobility*, bei der eine Anwendung von einem Active Space in einen anderen migriert. Durch die Strukturierung von Anwendungen in Gaia nach dem MVC-Prinzip (s. Abbildung 9.1) gibt es zwei Möglichkeiten, wie diese Art der Migration stattfinden kann:

- Bei einer *interaktiven Migration* werden nur die interaktiven Komponenten in den neuen Active Space übertragen. Diesem Prinzip liegt die Unterscheidung zwischen

Ausführungs- und Auswirkungsort einer Anwendung zugrunde (s. hierzu auch Abschnitt 4.3.1). Es wird hier also nur der Auswirkungsort an den aktuellen Active Space angepasst. Die Auswahl, welche interaktiven Komponenten migriert werden, trifft der Benutzer und kann im Benutzerprofil gespeichert werden.

- Bei einer *vollständigen Migration* wird die gesamte Anwendung in den neuen Active Space übertragen, d. h., es ändern sich sowohl der Auswirkungs- als auch der Ausführungsort. Implementiert wird diese Migration durch einen „suspend & resume"-Mechanismus, der den Zustand der Anwendung vor der Migration speichert und nach der Migration wiederherstellt.

9.1.1.4 Vergleich

Gaia zählt zu einen der ersten Projekte, die sich der Umsetzung intelligenter Umgebungen gewidmet haben. Es bietet eine breite Palette an Konzepten an, die parallelen im eHome-Projekt haben. Hierzu zählen etwa die Intra- und die Inter-Space Mobility.

Beide Ansätze unterscheiden sich jedoch an der Umsetzung der Konzepte. Zunächst setzt Gaia voraus, dass alle Avtive Spaces miteinander vernetzt sind. Dadurch wird beispielsweise die Migration von Anwendungen zwischen den Active Spaces realisiert. Ferner sieht Gaia zentral verfügbare Dienste, den Provider Service und jeweils einen User Proxy pro Benutzer vor. Der Provider Service kennt damit *alle* Benutzer sowie deren Daten und kann zudem Bewegungsprofile der Benutzer im Rahmen der Inter-eHome-Mobilität erstellen. Solch ein zentralistischer Ansatz wurde im Rahmen dieser Arbeit mit Hinblick auf den Schutz der Privatsphäre explizit verworfen (s. Abschnitt 5.2).

Im weiteren Verlauf dieses Kapitels wird Gaia in Bezug auf die Privatsphäre und Sicherheit erneut aufgegriffen.

9.1.2 P2PComp

In [FHMO04] wird ein System mit dem Namen „P2PComp" beschrieben. Es verfolgt Ziel, eine verteilte Kommunikation zwischen OSGi-Komponenten zu ermöglichen. Dazu wurde eine Abstraktionsschicht geschaffen, die den Ausführungsort der Komponenten verbirgt und den Zugriff auf die angebotenen Dienste vereinheitlicht, unabhängig davon, ob ein Methodenaufruf lokal oder entfernt ausgeführt wird. Die eigentliche Kommunikation wird auf Basis von JXTA realisiert, um die Entwicklung für mobile Peer-to-Peer-Anwendungen zu unterstützen.

Die Umsetzung dieser Idee erfolgt mithilfe sogenannter *Ports* (s. Abbildung 9.2), die den Endpunkt eines Kommunikationskanals darstellen. Es werden drei Arten von Ports unterschieden:

- *Provides-Ports* stellen die Schnittstelle angebotener Dienste dar, die von anderen Komponenten benutzt werden können.

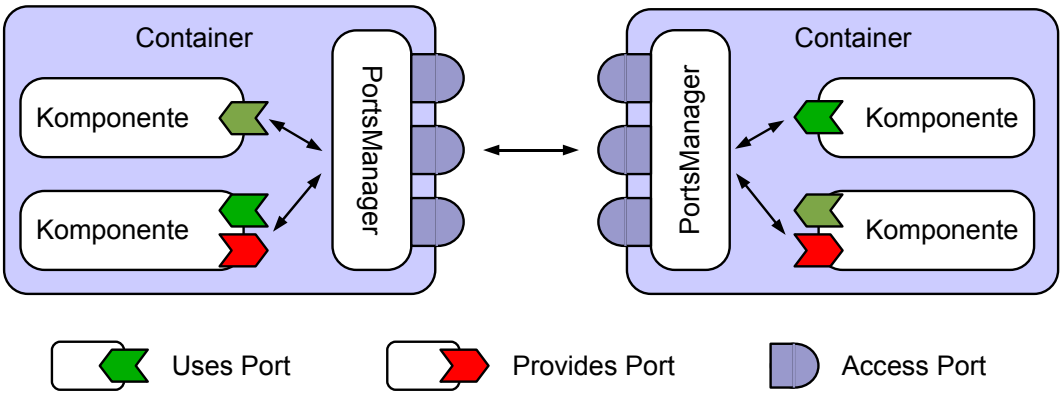


Abbildung 9.2: Ports-Konzept von P2PComp (nach [FHMO04]).

- *Uses-Ports* verkörpern die Verbindungen zu den Provides-Ports anderer Komponenten. Sie dienen also dazu, Dienste zu benutzen, ohne zu wissen, wo sich die Instanz des benutzten Dienstes befindet.

- *Access-Ports* bilden die Kommunikationsschnittstelle zwischen OSGi-Containern. Als Container wird hier die Laufzeitumgebung der Komponenten bezeichnet. Das Protokoll, das hier zur Kommunikation verwendet wird, ist beliebig. Der genaue Ablauf der Kommunikation wird vor den Komponenten nicht verborgen.

Die Realisierung der Provides- und Uses-Ports erfolgt über den *PortsManager*, der als zusätzliches OSGi-Bundle in die Laufzeitumgebung eingefügt wird. Er ersetzt die OSGi-Registry zum Zugriff auf andere Komponenten und kann zusätzlich mit anderen PortsManagern interagieren, sodass lokale und entfernte Komponenten benutzt werden können. Zur Kommunikation benutzt der PortsManager den *P2PService*. Dieser Dienst implementiert die Access-Ports auf Basis von JXTA und realisiert somit die Verbindung mehrerer Container.

9.1.2.1 Vergleich

Mit P2PComp ist es möglich, Dienste zu nutzen, die sich auf einem entfernten Gateway befinden können. Dieser Ansatz ist daher ähnlich zur Ausführung persönlicher Dienste auf einem Handheld, die mit Unterdiensten in eHomes kommunizieren können.

Die Umsetzung der Verteilung in P2PComp unterscheidet sich jedoch in einigen Punkten von der in dieser Arbeit. In P2PComp sind die Dienste an der Komponierung aktiv beteiligt, indem sie über die OSGi-Registry Referenzen für ihre Unterdienste anfragen. Im Kontext der vorliegenden Arbeit wird die Komponierung der Dienste jedoch durch die Laufzeitumgebung, dem eHomeConfigurator, vorgenommen (s. Abschnitt 3.3.1). Ferner werden in P2PComp Aspekte wie die Auswahl und Mitnahme von Diensten, Benutzerinteraktion oder Authentifizierung nicht berücksichtigt. Außerdem werden in P2PComp nur Komponenten betrachtet, die in einem Container aktuell ausgeführt werden. Die Personalisierung durch

eine individuelle Auswahl von Komponenten und Diensten wird nicht berücksichtigt, d. h., es ist nicht vorgesehen, abhängig vom aktuellen Kontext des Systems bestimmte Dienste zusätzlich zur Verfügung zu stellen oder zu entfernen. Schließlich sieht P2PComp wie Gaia die Vernetzung mehrerer Umgebungen vor, die im Rahmen dieser Arbeit nicht verfolgt wurde.

9.1.3 JXTA für Virtual Home Environments

Ein zu P2PComp ähnlicher Ansatz wird in [LSM03] und [LMBE03] beschrieben. Mithilfe von JXTA werden mehrere eHomes zu einer virtuellen Umgebung verbunden, dem sogenannten *Virtual Home Environment* (VHE). Damit sollen Benutzer die Möglichkeit erhalten, ihre Geräte von überall kontrollieren und überwachen zu können. Es findet eine Unterscheidung von inhome- und interhome-Kommunikation statt. Während im ersten Fall Dienste lokal auf einem Gateway komponiert und Geräte über OSGi angesteuert werden, sind im zweiten Fall mehrere Gateways beteiligt. Zur Realisierung der Kollaboration der Gateways wurde ein auf JXTA basierender *VHE P2P Service* entwickelt.

Die Umsetzung dieses Konzeptes erfolgt durch zusätzliche Dienste, die auf den Gateways installiert werden. Der *VHE Virtual Device Service* bietet einen einheitlichen Zugriff auf Geräte. Diese werden anhand ihrer Funktionalität, ihres Standorts und ihrer Benutzer gruppiert, wobei jede Gruppe jeweils durch einen eigenen Dienst in der OSGi-Laufzeitumgebung repräsentiert wird. Dabei überlappen Gruppen in der Regel, wodurch jedes Gerät typischerweise in mehreren Gruppen enthalten ist. Da jede Gruppe durch einen einzigen Dienst dargestellt wird, kann der Benutzer somit alle Geräte einer Gruppe als ein einziges virtuelles Gerät ansprechen.

Der *VHE P2P Communication Service* ist für die Verbindung der Gateways zuständig. Damit soll eine übergangslose Migration der Anwendungen aus dem *inhome*-Bereich in den interhome-Bereich stattfinden. Dienste und Geräte eines eHomes können so von mobilen Geräten aus beliebigen Orten aus abgerufen und gesteuert werden.

9.1.3.1 Vergleich

Der VHE-Ansatz beschränkt sich auf die Verfügbarkeit von Geräten, die von externen Gateways und Benutzern kontrolliert und überwacht werden können. In Bezug auf die Mobilität des Benutzers wird die Fragestellung betrachtetet, wie die Geräte der Heimumgebung für den entfernten Zugriff zur Verfügung gestellt werden können.

Es werden jedoch keine Dienste oder Daten betrachtet, die der Benutzer für die Personalisierung weiterer eHomes mitnehmen möchte, wie es in der vorliegenden Arbeit realisiert wurde. Außerdem sieht der VHE-Ansatz die Vernetzung aller Umgebungen eines Benutzers vor, die, wie bereits weiter oben erläutert, im Rahmen dieser Arbeit mit Hinblick auf Privatsphäre verworfen wurde.

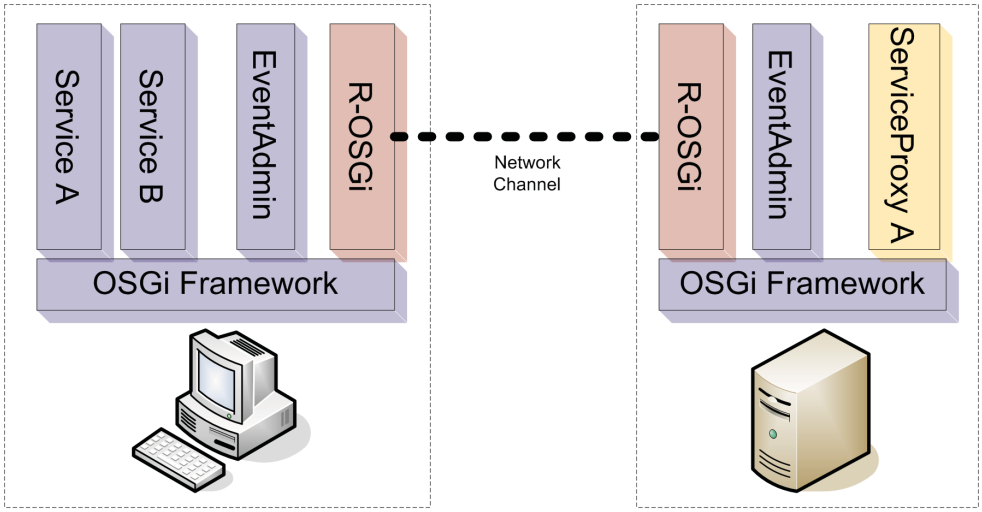


Abbildung 9.3: Übersicht über die R-OSGI-Architektur (aus [RAR07]).

9.1.4 R-OSGi

Ein weiterer Ansatz zur Erweiterung von OSGi um Verteilungsaspekte ist *R-OSGi*, das als Teil des *flowSGi*-Projekts an der ETH Zürich entwickelt wurde [RAR07].

R-OSGi ermöglicht dem Entwickler, eine für OSGi entwickelte Anwendung *unverändert* verteilt ausführen zu können. Zu diesem Zweck kann ein Deployment der einzelnen OSGi-Bundles auf verschiedene Gateways erfolgen. R-OSGi ist seinerseits als OSGi-Bundle realisiert und wird auf jedem Gateway gestartet, welches für die Verteilung vorgesehen ist. Die Kommunikation zwischen den einzelnen Bundles erfolgt anschließend über ein TCP/IP-basiertes Netzwerk.

R-OSGi ermöglicht das verteilte Deployment durch die Umsetzung von vier technischen Konzepten. Erstens unterstützt R-OSGi das dynamische Erzeugen von Stellvertreterobjekten (Proxies), wenn ein Dienst angesprochen wird, der sich auf einem anderen Gateway befindet. Zweitens realisiert R-OSGi eine verteilte Service Registry, die das Auffinden entfernt ausgeführter Dienste ermöglicht. Das zugehörige Service Discovery Protocol setzt dabei auf dem bekannten *Service Location Protocol* (SLP) auf [Gut99]. Drittens werden Netzwerkprobleme, wie etwa die Unerreichbarkeit von Gateways, durch die Abbildung auf das OSGi-Lebenszyklusmodell gehandhabt. Viertens werden durch das sogenannte Type Injection alle Java-Klassen, die von einem Bundle exportiert werden und auf einem entfernten Gateway verfügbar sein sollen, mit dem zugehörigen Stellvertreter-Bundle übertragen.

9.1.4.1 Vergleich

Mit der Erweiterung von OSGi um Verteilungsaspekte verfolgt R-OSGi ein ähnliches Ziel wie die vorliegende Arbeit mit der Ausführung von Top-Level-Diensten auf Handhelds.

Es existieren jedoch auch Unterschiede zwischen beiden Ansätzen. Zunächst sieht R-OSGi vor, dass die Dienste aktiv an der Ermittlung von Referenzen auf ihre Unterdienste beteiligt sind. In der vorliegenden Arbeit werden die Dienste hingegen anhand ihrer Spezifikationen

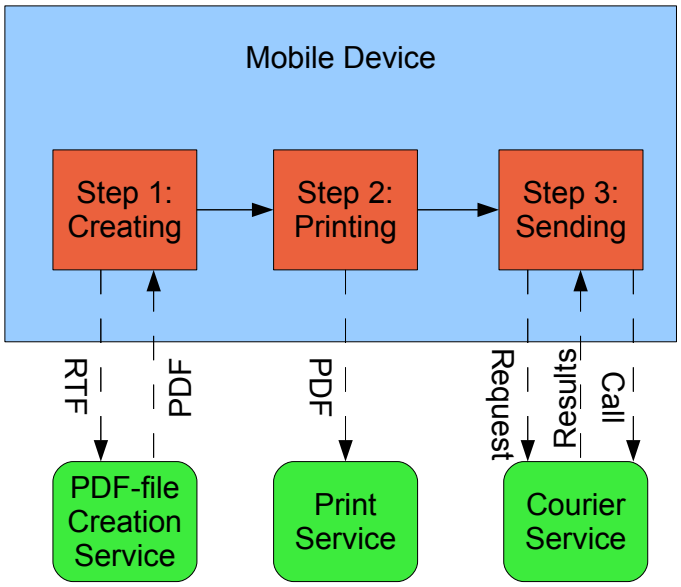


Abbildung 9.4: DIANE-Beispiel: Arbeitsprozess auf mobilem Gerät (angelehnt an [Vay05]).

durch den eHomeConfigurator miteinander komponiert und erhalten die Referenzen auf ihre Dienste per Dependency Injection. Ferner verwendet R-OSGi SLP zur Auffindung von Gateways und Diensten, während in dieser Arbeit JXTA eingesetzt wurde.

9.1.5 DIANE

Das *DIANE*-Projekt der Universität Jena beschäftigt sich mit der Entwicklung und Auswertung von Konzepten für die effiziente und effektive Nutzung von Ressourcen in Form von Diensten in Ad-Hoc-Netzwerken [KKR07, KKR09, KKRSK07, Vay05]. Dabei spielen mobile Geräte und Web Services eine wichtige Rolle.

DIANE befasst sich eingehend mit der Beschreibung von Dienstfunktionalitäten, der Formulierung von Anfragen zu solchen Dienstfunktionalitäten und dem automatisierten Matching zwischen diesen Beschreibungen und Anfragen. Dies soll das Auffinden geeigneter Dienste für den Benutzer vereinfachen, um beispielsweise rechenintensive Prozesse von seinem mobilen Gerät auslagern zu können, indem er einen entfernten Dienst nutzt, der eine passende Funktionalität zur Verfügung stellt.

Zu diesem Zweck wurde die *DIANE Service Description Language* entwickelt, mit der es möglich ist, eine flexible Beschreibung der Funktionalitäten von Diensten zu formulieren. Auch Anfragen zu solchen Funktionalitäten können mithilfe dieser Sprache gestellt werden. Dies wird nun anhand eines kurzen Beispiels veranschaulicht.

Abbildung 9.4 stellt eine Situation dar, in der ein mobiler Benutzer zuvor angefertigten Bericht drucken möchte, der auf seinem mobilen Gerät vorhanden ist. Da der Bericht im RTF-Format vorliegt, soll es vor dem Drucken in das PDF-Format umgewandelt werden. Der Prozess zum Umwandeln erfordert eine hohe Rechenkraft. Daher soll er ausgelagert werden. Die Anfrage nach einem passenden Dienst und das Auffinden eines solchen geschieht

über DIANE. Nachdem eine Bindung zu einem entsprechenden Dienst aufgebaut worden ist, kann die Konvertierung durchgeführt werden. Anschließend kann das Dokument über DIANE an einen geeigneten Druckdienst gesendet werden. Im letzten Schritt dieses Beispiels möchte der Benutzer den gedruckten Bericht nun mit einem Kurierdienst versenden. Auch hierfür kann DIANE eingesetzt werden, um einen passenden Dienst zu finden und ihn mit der Lieferung zu beauftragen.

9.1.5.1 Vergleich

Ähnlich zur vorliegenden Arbeit ermöglicht DIANE die Interaktion von Diensten, die auf einem mobilen Gerät aufgeführt werden mit solchen, die extern, beispielsweise in einem eHome verfügbar sind. Auch eine dynamische Umkonfigurierung aufgrund der Benutzermobilität ist mit DIANE möglich.

Die verfolgten Ziele beider Ansätze unterscheiden sich jedoch. Das Hauptaugenmerk liegt bei DIANE im Bereich *Service Discovery* bzw. (semantisches) *Service Matching*. Dabei sind die Dienste bei DIANE aktiv an ihrer Komponierung beteiligt. Im Rahmen dieser Arbeit wird die Komponierung von Diensten hingegen durch den eHomeConfigurator durchgeführt wird. Dadurch wird die Entwicklung der Dienste vereinfacht.

9.1.6 DynAMITE

Das *DynAMITE*-Projekt (Dynamisch Adaptive Multimodale IT-Ensembles) am Fraunhofer Institute for Computer Graphics Research in Darmstadt befasst sich mit der Entwicklung eines *Softwarerahmenwerks* für die multimodale und intelligente Interaktion mit verteilten und dynamisch veränderlichen Geräteinfrastrukturen, wie sie auch in einem eHome vorzufinden sind [Shi06].

Mit dem entwickelten DynAMITE-Rahmenwerk soll die spontane Kooperation und Interaktion von Geräten und Softwarekomponenten unterschiedlicher Hersteller ermöglicht werden. Dabei kommen auch mobile Geräte zum Einsatz. Um den Benutzer bei seiner aktuellen Tätigkeit bestmöglich zu unterstützen, bietet ihm das Rahmenwerk die hierfür *passenden* Geräte zur Interaktion an. Zu diesem Zweck wurde mit dem *DynAMITE Environment Model* ein formales Kontextmodell entwickelt, das der Situationsbeschreibung dient. Anhand der aktuellen Kontextinformationen wird eine Anfrage an eine passende Dienstreferenz gestellt und dem Benutzer mithilfe des Kontextmodells eine Auswahl möglicher Geräte für seine aktuelle Tätigkeit angeboten, die er anschließend nutzen kann.

Beispielsweise sei angenommen, dass ein Benutzer abends nach Hause kommt und auf seinem mobilen Gerät selbst erstellte Bilder und Videos gespeichert hat, die er nun seiner Frau präsentieren möchte. Da die Anzeige des mobilen Geräts nur eine geringe Auflösung bereitstellt, wird dem Benutzer mithilfe des Kontextmodells zunächst eine Auswahl möglicher Geräte angeboten, welche die Funktionalität der Bildausgabe anbieten. Der Benutzer entscheidet sich schließlich für die Ausgabe der Bilder über einen im Wohnzimmer befindlichen Beamer. Für das Vorführen der Videos bietet ihm das Rahmenwerk an, zusätzlich die Lautsprecher der im Wohnzimmer installierten Hi-Fi-Anlage zu benutzen.

9.1.6.1 Vergleich

DynAMITE weist starke Ähnlichkeiten zum zuvor vorgestellten DIANE-Ansatz auf. Daher sind die Gemeinsamkeiten und Unterschiede von DynAMITE zu dem im Rahmen der vorliegenden Arbeit verfolgten Ansatz ähnlich zu denen von DIANE. Die Art und Weise der Komponierung von Diensten stellt das besondere Unterscheidungsmerkmal dar.

9.1.7 Mobile Geräte als Benutzerschnittstelle

In diesem Abschnitt wird ein knapper Überblick über einige ausgewählte Projekte gegeben, in denen mobile Geräte von Benutzern zur *Interaktion* mit intelligenten Umgebungen verwendet werden können.

Im Rahmen des *PERCI*-Projekts (PERvasive serviCe Interaction) [RBLS07] wird die prototypische Umsetzung von Benutzerschnittstellen in intelligenten Umgebungen analysiert. Es werden dabei verschiedene Phasen durchlaufen: Zunächst werden Benutzerschnittstellen für das eingesetzte mobile Gerät (Nokia 3320) entworfen, die anschließend als Benutzeroberfläche auf dem mobilen Gerät umgesetzt werden. Damit wird die Steuerung verschiedener Geräte in einer intelligenten Umgebung über verschiedene Netzwerkschnittstellen ermöglicht. Dabei kommen GPRS/UMTS, Bluetooth, UPnP und JAVA RMI zum Einsatz.

Auch im *SMADA*-Projekt (*Speech-driven Multimodal Automatic Directory Assistance*) werden mobile Geräte zur Interaktion mit intelligenten Umgebungen genutzt [OKJ+01]. Dabei werden die grafischen Benutzerschnittstellen um *multimodale* Aspekte angereichert, d. h., es kommen Elemente zur Sprach- und Handschrifterkennung hinzu, die eine Erweiterung der üblichen Benutzerinteraktion mit der Umgebung ermöglichen.

Weitere Ansätze, die sich mit der Benutzerinteraktion in intelligenten Umgebungen beschäftigen, werden in den Projekten *EMBASSI* [KHS01] und *SIKOWO* [Bay10] verfolgt.

In allen genannten Projekten werden mobile Geräte lediglich als Benutzerschnittstelle zur Interaktion mit der Umgebung verwendet. Die vorliegende Arbeit erweitert diesen Ansatz, indem sie dem Benutzer zusätzlich die Ausführung von Diensten auf seinem Handheld sowie die Mitnahme seiner Daten auf diesem ermöglicht.

9.2 Ansätze zur Benutzermodellierung und Personalisierung

Laut Kobsa [Kob07] wurden in der Vergangenheit viele unterschiedliche Systeme zur Benutzermodellierung entwickelt, die sich meist in ihrer Anwendungsdomäne unterscheiden. Im Folgenden werden einige dieser Systeme vorgestellt und mit dem im Rahmen dieser Arbeit entwickelten Benutzermodell verglichen.

9.2.1 CUMULATE

CUMULATE ist ein System zur Modellierung von Benutzern im Anwendungsbereich E-Learning [BSS05]. Es handelt sich dabei um einen generischen Server, der von unterschiedlichen,

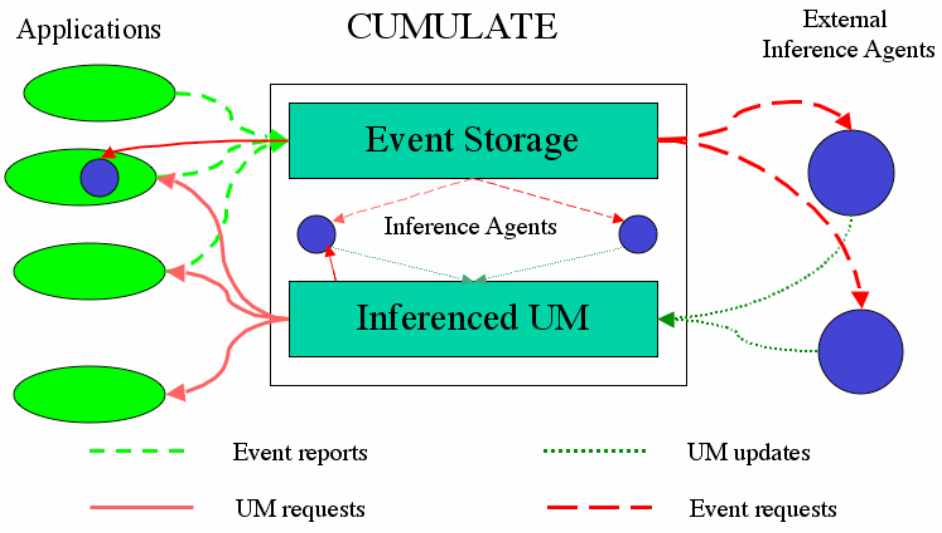


Abbildung 9.5: Grobarchitektur von CUMULATE (Quelle: [BSS05]).

verteilten Anwendungen verwendet werden kann. Sie können dabei sowohl Benutzerdaten beim Server anfragen als auch neue Daten erzeugen.

Der Server selbst speichert die Daten auf zwei Ebenen, wie in Abbildung 9.5 dargestellt ist. Eingehende Daten werden zunächst in Rohform im *Event Storage* abgelegt. Aus diesen können interne bzw. externe Inferenzagenten neue Informationen generieren und sie im *Inferenced UM* speichern. Die Daten werden in Form von Attributen als Name-Werte-Paare modelliert. Die Anwendungen können Daten sowohl aus dem Event Storage oder aus dem Inferenced UM anfragen.

9.2.1.1 Vergleich

Während sich das Hauptziel von CUMULATE sich mit dem des im Rahmen dieser Arbeit entwickelten Benutzermodells deckt, weisen beide Ansätze einige Unterschiede auf. In der vorliegenden Arbeit wird nicht ein zentralistischer Ansatz wie bei CUMULATE verfolgt. Stattdessen existiert in jedem eHome und auf jedem Handheld eine eigene Instanz des Benutzermodells. Ein Synchronisierungsmechanismus sorgt für die Konsistenz der Daten, wobei jeder Benutzer die Gesamtheit seiner Daten auf dem mobilen Benutzermodell verfügbar hat. Dadurch behält der Benutzer die Kontrolle über seine Daten, wodurch der Schutz der Privatsphäre verbessert wird.

9.2.2 Personis, Personis-lite und PersonisAD

Personis ist ein generisches System zur Benutzermodellierung, das an der Universität Sydney entwickelt wurde [KKL02]. Es basiert auf dem Accretion/Resolution-Ansatz (s. Abschnitt 2.1.1) und verwaltet Benutzerdaten in hierarchischen Namensräumen und Sichten. Dabei können die Sichten als eine Art Zugriffskontrolle verstanden werden, weil auf Ba-

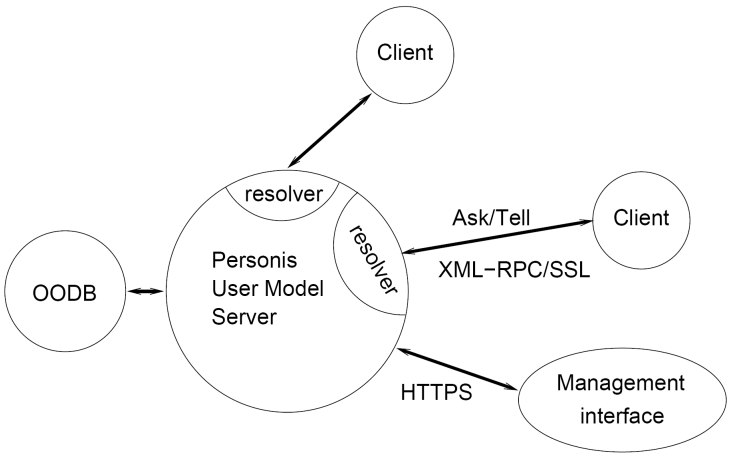


Abbildung 9.6: Architekturübersicht über Personis (Quelle: [KKL02]).

sis dieser Schichten entschieden wird, welche Anwendungen auf welche Daten zugreifen dürfen [KKL03].

Eine Übersicht über die Personis-Architektur ist in Abbildung 9.6 dargestellt. Anwendungen, die sogenannten *Clients*, können Daten an Personis liefern oder Daten von Personis anfordern. Wenn sie Daten anfordern, werden die gewünschten Information von den *Resolvern* bereitgestellt. Gegebenenfalls werden die gewünschten Informationen durch die Analyse von Rohdaten ermittelt. Die Kommunikation findet auf Basis von XML statt und ist durch SSL gesichert. Die Benutzer können ihre Daten über eine Managementschnittstelle verwalten.

Die Entwickler haben ferner *Personis-lite*, eine abgespeckte Version von Personis, entwickelt, die auf mobilen Geräten ausgeführt werden kann [CKK05]. Die Anwendungsdomäne von Personis-lite ist das Ubiquitous Computing, in der auch Geräte und Sensoren ähnlich wie Benutzer modelliert werden. Dabei erhält jedes Gerät, jeder Sensor und jeder Benutzer ein eigenes Modell. Für die Integration der Modelle werden die Attribute `seen` und `seenby` verwendet. Wenn ein Sensor beispielsweise ein Gerät erkennt, werden die ID des Geräts im Attribut `seen` des Sensor-Modells und umgekehrt die ID des Sensors im Attribut `seenby` des Geräte-Modells festgehalten.

Eine weitere Entwicklung dieser Gruppe ist *PersonisAD* [ACKK07]. Es basiert auf einem Ansatz, der sogenannte *Active Models* [ACKK06] in Personis integriert. Das dabei verfolgte Ziel ist die Unterstützung der Entwickler (personalisierbarer) Anwendungen für das Ubiquitous Computing, und damit auch für eHomes. Wie bei Personis-lite werden die Entitäten einer Umgebung mit eigenen Modellen ausgestattet, sodass Kontextänderungen in der Umgebung umgehend an die Dienste weitergeleitet werden können. Einzelne Attribute der Modelle können dabei mit Regeln der Form

```
value ~ pattern : action
```

assoziiert werden. Dabei steht `value` für den Wert des Attributs, `pattern` für einen regulären Ausdruck und `action` für eine Aktion (`tell` bzw. `ask`). Solche Regeln werden ausgewertet,

wenn neue Ereignisse auftreten, die das zugehörige Attribut betreffen. Abhängig von der Art des Ereignisses können neue Daten generiert oder Anwendungen benachrichtigt werden.

Auf Basis dieses Ansatzes wurden die Anwendungen *MyPlace* und *MusicMix* entwickelt. MyPlace dient der Lokalisierung von Personen in einem Gebäude. MusicMix hingegen spielt die Wunschmusik der Personen ab, die sich in einem Raum befinden. Benutzerkonflikte werden durch die Kombination der Wiedergabelisten der Benutzer gelöst.

9.2.2.1 Vergleich

Viele der Funktionalitäten von Personis, insbesondere von Personis-lite und PersonisAD werden in eHomes benötigt. In diesen Ansätzen wurde ein besonderes Augenmerk auf die Lokalisierung von Personen in Gebäuden gelegt.

Dieser Aspekt war für die vorliegende Arbeit weniger von Bedeutung. Erstens liefern die im Rahmen dieser Arbeit entwickelten Demonstratoren präzise Informationen über den Aufenthaltsort der Benutzer. Zweitens wurde für reelle Umgebungen die Verfügbarkeit einer angemesseneren Technik angenommen. Personis setzt ferner voraus, dass die Benutzer dem System zuvor bekannt sein müssen, um erkannt zu werden. Der hier vorgestellte Ansatz kann aufgrund der Nutzung von Anonymisierungstechniken auch zuvor unbekannte Benutzer handhaben. Ferner ist aus der Literatur nicht ersichtlich, wie die Privatsphäre Benutzer und der Schutz von Benutzerdaten in Personis gewährleistet wird. Dieser Aspekt macht hingegen einen wesentlichen Teil dieser Arbeit aus. Schließlich bietet das Benutzermodell in dieser Arbeit selbst keine Resolver zum Ableiten neuer Informationen aus den Rohdaten an, es können jedoch leicht welche hinzugefügt werden.

9.2.3 Der UserModelService

Das Konzept der *ubiquitären Benutzermodellierung* wurde erstmals von Heckmann in [Hec01] eingeführt und in weiteren Arbeiten vertieft [Hec06, Hec03a, HSBK05, HSB+05]. Heckamnn verfolgt in seinen Arbeiten das Ziel, durch einen geeigneten Benutzermodellierungsansatz die Adaptionsleistungen von Anwendungen und intelligenten Umgebungen an ihre Benutzer zu verbessern.

Um ein einheitliches Verständnis und einen Austausch von Benutzerdaten zwischen unterschiedlichen Anwendungen zu gewährleisten, wurde auf semantischer Ebene die Benutzermodellontologie GUMO entwickelt. Ein zentraler webbasierter Benutzermodelldienst, der sogenannte *UserModelService*, ist für die anwendungsunabhängige Verwaltung von Benutzerdaten zuständig.

Abbildung 9.7 zeigt eine prozedurale Übersicht über die Schnittstellen des UserModelService. Benutzer, Sensoren oder Anwendungen können mithilfe der Sprache *UserML*, die speziell für den Austausch von Benutzerdaten entwickelt wurde, neue Daten in Form von SituationalStatements (s. Abschnitt 4.4.3) an den *User Model Adder* senden (1). Dieser speichert die eingehenden Daten in verschiedenen Datenbanken ab, die verteilt sein können (2). Falls eine Anwendung Benutzerdaten mithilfe der Anfragesprache UserQL anfragt (3), werden die betroffenen Daten aus den zugehörigen Datenbanken ermittelt (4.1). Bevor

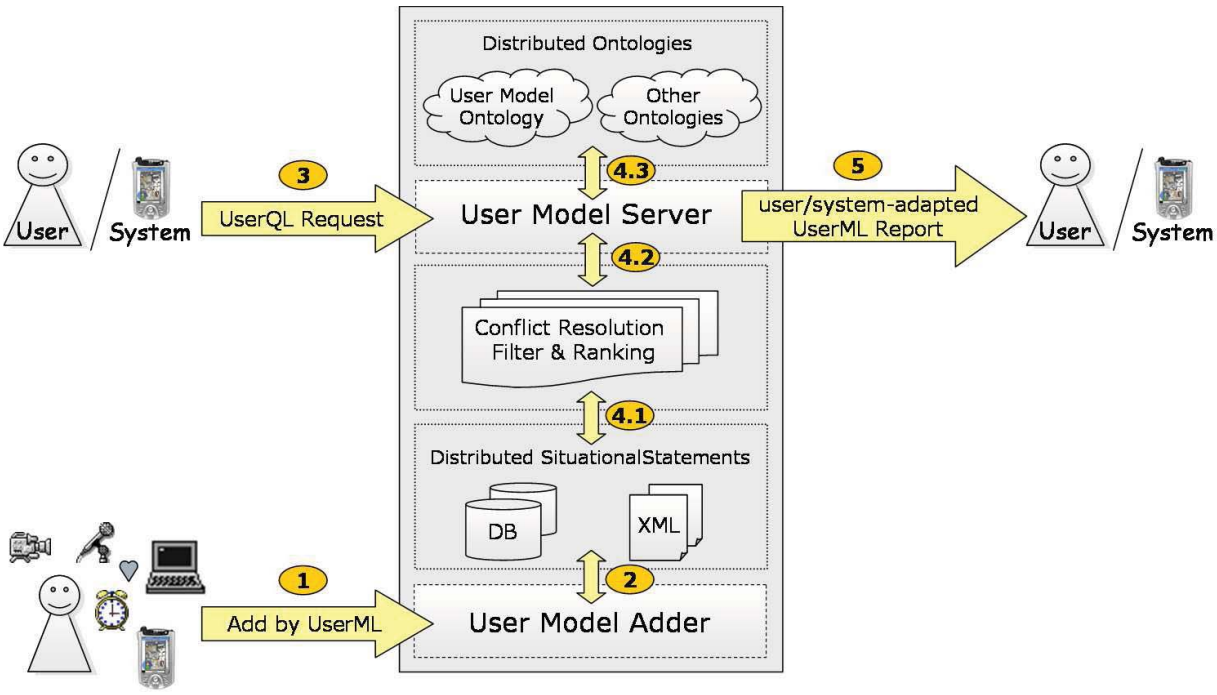


Abbildung 9.7: Prozedurale Übersicht über den UserModelService (Quelle: [Hec06]).

sie jedoch an die Anwendung übermittelt werden, werden zunächst mögliche Konflikte gelöst (4.2) und semantische Interpretationen durchgeführt (4.3). Schließlich werden die Informationen der Anwendung über UserML gesendet. Die Kommunikation zwischen den Entitäten basiert dabei auf HTTP.

9.2.3.1 Vergleich

Das Benutzermodell der vorliegenden Arbeit profitiert teilweise von den Ergebnissen der Arbeiten von Heckmann. Insbesondere wurde die Ontologie GUMO als semantische Grundlage für den Austausch von Benutzerdaten zwischen dem Benutzermodell und den eHome-Diensten herangezogen (s. Abschnitt 4.4). Ferner wurde das Konzept der SituationalStatements beim Entwurf der Architektur des Benutzermodells berücksichtigt.

Im Gegensatz zu Heckmann wurde im Rahmen dieser Arbeit jedoch kein zentralistischer Ansatz für den Benutzermodelldienst verfolgt. Stattdessen besitzt jedes eHome und jedes Handheld ein eigenes Benutzermodell. Dadurch wird vermieden, dass eine einzelne Organisation die Daten und Bewegungsprofile aller Benutzer kennt. Ferner ist aus den Arbeiten von Heckmann nicht ersichtlich, ob und wie Anonymität realisiert wurde, die einen wesentlichen Bestandteil der vorliegenden Arbeit ausmacht.

9.2.4 Weitere Ansätze

In diesem Abschnitt werden einige weitere Ansätze beschrieben, die sich mit dem Aspekt der Benutzermodellierung befasst haben.

In [Bun05] wird ein gemischter Ansatz für die Personalisierung von Microsoft Word vorgestellt. Die Personalisierung kann dabei sowohl direkt durch den Benutzer (*Anpassbarkeit*) als auch auf Basis eines Benutzermodells (*Adaptivität*) vorgenommen werden. Bevor das System erste Vorschläge zur Anpassung der Benutzeroberfläche macht, werden Auswirkungen auf die Performanz des Benutzers untersucht. Dazu wird eine vereinfachte Variante der *GOMS*-Analyse eingesetzt, GOMS steht dabei für Goals, Operators, Methods und Selection [CNM83].

In [CCP04] wird das Multiagentensystem *D-Me* (Digital Me) beschrieben. Je ein Agent unterstützt seinen Benutzer bei der Interaktion mit intelligenten Umgebungen, beispielsweise bei der Abarbeitung seiner Aufgaben. Für die Unterstützung der Mobilität sieht das System ein hierarchisches Benutzermodell vor. In jeder Umgebung kann nur auf den zugehörigen Bereich des Benutzermodells zugegriffen werden. Eine Art Zugriffskontrolle auf die Daten wird dadurch realisiert, indem die Daten als öffentlich oder privat deklariert werden können.

Ein weiterer Ansatz zur verteilten Modellierung von Benutzern in intelligenten Umgebungen wird in [SLZ05, Lor05] vorgestellt. Das System basiert auf einem Multiagentenansatz, wobei jeder Sensor oder Aktor durch einen Agenten repräsentiert wird. Lokale Informationen über Benutzer werden über Sensoren gesammelt und in lokalen Benutzermodellen gespeichert, die selbst wiederum Agenten sind. Ferner werden die Informationen über sogenannte Broker anderen interessierten Agenten zugänglich gemacht. Somit sind die Benutzerdaten innerhalb der Umgebung verteilt in einzelnen Geräten enthalten und werden über Broker ausgetauscht.

Auch in [HWRB05] wird ein agentenbasierter Ansatz zur Modellierung von Benutzern in intelligenten Umgebungen wie eHomes verfolgt. Jedes Gerät wird hier durch einen Treiber kontrolliert. Die von dem Treiber gesammelten Daten werden durch einen zugehörigen Agenten analysiert. Jeder Agent leitet seine Informationen wiederum weiter an eine zentrale Einheit, die alle Daten in einer Wissensbasis entsprechend der Ontologie SOUPA [CPFJ04] speichert. Die zentrale Einheit kann zudem Kommandos generieren und an die Agenten schicken. Der Fokus dieses Ansatzes ist insbesondere auf die Interpretation gesammelter Rohdaten beispielsweise für die Interaktion von Benutzern und Robotern gerichtet. Datenschutzaspekte werden nicht betrachtet.

In [Moz98] wird ein intelligentes Haus beschrieben, das sich auf Basis neuronaler Netze an die Bedürfnisse und Gewohnheiten seiner Bewohner anpassen kann. Beispielsweise kann es das Licht und die Raumtemperatur oder die Lautstärke der Musik anpassen. Dabei versucht das Haus, einen Kompromiss zwischen Komfort und Energieeffizienz zu treffen. Der Heizungsdienst versucht z. B. die Temperatur niedriger einzustellen als vom Benutzer gewünscht, um Energie zu sparen.

Schließlich wird in [FKJ97] ein System beschrieben, das im Rahmen des *AVANTI*-Projekts entwickelt wurde. Es verwendet ein Benutzermodell, um die Anzeige von Multimediainformationen an den Benutzer anzupassen, der die Informationen anfordert. Die

Darstellung kann dabei anhand unterschiedlicher Kriterien wie Sprache, Textgröße oder auch Behinderungen angepasst werden. Dieses System ist sowohl für Webseiten einsetzbar als auch für öffentliche Informationsdisplays. Die Benutzer können sich an einem Display kontaktlos anmelden, indem sie ein RFID-Chip verwenden. Wird solch ein Chip vorgehalten, werden die zugehörigen Daten aus dem zentralen Benutzermodell angefordert, um die Anzeige an den Benutzer anzupassen. Während der Benutzer mit dem System agiert, werden weitere Information über das Benutzerverhalten gesammelt und für zukünftige Nutzung an den zentralen AVANTI-Server übermittelt.

9.3 Ansätze zur Privatsphäre und Sicherheit

In diesem Abschnitt werden verwandte Arbeiten beschrieben, die sich mit der Privatsphäre bzw. Sicherheit. Zunächst werden Ansätze zur Datensparsamkeit beschrieben. Anschließend werden Ansätze zur Anonymität auf Basis von MIXen vorgestellt. Hiernach folgen einige konkrete Projekte, die sich mit der Privatsphäre und Sicherheit in intelligenten Umgebungen befasst haben. Danach wird ein Ansatz vorgestellt, der Dienstanbieter bei der Einhaltung von Datenschutzrichtlinien unterstützt. Abschließend wird ein Überblick über einige weitere Ansätze aus dem Themengebiet zu Privatsphäre und Sicherheit gegeben.

9.3.1 Datensparsamkeit

Datensparsamkeit soll oft durch Identitätsmanagement erreicht werden, weil ein Benutzer dadurch entscheiden kann, welche Daten er seinen Kommunikationspartnern gegenüber offenlegt. Dabei wird das Ziel verfolgt, die Menge der offengelegten Daten zu minimieren.

In Folgenden werden einige Ansätze vorgestellt, die sich mit Identitätsmanagement und Datensparsamkeit beschäftigen. Diese Ansätze sind für die Verwendung im Internet vorgesehen, können aber auch in intelligente Umgebungen übertragen werden, wie in Abschnitt 9.3.1.3 gezeigt wird.

9.3.1.1 PRIME

PRIME ist ein von der EU gefördertes Forschungsprojekt zu Identitätsmanagement und Privatsphäre im Internet [HBPP05, CSS+05, PRI10a]. Zu den Projektpartnern gehörten unter anderem IBM und T-Mobile sowie viele europäische Universitäten. Die RWTH Aachen gehörte ebenfalls zu den Projektpartnern. Zwar lief das PRIME-Projekt Anfang 2008 ab, die Forschung wird jedoch im Nachfolgeprojekt *PrimeLife* [Pri10b] fortgeführt.

Das Ziel von PRIME ist die Erforschung eines Identitätsmanagementsystems, mit dem Benutzer Informationsdienste nutzen und gleichzeitig ihre Privatsphäre schützen können. Besonderes Augenmerk wird darauf gelegt, dass Personen die Kontrolle über ihre Daten erhalten. Sie sollen selbst entscheiden können, welche Daten an wen weitergegeben werden dürfen und wie mit ihnen umgegangen werden soll. Dazu können zwischen Benutzern und Organisationen verbindliche Richtlinien ausgehandelt werden, in denen der Umgang mit persönlichen Daten festgeschrieben wird.

Ein weiterer wichtiger Aspekt bei PRIME ist die Ausrichtung nach dem Grundsatz der Datensparsamkeit. Alle Teilnehmer einer Transaktion sollen nur die Daten sammeln, die sie für die Transaktion brauchen. Nach der Transaktion sollen die Daten wieder gelöscht werden. Die Daten sollen immer so allgemein wie möglich gehalten werden, sodass wenig Alleinstellungsmerkmale darin enthalten sind. Statt zum Beispiel einen Nachweis über das tatsächliche Alter zu erbringen, soll die Aussage „Person ist älter als 18 Jahre“ ausreichen.

PRIME behandelt auch das Thema Anonymität. Es wird nicht versucht, vollständige Anonymität zu erreichen, sondern immer einen ausgewogenen Grad davon. Bei unsensiblen Transaktionen, zum Beispiel dem Betrachten einer Internetseite, wird nur ein geringer Grad vorgesehen. Wenn die Transaktionen sensibler sind, können anonyme Credentials benutzt werden. Als anonymes Credential-System wird idemix eingesetzt, um die Unverkettbarkeit von Transaktionen zu gewährleisten.

Standardmäßig wird jede Transaktion über ein anonymes Netzwerk durchgeführt. Zum Einsatz kommen dabei zum Beispiel MIXe (s. Abschnitt 9.3.2). Außerdem werden Pseudonyme eingesetzt. Unterschieden wird dabei zwischen einfachen Pseudonymen, bestehend aus zufälligen Zeichenketten, und kryptografischen Pseudonymen, die durch idemix realisiert werden. Hierbei wird die Gestaltung der Pseudonyme am Grad der Anonymität ausgerichtet.

Vergleich Die von PRIME verfolgten Ziele gleichen sehr stark denen, die im Rahmen dieser Arbeit verfolgt wurden. Ferner wird in beiden Ansätzen idemix eingesetzt. Auch die Möglichkeit, zwischen unterschiedlichen Anonymitätsstufen für eine Sitzung wählen zu können, ist beiden Ansätzen gleich. Während in dieser Arbeit der Schutz der Privatsphäre in eHome betrachtet wurde, verfolgt PRIME das gleiche Ziel hingegen im Internet. Daher unterschieden sich die zugrunde liegenden Architekturmodelle. Im Gegensatz zur vorliegenden Arbeit sieht PRIME keine zentrale Stelle vor, an der eine Authentifizierung erfolgen muss, um Dienste zu nutzen. Stattdessen wird die Authentifizierung der Benutzer jeweils direkt von den jeweiligen Webdiensten vorgenommen. Folglich existiert keine Interceptor-ähnliche Komponente in PRIME, die die Zugriffskontrolle durchführt. Ein weiteres Unterscheidungsmerkmal beider Ansätze liegt in der Anonymisierung der Kommunikation auf der Netzwerkebene. Weil PRIME im Internet genutzt wird, können hier MIXe eingesetzt werden. In der vorliegenden Arbeit kann der MIX-Ansatz nicht angewendet werden, weil keine Zwischenknoten zwischen dem Handheld eines Benutzers dem eHome-Gateway existieren. Um eine Wiedererkennung des Handhelds zu verhindern, können jedoch Identifikatoren wie die MAC- oder IP-Adresse nach jeder Sitzung softwaregestützt geändert werden. Ferner wurde im Rahmen dieser Arbeit ein einfacher Ansatz zur Verabredung von Richtlinien dadurch realisiert, dass die Benutzer bestimmen können, welche Dienste auf welche Daten zugreifen dürfen. Der Benutzer kann in beiden Fällen jedoch nicht überprüfen, ob sich ein eHome oder ein Webdienst an die Richtlinien hält.

9.3.1.2 Platform for Privacy Preferences - P3P

Die *Platform for Privacy Preferences* (P3P), die 2002 vom World Wide Web Consortium (W3C) verabschiedet wurde [Lan01a, CL02, CLMR06], ist eine technische Plattform zum

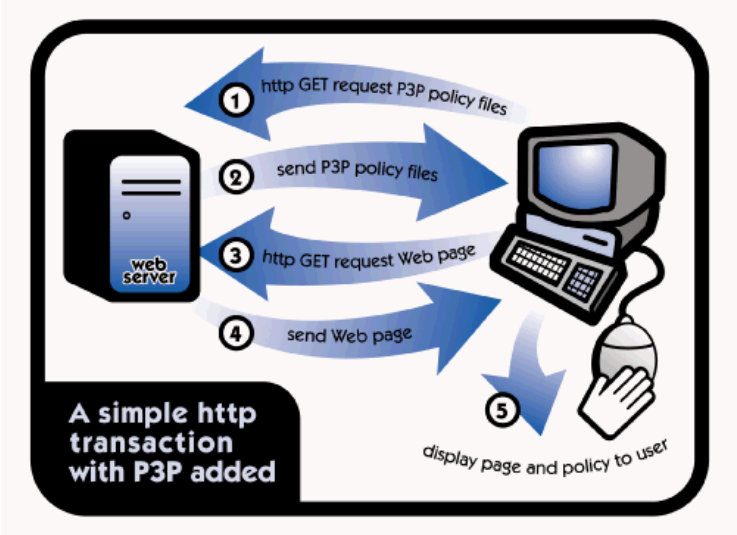


Abbildung 9.8: Ablauf eines Webseitenaufrufs mit P3P (Quelle: [Wor10]).

Austausch von Datenschutzinformationen. Sie soll den Besuchern von Webseiten eine bessere Kontrolle über die Erhebung und Verarbeitung von personenbezogenen Daten ermöglichen. Dabei wird eine standardisierte Syntax verwendet, mit der eine maschinenlesbare Fassung der Datenschutzerklärung der Webseite erstellt werden kann. Diese Fassung liegt in einem XML-Dokument vor und beinhaltet unter anderem die Informationen darüber, wer Daten sammelt, welche Daten gesammelt werden und wofür diese Daten benötigt werden.

Durch die Maschinenlesbarkeit kann die Erklärung durch Software interpretiert und automatisch mit den Präferenzen des Benutzers abgeglichen werden. Abbildung 9.8 zeigt den Ablauf eines Webseitenaufrufs mit P3P. Wenn der Benutzer eine Webseite aufruft, erfragt der Browser zuerst die P3P-Datenschutzrichtlinie der Webseite. Anschließend wird die Webseite aufgerufen. Der Browser vergleicht währenddessen die Richtlinien der Webseite mit den Datenschutzpräferenzen des Benutzers, die er einmalig in einem maschinenlesbaren Format festgelegt. Wenn keine vollständige Übereinstimmung vorliegt, wird der Benutzer darauf hingewiesen. Solch ein Hinweis geschieht in vielen Browsern durch grafische Symbole [LCM02]. Dadurch hat der Benutzer eine Grundlage für seine Entscheidung, persönliche Daten herauszugeben oder für sich zu behalten.

Vergleich P3P hat den Vorteil, dass sich Benutzer nicht immer wieder durch komplizierte Datenschutzerklärungen wühlen müssen, weil der Abgleich der Datenschutzrichtlinien automatisch erfolgt. Ein Nachteil von P3P besteht jedoch darin, dass es für den Benutzer vorab nicht prüfbar ist, ob die Webseite die abgegebene Erklärung auch einhält. Es muss daher bereits eine Vertrauensbasis bestehen, da der Benutzer sonst trotz Datenschutzerklärung nicht bereit sein dürfte, seine Daten freizugeben. In der vorliegenden Arbeit wird die Datensparsamkeit dadurch erreicht, dass ein Benutzer interaktiv über sein Handheld mit dem eHome aushandeln, welche Dienste er nutzen und welche Daten er preisgeben möchte. Diese Aushandlung geschieht jedoch nicht automatisch, zumindest nicht beim

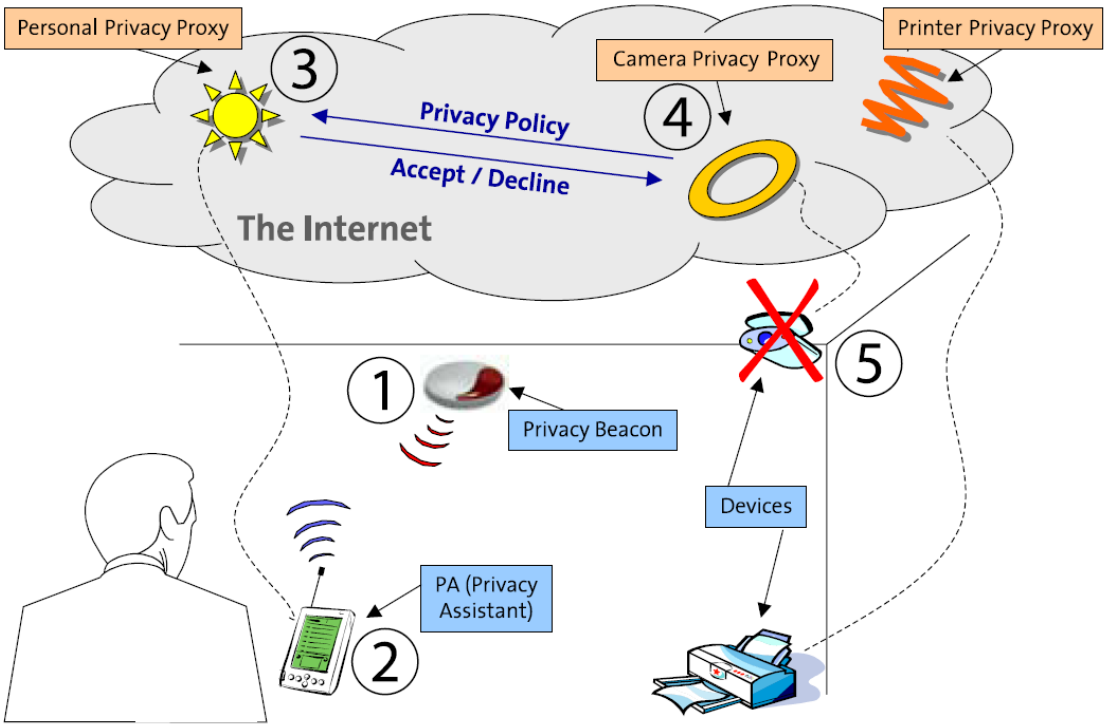


Abbildung 9.9: Überblick über den Aufbau von PawS (Quelle: [Lan02]).

ersten Anmelden des Benutzers in einem neuen eHome. Für weitere Anmeldungen beim gleichen eHome können die Ergebnisse der ersten Aushandlung aber automatisch ausgewählt werden. Eine mögliche Automatisierung für die Erstanmeldung mit P3P wäre für zukünftige Erweiterungen denkbar.

9.3.1.3 PawS

In [Lan02] wird das System *PawS* vorgestellt, mit dem P3P für intelligente Umgebungen genutzt werden kann. In diesem System spielen sogenannte *Privacy Proxies* die entscheidende Rolle. Der Privacy Proxy eines Benutzers speichert seine Präferenzen bezüglich der Privatsphäre in Form einer P3P-Richtlinie. Der Benutzer trägt ein mobiles Gerät mit sich, das aufgrund knapper Ressourcen nicht selbst das Privacy Proxy enthält, sondern eine Verbindung zu diesem vermittelt, der sich im Internet befindet. Jeder in einer intelligenten Umgebung installierte Dienst besitzt ebenfalls einen Privacy Proxy, der die Datenschutzerklärung des Dienstes speichert. Sogenannte *Privacy Beacon* weisen den Benutzer darauf hin, dass ein Dienst Daten sammelt, und machen gleichzeitig den Privacy Proxy des zugehörigen Dienstes bekannt.

Betritt nun ein Benutzer eine intelligente Umgebung, werden zwischen den Proxies die Erklärungen ausgetauscht und abgeglichen. Durch den Abgleich können vorhandene Dienste so konfiguriert werden, dass sie sich entsprechend der Präferenzen des Benutzers bzgl. der Privatsphäre verhalten. Abbildung 9.9 zeigt eine Umgebung mit einer Kamera und einem Drucker. Ferner sind die Privacy Proxies des Benutzers und der Geräte dargestellt. Der

Benutzer betritt nun die Umgebung, ist aber nicht damit einverstanden, dass die Kamera die Räume überwacht. Durch den Abgleich der Benutzerpräferenzen mit den vorherrschenden Bedingungen wird eine verbindliche Übereinkunft getroffen. Diese enthält Informationen darüber, welche Daten ausgetauscht werden, wer diese Daten verwendet und wofür sie verwendet werden. Teil der Übereinkunft ist die Abschaltung der Kamera, weil deren Datenschutzerklärung nicht den Präferenzen des Benutzers entspricht.

Vergleich Mit PawS wird das Ziel der Datensparsamkeit mithilfe von P3P verfolgt. Der dort vorgeschlagene Ansatz weist einige Ähnlichkeiten zum Aushandlungsansatz auf, der im Rahmen der vorliegenden Arbeit vorgeschlagen wurde. Ein Unterschied zwischen beiden Ansätzen besteht darin, dass der Benutzer seine Präferenzen mit PawS nur einmal festlegen muss und diese für jede Umgebung gültig sind. Im Rahmen der vorliegenden Arbeit kann er hingegen von der Flexibilität profitieren, unterschiedlichen eHomes unterschiedliche Mengen von Daten offenzulegen. Dadurch wird der Tatsache Rechnung getragen, dass der Benutzer für jedem eHome unterschiedlich stark vertraut. Eine einmal getroffene Entscheidung kann für das zugehörige eHome jedoch gespeichert und beim zukünftigen Betreten desselben eHomes automatisch ausgewählt werden. Ein weiterer Unterschied zu PawS liegt darin, dass PawS alle Benutzerdaten, zusammen mit der ausgehandelten Übereinkunft, in einer Internetdatenbank speichert. Es wird jedoch nicht näher erläutert, wo sich diese Datenbank befindet und wer sie betreibt. Dieser zentralistische Ansatz widerspricht, wie bereits erwähnt, dem dezentralen Benutzermodellansatz, der im Rahmen dieser Arbeit verfolgt wurde. Ferner setzt PawS eine dauerhafte Internetverbindung voraus, weil sich das Privacy Proxy des Benutzers nicht auf dem mobilen Gerät, sondern im Internet befindet. Im eHome-Projekt ist eine Internetverbindung nicht notwendig, weil der Benutzer alle benötigten Informationen auf seinem Handheld mit sich führt.

9.3.2 Anonymität durch MIXe

MIXe wurden erstmals in [Cha81] vorgestellt und im Laufe der Zeit in weiteren Arbeiten erweitert. Das verfolgte Ziel dabei ist der Schutz der *Kommunikationsbeziehung* zwischen Sendern und Empfängern von Nachrichten. Zum Schutz der Privatsphäre tragen MIXe insofern bei, als dass sie Senderanonymität gewährleisten können.

Die Unverfolgbarkeit von Nachrichten von einem Sender zu einem Empfänger wird über mehrere Zwischenknoten, den sogenannten MIXen, gewährleistet. Dabei verschlüsselt der Sender seine Nachricht mehrfach und sendet sie über das MIX-Netz an den Empfänger. Jedes MIX sammelt genügend viele Nachrichten und versendet sie so verändert weiter, dass Außenstehende die versendeten Pakete nicht zu den Eingangspaketen zuordnen können. Die Veränderung ist dabei das Ergebnis der schrittweisen Entschlüsselung und Umsortierung der Nachrichten [Kes99].

Abbildung 9.10 zeigt ein schematisches MIX-Netz. Zu betrachten ist die Abbildung von links nach rechts. Links oben wird eine Nachricht erzeugt und zweimal verschlüsselt. Aus der ursprünglichen Nachricht wird Nachricht M2 und nach deren Verschlüsselung Nachricht M1. Die so verschlüsselte Nachricht ist hier als blaues Quadrat mit den enthalten

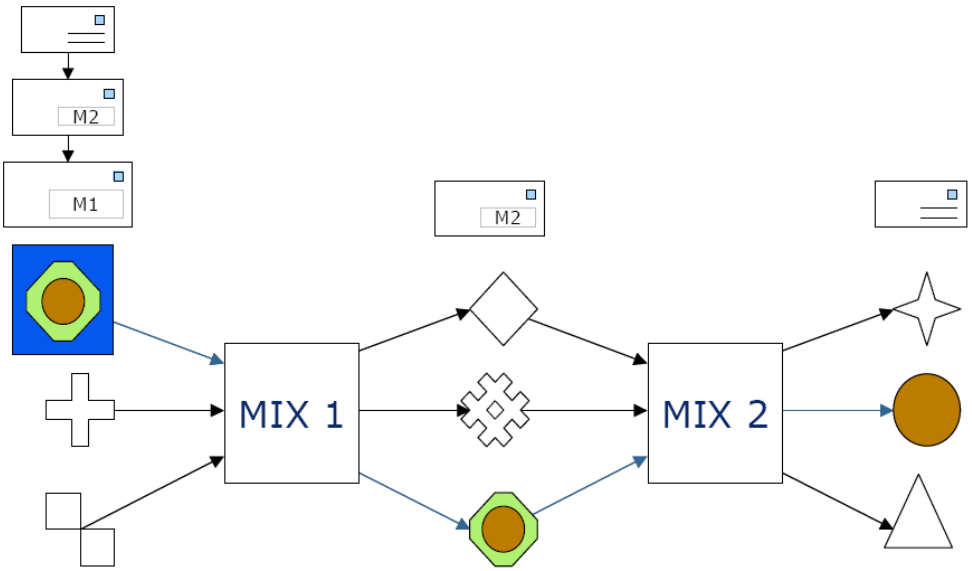


Abbildung 9.10: Konzept eines MIX-Netzes (Quelle: [FJMP97]).

Verschlüsselungen dargestellt, die andere Formen aufweisen. Diese so verschlüsselte Nachricht geht mit anderen verschlüsselten Nachrichten, die alle eine unterschiedliche Form aufweisen, in den ersten MIX. Dieser entfernt eine Verschlüsselung und sortiert sie um. Ein Beobachter von außen könnte nun schon nicht mehr nachvollziehen, welche eingegangene Nachricht zu welcher ausgegangenen gehört. Sehr wohl könnte dies noch der MIX selbst. Dieses Problem erledigt sich aber mit dem zweiten MIX. Der zweite MIX nimmt wieder eine Verschlüsselung fort, sortiert die Nachrichten um und stellt dem Empfänger die Nachricht zu.

Im Folgenden werden zwei Ansätze beschrieben, die den MIX-Ansatz auch in intelligenten Umgebungen einsetzen.

9.3.2.1 MIXe in Gaia

In [AMCK+02] wird das sogenannte *Mist* beschrieben. Es handelt sich dabei um einen MIX-Ansatz, der im Rahmen des Gaia-Projekts (s. Abschnitt 9.1.1) realisiert wurde. Das dabei verfolgte Ziel ist die Geheimhaltung des Aufenthaltsortes von Benutzern. In Mist existieren spezielle Portale, zu denen sich ein Benutzer über das Mist-Netzwerk verbinden und darüber mit Diensten kommunizieren kann. Durch das Mist-Netzwerk wird der Aufenthaltsort des Benutzers vor dem Portal und den Diensten verborgen. Weitere Anmerkungen zur Sicherheit und Privatsphäre in Gaia folgen in Abschnitt 9.3.3.3.

9.3.2.2 MUSDAC

Eine weitere Plattform, die den Schutz der Privatsphäre von Benutzern in intelligenten Umgebungen auf Basis von MIXen gewährleistet, ist das in [RICLC06] vorgestellte *Multi-protocol Service Discovery and Access* (MUSDAC). MUSDAC wurde primär mit dem Ziel entwickelt, das Auffinden von Diensten über verschiedene Protokolle und über Umgebungs-

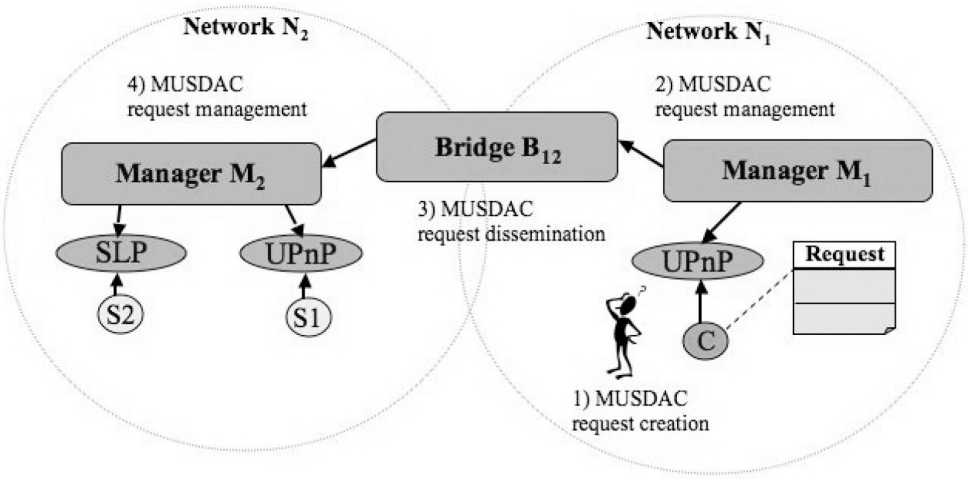


Abbildung 9.11: Interaktion zwischen MUSDAC-Komponenten (Quelle: [CRI07a]).

grenzen hinweg zu ermöglichen. Es bildet ein virtuelles Netzwerk, bestehend aus sogenannten *Managern*, *Bridges* und *Transformern*. Ein solches Netzwerk ist in Abbildung 9.11 dargestellt. Ein Manager verarbeitet Such- und Zugriffsanfragen innerhalb einer Umgebung. Bridges verbinden mehrere Umgebungen miteinander und geben Such- und Zugriffsanfragen an entfernte Manager weiter. Ein Manager ist ferner mit einem oder mehreren Transformern verbunden. Die Transformer übersetzen MUSDAC-Anfragen in Anfragen der verschiedenen Protokolle und verbinden so die Manager mit den Diensten und Benutzern.

Setzt ein Benutzer eine Suchanfrage an seinen lokalen Manager ab, so versucht der Manager einen passenden Dienst zu finden. Findet er in der lokalen Umgebung keinen passenden Dienst, gibt er die Anfrage über Bridges an entfernte Manager weiter. Die Anfrage wird so weit verbreitet, bis passende Dienste gefunden werden. Eine Anfrage kann vom Benutzer mit einer Beschreibung des gesuchten Dienstes versehen werden. Diese Beschreibung kann private Informationen enthalten, die geschützt werden sollen. Auch die Information, welcher Benutzer welche Dienste sucht, ist sensibel. Daher wird MUSDAC in [CRI07b] um Maßnahmen erweitert, die den Schutz der Privatsphäre gewährleisten sollen. Insbesondere wird auf Basis eines zugrunde liegenden MIX-Netzwerks die Quelle der Anfrage vor den beteiligten Managern geheim gehalten, die Anfrage wird also anonymisiert. Dadurch wird die Verkettbarkeit mehrerer Anfragen und somit die Erstellung eines Dienstnutzungsprofils durch externe Manager unterbunden.

9.3.2.3 Vergleich

Obwohl MIXe einen guten Ansatz zur Anonymisierung von Kommunikationsbeziehungen bieten, eignet sich dieser Ansatz nicht für den Einsatz im eHome-Projekt. Der eHome-Prototyp sieht nämlich keine Zwischenknoten zwischen dem Handheld eines Benutzers und dem eHome-Gateway vor, die Anonymisierung der Netzwerkkommunikation durch MIXe ist somit nicht möglich. Darüber hinaus wurde im Rahmen der vorliegenden Arbeit eine mögliche Vernetzung mehrerer eHomes explizit verworfen (s. Abschnitt 5.2.2). Ferner muss

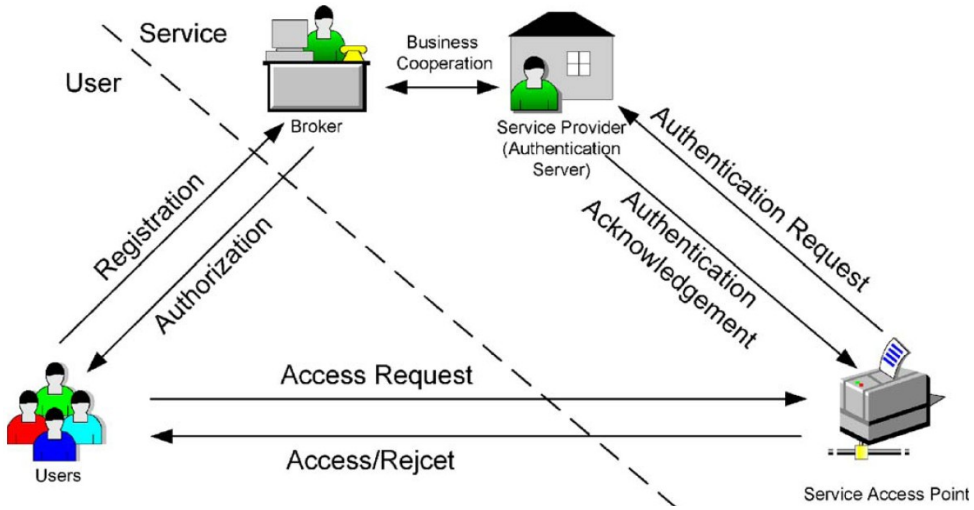


Abbildung 9.12: Architektur des Ansatzes von Lou et al. (Quelle: [RL07]).

ein Benutzer keine Suchanfragen stellen, die über ein verteiltes Netzwerk propagiert wird. Stattdessen erhält er vor seiner Anmeldung beim eHome die Liste der verfügbaren Dienste und kann mit ihnen, falls gewünscht anonym, interagieren.

MIXe werden häufig verwendet, um Dienstzugriffe geheim zu halten. Dies kann nur funktionieren, wenn jeder Dienst selbst die Zugriffskontrolle durchführt. Im Rahmen der vorliegenden Arbeit übernimmt jedoch ein Interceptor diese Aufgabe. Dadurch werden Dienstzugriffe zwar nicht verheimlicht. Stattdessen wird die Zuordnung eines Dienstzugriffs zu einem konkreten Benutzer durch den Einsatz anonymer Credentials unterbunden. Auch dadurch bleiben die Personen anonym und ihre Zugriffe unverkettbar.

Schließlich sei bemerkt, dass MIXe nur die Kommunikationsbeziehungen schützen können aber nicht die Benutzerdaten auf Anwendungsebene.

9.3.3 Sicherheit und Privatsphäre in intelligenten Umgebungen

In diesem Abschnitt werden einige Projekte beschrieben, die Ansätze zur Gewährleistung der Privatsphäre und Sicherheit in intelligenten Umgebungen umgesetzt haben.

9.3.3.1 UbiSeC Lab

Im *UbiSeC Lab* an der Illinois Institute of Technology (IIT) werden u. a. Ansätze zur Gewährleistung der Sicherheit und Privatsphäre in intelligenten Umgebungen entwickelt. In [RL07] wird ein System vorgestellt, das den Widerspruch zwischen Privatsphäre und Sicherheit auf Basis kryptografischer Verfahren wie blinde Signaturen aufweichen soll. Besonderer Wert wird dabei auf die Unterstützung mobiler Benutzer gelegt, die sich zwischen mehreren Umgebungen bewegen können.

Der Aufbau des Ansatzes ist in Abbildung 9.12 grob dargestellt. Er unterscheidet generell zwischen Benutzern, Diensten und einem Authentifizierungsserver. Der Authentifizierungs-

server, auch als *Service Provider* bezeichnet, ist sowohl für die Verwaltung der Dienste einer Umgebung als auch ihrer Zugriffsrechte zuständig. Zusätzlich kann ein sogenannter *Broker* berücksichtigt werden, der zu mehr Flexibilität führt.

Der Ansatz unterscheidet zwei Phasen der Dienstnutzung. In der ersten Phase wird der Benutzer *autorisiert*, um die Dienste in der Umgebung nutzen zu können. Hierfür registriert bzw. authentifiziert er sich beim Service Provider und erhält anschließend ein oder mehrere Credentials in Form einer Credential-Kette. Jedes Credential kann nur einmal benutzt werden und ist anschließend ungültig. Für jeden Diensttyp, beispielsweise für den dargestellten Druckerdienst, erhält der Benutzer eine eigene Credential-Kette.

In der zweiten Phase findet die *Dienstnutzung* statt. Mithilfe der zuvor erhaltenen Credentials kann der Benutzer auf die Dienste in der Umgebung zugreifen, ohne dabei seine wahre Identität offenzulegen. Findet nun ein Zugriff auf einen Dienst statt, nimmt der Dienst das Credential entgegen und übergibt es an den Service Provider, der die Gültigkeit des Credentials überprüft. Nach einer positiven Antwort führt der Dienst die gewünschte Funktionalität aus und das Credential ist verbraucht.

Um den Missbrauch von (möglicherweise gestohlenen) Credentials zu vermeiden, schlagen die Autoren vor, zusätzlich eine vertrauenswürdige Instanz (TTP) einzuführen. In Zweifelsfall kann das TTP überprüfen, ob der Benutzer der rechtmäßige Besitzer eines vorgezeigten Credentials ist.

Vergleich Der UbiSeC-Ansatz verfolgt die gleichen Ziele wie in der vorliegenden Arbeit. Ähnlich zu den hier vorgestellten Session-Credentials werden auch im UbiSeC Credentials für die Zugriffskontrolle eingesetzt. An folgenden Punkten unterscheiden sich beide Ansätze. Zunächst sind die zugrunde liegenden Architekturmodelle unterschiedlich. Im UbiSeC wird angenommen, dass Dienste in intelligenten Umgebungen verteilt und selbstständig ausgeführt werden. Daher müssen die Dienste im UbiSeC aktiv an der Durchführung der Zugriffskontrolle teilnehmen. In der vorliegenden Arbeit werden jedoch alle Dienste auf dem eHome-Gateway ausgeführt und der Interceptor führt die Zugriffskontrolle durch (s. Abschnitt 7.3). Ferner wird in der vorliegenden Arbeit explizit die Verwendung von TTP-Credentials vorgeschlagen, um die Verkettbarkeit von Benutzersitzungen zwischen mehreren eHomes zu vermeiden. In UbiSeC hingegen wird nicht konkret beschrieben, wie die Authentifizierung des Benutzers durchgeführt werden soll. Außerdem müssen einem Benutzer in UbiSec mehrere Credentials ausgestellt werden, falls er auch mehrere Dienste zugreifen möchte. Der hier verfolgte Ansatz hingegen erlaubt den anonymen Zugriff auf mehrere Dienste mit nur einem einzigen Session-Credential. Darüber hinaus können im UbiSeC-Ansatz zusammenarbeitende TTPs und Service Provider die Anonymität von Benutzern aufheben. Dies kann im vorliegenden Ansatz durch die Nutzung von idemix-Credentials nicht passieren. Schließlich machen die Autoren des UbiSeC-Ansatzes keine Angaben über den Schutz anfallender Benutzerdaten oder den Schutz von Diensten vor unbefugten Zugriffen anderer Dienste.

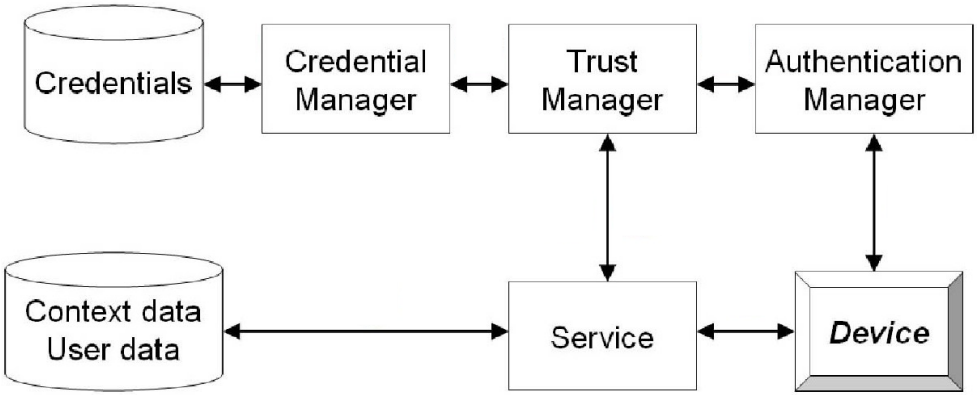


Abbildung 9.13: Middleware for Secure Home Access and Control am C-LAB in Paderborn (Quelle: [MMS+07]).

9.3.3.2 C-LAB

Der folgende Ansatz beschreibt die Architektur einer auf OSGi aufsetzenden Plattform für eHomes, die einen sicheren Umgang mit Benutzerprofilen gewährleisten soll [MMS+07, SZM06]. Dieser Ansatz wurde am C-LAB in Paderborn, einem gemeinsamen Forschungs- und Entwicklungslabor der Siemens AG und der Universität Paderborn, entwickelt.

Die Architektur der Plattform ist in Abbildung 9.13 dargestellt, wobei alle Komponenten als OSGi-Dienste realisiert sind. Anwendungen stellen Benutzern Dienstfunktionalitäten zur Verfügung und können auf einem PC oder auf einem Handheld ausgeführt werden. Zu ihren Funktionalitäten gehören etwa das Ändern von persönlichen Daten oder das Ausführen von Geldtransaktionen. Aufgrund des sensiblen Charakters der Funktionalitäten und der dabei verwendeten Daten soll der Zugriff auf Daten und Dienste geschützt werden.

Möchte eine Anwendung auf Dienste oder Daten zugreifen, muss sie sich zuvor autorisieren lassen, um ein passendes Credential zu erhalten. Durch die Autorisierung werden die Rechte des Benutzers, der die Anwendung nutzt, auf die Anwendung übertragen. Hierfür wendet sich die Anwendung an den Trust Manager. Dieser stellt selbst keine Credentials aus, sondern fragt sie beim Credential Manager an und signiert sie bei Bedarf.

Eine Besonderheit dieses Ansatzes liegt in der Authentifizierung, die darüber entscheidet, welche Sicherheitsstufe einem Benutzer zugewiesen wird. Die unterschiedlichen Sicherheitsstufen sind mit verschiedenen Rechten verbunden, die sich aus der Art der Anmeldung ableiten. Die Authentifizierungsmechanismen sind unterschiedlich gewichtet. Beispielsweise bekommt eine Anmeldung mit Passwort eine Gewichtung von 25 Punkten. Eine Authentifizierung über einen Fingerabdruck ist wesentlich sicherer und ist mit 80 Punkten gewichtet. Anhand dieser Punkte wird zur Laufzeit entschieden, ob der Benutzer auf einen Dienst zugreifen darf. Beispielsweise werden für das Onlinebanking mindestens 75 Punkte verlangt. Eine Authentifizierung per Fingerabdruck genügt hierfür, eine Passworteingabe hingegen nicht. Um die Sicherheitsstufe eines Benutzers zu ermitteln, kooperiert der Trust Manager mit dem Authentication Manager. Letzterer teilt dem Trust Manager den Benutzernamen und seine Sicherheitsstufe mit.

Mit diesen Daten kann der Credenital Manager ein Credential in Form einer Zugriffskontrollliste ausstellen. Das Credential besteht aus:

- einem **Anwendungsbezeichner** zur Protokollierung von Zugriffen,
- einem **öffentlichen Schlüssel der Anwendung** zur Identifizierung der Anwendung,
- einem **öffentlichen Schlüssel des Ausstellers**, damit Geräte oder der Profilmanager erkennen können, wer das Credential ausgestellt hat,
- einer **Zugriffskontrollliste**, die Zugriffsrechte des zugehörigen Benutzers auf Profildaten enthält und
- einem **Zeitstempel**, um die Gültigkeitsdauer des Credentials festzulegen.

Der Trust Manager signiert das Credential, wenn der Benutzer und der Aussteller vertrauenswürdig sind. Dann wird das signierte Credential an die Anwendung weitergeleitet, die damit die benötigten Dienste und Daten verwenden kann.

Vergleich Der zuvor vorgestellte Ansatz legt einen besonderen Wert auf die Zuweisung von Sicherheitsstufen zu Authentifizierungsmechanismen, auf deren Grundlage die Zugriffskontrolle für Daten und Dienste realisiert wird. Im Gegensatz zur vorliegenden Arbeit wird keine Möglichkeit zur anonymen Authentifizierung angeboten. Ferner werden die Credentials in der vorliegenden Arbeit nicht für Anwendungen ausgestellt, sondern für die Benutzer. Gemeinsam ist den Credentials in beiden Ansätzen jedoch, dass sie Zugriffskontrolllisten enthalten. Zwar werden auch in der vorliegenden Arbeit mehrere alternative Authentifizierungsmechanismen angeboten. Im Gegensatz zum C-LAB-Ansatz sind jedoch nicht mit Sicherheitsstufen assoziiert, sondern mit Anonymitätsstufen, wodurch der Fokus auf den Schutz der Privatsphäre gelegt wird.

9.3.3.3 Gaia

Privatsphäre und Sicherheit werden auch in dem in Abschnitt 9.1.1 beschriebenen Gaia-Projekt behandelt [CAMN+03]. Dabei ist der Fokus auf Authentifizierungsmechanismen und auf die Zugriffskontrolle gerichtet. Die Authentifizierung kann in Gaia über verschiedene Mechanismen umgesetzt werden. Die Verwendung von Passwörtern ist genauso möglich wie die von biometrischen Merkmalen. Für die Authentifizierung können auch externe Verfahren wie etwa *Kerberos* [NT94] eingebunden werden.

Weil Gaia keine zentrale Steuereinheit wie das eHome-Gateway vorsieht, agieren die Geräte dort autonom. Es muss daher sichergestellt werden, dass kein Gerät sich als ein anderes ausgeben kann. Daher müssen sich in Gaia nicht nur Personen authentifizieren, sondern auch Geräte.

Um Zugriffe auf Dienste zu kontrollieren, wurde in Gaia ein rollenbasierter Ansatz verfolgt. Es existieren drei Arten von Rollen: system, application und space roles. Die system role beinhaltet Rechte für Ressourcen innerhalb eines Systems. Ein System ist

```
1 {
2 USERID = (bsgill)
3 ROLES = (student) (ACM member)
4 ATTRIBUTES = (Age = 23) (Field of Study = Computer Science)
5 }signed by AC
```

Listing 9.1: Ein Gaia-Credential (aus [CGRZ04])

dabei beispielsweise eine Universität oder Firma. Application roles werden benutzt, um die Zugriffsrechte für Anwendungen festzulegen. Beide Arten von Rollen werden dann zusammen in eine space role transformiert. Eine solche Rolle legt fest, welche Rechte eine Person in einem Raum besitzt. Die hängen davon ab, welche Rechte diese Person generell besitzt (system role) und welcher Tätigkeit sie gerade nachgeht (application role).

Nachdem Personen oder Geräte authentifiziert wurden, erhalten sie ein Credential. In diesem Credential werden die Rollen und verschiedene Attribute durch den Aussteller zertifiziert. Ein solches Credential ist in Listing 9.1 dargestellt. Mit den Credentials können während der Zugriffskontrolle Rollen und Attribute nachgewiesen werden.

Vergleich In Gaia steht die Zugriffskontrolle auf Dienste und Geräte im Mittelpunkt, während der Schutz der Privatsphäre einen wichtigen Teil der vorliegenden Arbeit ausmacht. In Gaia wird der Schutz der Privatsphäre hauptsächlich mit der Geheimhaltung des Aufenthaltsorts eines Benutzers gleichgesetzt. Gaia sieht jedoch keine Möglichkeit zur anonymen Authentifizierung vor. Daher sind die in Gaia ausgestellten Credentials im Gegensatz zur vorliegenden Arbeit nicht anonym. Ein weiterer Unterschied beider Ansätze liegt darin, dass in Gaia nur die rollenbasierte Zugriffskontrolle umgesetzt wurde. Im Rahmen dieser Arbeit kann die Zugriffskontrolle dagegen auch auf Basis von Session-Credentials durchgeführt werden. Ferner müssen in Gaia neben den Benutzern auch die Geräte authentifiziert werden, weil sie eigene Dienste anbieten können. Im Rahmen der vorliegenden Arbeit gilt dies nur für die Handhelds, übrige Dienste werden vom eHome-Gateway aus gesteuert und müssen nicht authentifiziert werden.

9.3.4 Kooperative Schutzmechanismen

Kooperative Schutzmechanismen basieren auch auf Datenschutzrichtlinien. Zusätzlich zum Vorweisen einer solchen Richtlinie nimmt der Dienstanbieter jedoch auch eine aktive Rolle beim Schutz der Daten und der Nachvollziehbarkeit ihrer Verarbeitung ein. Hierbei kommt meist das Konzept eines *Zugriffskontrollmonitors* zum Einsatz, der den Zugriff regelt und überwacht. Das Vertrauen zum Dienstanbieter wird durch ein Zugriffskontrollsystem ergänzt. Innerhalb der Dienstanwendung ist das korrekte Verhalten des Zugriffskontrollsystems nachzuweisen.

Grundlage vieler solcher Mechanismen ist die Arbeit von Korba und Kenny, die ein Konzept vorgestellt haben, welches von einem abstrakten *Digital Rights Management System (DRM)* ausgeht [FFSS01]. Sie schlagen eine Variante vor, die sie als *Privacy Rights*

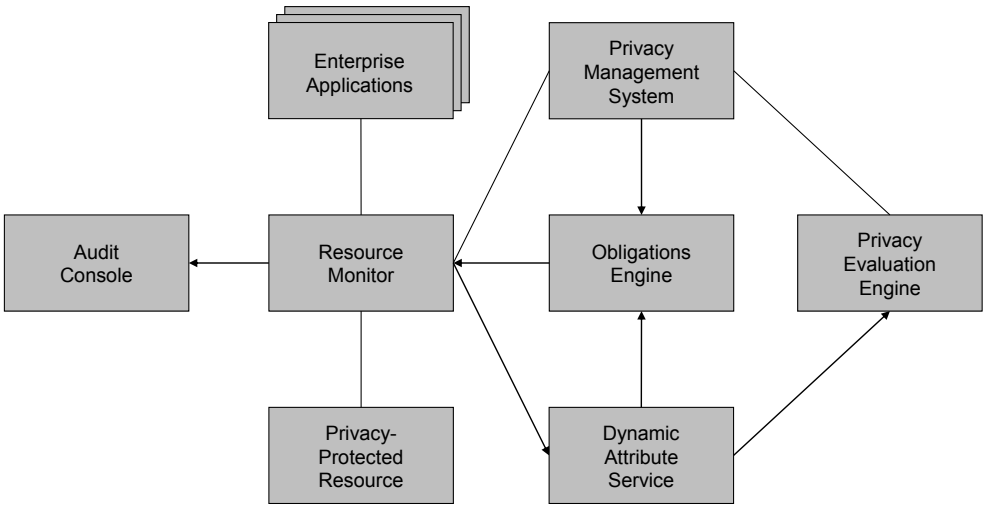


Abbildung 9.14: Platform for Enterprise Privacy Practices (Quelle: [KSW03]).

Management (PRM) bezeichnen. Zur Formulierung einer Richtlinie verwenden die Autoren die *Open Digital Rights Language (ODRL)* und schlagen als technische Abbildung von PRM ein System zur digitalen Rechteverwaltung vor. Beide machen jedoch keine Angaben dazu, wie ein solches System konkret aussehen soll. Eine auf diesen Arbeiten basierende Entwicklung ist z. B. das *Windows Rights Management*.

Nachfolgend wird stellvertretend für diese Art von Mechanismen die vom Züricher IBM Forschungslabor entwickelte Plattform E-P3P vorgestellt.

9.3.4.1 Platform for Enterprise Privacy Practices (E-P3P)

Die *Platform for Enterprise Privacy Practices*, auch als *Enterprise P3P (E-P3P)* bezeichnet, erweitert den oben beschriebenen P3P-Ansatz [KSW03]. E-P3P unterstützt Unternehmen darin, veröffentlichte Datenschutzrichtlinien innerhalb des Unternehmens durchzusetzen. Der Schwerpunkt liegt bei der Formulierung und Umsetzung von Richtlinien in Form von Regeln, die den Umgang mit Benutzerdaten vorgeben. Wenn innerhalb des Unternehmens auf eine bestimmte Kategorie von Benutzerdaten zugegriffen werden soll, wird anhand der Regeln entschieden, ob der Zugriff für eine bestimmte Benutzergruppe erlaubt oder verweigert wird. Mit solchen Regeln können detaillierte Zugriffsbedingungen beschrieben werden. Daneben sind noch Verpflichtungen enthalten, die mit dem Zugriff auf die Daten verbunden sind, z. B. muss eine automatische Benachrichtigung erfolgen, wenn Daten an Dritte weitergegeben werden.

Abbildung 9.14 zeigt eine Übersicht der Architektur von E-P3P. Zur Überwachung der Einhaltung der Richtlinien dient der sogenannte Resource Monitor, der sich zwischen den Anwendungen und den Daten befindet. Der Monitor, über den alle Datenzugriffe laufen, fragt bei jedem Zugriff das Privacy Management System, ob der Zugriff erlaubt oder verweigert werden soll. Letzteres wiederum leitet die Anfrage weiter an die Privacy Evaluation Engine. Diese entscheidet nun, ob ein Zugriff erlaubt oder verweigert wird. Zudem wird

überprüft, ob für diesen Zugriff bestimmte Verpflichtungen einzuhalten sind, die über die *Obligation Engine* abgewickelt werden. Falls z. B. eine Autorisierung stattfinden soll, führt der *Resource Monitor* die Anfrage durch und protokolliert dies in der *Audit Console*.

Dieser Ansatz hat seinen Vorteil darin, dass Dienstanbieter bei der unternehmensweiten Durchsetzung von Datenschutzrichtlinien unterstützt werden. Es funktioniert jedoch nur dann, wenn der Anbieter sich an seine Richtlinien halten möchte. Fall es sich jedoch um einen bösartigen Anbieter handelt, kann es nicht gewährleisten, dass die Richtlinien tatsächlich eingehalten werden. Aus der Sicht des Benutzers weist es daher ähnliche Probleme auf wie der P3P-Ansatz.

9.3.5 Weitere Ansätze

In [BDP07] wird *DAIDALOS*, ein Ansatz zum Schutz der Privatsphäre im Ubiquitous Computing, vorgestellt. In diesem Ansatz spielen drei Komponenten zusammen, um zuverlässig die Privatsphäre von Benutzern zu schützen: der Trust Manager berechnet die Vertrauenswürdigkeit von Kommunikationspartnern, der Identity Manager erstellt und verwaltet digitale Identitäten von Benutzern und der Privacy Negotiation Manager handelt Vereinbarungen zur Privatsphäre zwischen den Kommunikationspartnern aus. Die Autoren betonen, dass der Schutz der Privatsphäre nicht nur während, sondern auch vor und nach einer Kommunikationsbeziehung stattfindet. Daher haben sie ein Drei-Phasen-Modell aufgestellt: In der Phase *Pre-Communication* wird die Vertrauenswürdigkeit bestimmt, die Vereinbarung zur Privatsphäre ausgehandelt und die digitale Identität ausgewählt. In der Phase *Communication* findet die Kommunikation statt und die Freigabe von Daten wird überwacht. In der Phase *Post-Communication* wird die Vertrauenswürdigkeit der Partner aktualisiert. Die dritte Phase hält bis zur nächsten Kommunikation der beiden Partner an. Bei einer erneuten Kommunikation werden die Phasen erneut durchlaufen.

In [Sou08] wird ein Überblick über bestehende Ansätze zur Authentifizierung von Benutzern gegeben. Ferner werden diese nach unterschiedlichen Authentifizierungsproblemen klassifiziert. In der ersten Klasse greift ein Benutzer auf einen Dienst zu. Dafür wird der Benutzer identifiziert und kann dadurch auf Ressourcen zugreifen. Meist wird hier ein *Ticket*-System wie Kerberos [NT94] eingesetzt. In der zweiten Klasse greift eine Gruppe von Benutzern auf Dienste zu. In dieser Klasse wollen Benutzer anonym bleiben, gleichzeitig aber ihre Zugriffsrechte nachweisen. Der Autor empfiehlt hier den Einsatz anonymer Credentials, wie sie auch im Rahmen der vorliegenden Arbeit eingesetzt wurden. In der dritten Klasse werden sichere Verbindungen zwischen Kommunikationspartnern betrachtet. Hierzu diskutiert der Autor eine Reihe von Lösungsansätzen, gibt aber keine Empfehlung. Er kritisiert abschließend den Mangel eines praktisch umgesetzten Systems, das mehrere dieser Klassen gleichzeitig abdeckt.

In [CI07] wird ein Überblick über verschiedene Ansätze zum Schutz der Privatsphäre gegeben. Dabei werden aus der Literatur bekannte Ansätze klassifiziert. Es werden vier Klassen beschrieben: (1) Der *Inhalt von Dienstzugriffen* wird meist durch Verschlüsselung oder über sichere Kommunikationskanäle geschützt. (2) Die *Existenz von Dienstzugriffen* wird durch versteckte Kanäle oder MIXe verborgen. (3) Protokolle wie P3P gewährleisten

die *Datensparsamkeit*. (4) Die *Weiterverarbeitung von Daten* kann durch Privacy Rights Management überwacht werden.

In [SSA06] wird argumentiert, dass das Sammeln und Speichern von Daten unvermeidbar ist, wenn eine intelligente Umgebung personalisiert werden soll. Dabei würden die Daten ohne die Kenntnis der Benutzer gesammelt und dauerhaft gespeichert. Daraus ergeben sich Risiken für die Privatsphäre. Die Autoren behaupten, dass durch den Einsatz von Pseudonymen und digitalen Identitäten sowie Verträgen über die Handhabung persönlicher Daten die Privatsphäre nicht geschützt werden kann. Nur wenn es eine Möglichkeit gäbe, die Durchsetzung der Verträge zu kontrollieren und zu erzwingen, könnte Privatsphäre in intelligenten Umgebungen geschützt werden. Dazu schlagen sie einen Ansatz vor, der auf sicheren Log-Daten beruht. Jeder Umgang mit persönlichen Daten wird auf eine sichere Art und Weise erfasst. Mit anderen Worten ist die Integrität sichergestellt und die Daten können nicht von unautorisierten Personen eingesehen werden. Zusammen mit NAPS [RS06], einem P3P-ähnlichem Protokoll, werden aus den Log-Daten sogenannte *Privacy Evidences* erzeugt. Sie protokollieren, wie persönlichen Daten tatsächlich verwendet wurden. Durch einen solchen Nachweis sind sowohl Personen als auch Dienstleister abgesichert, da sie alle Handlungen eindeutig belegen können.

In [LM07] wird das *Ninja*-Authentifizierungsmodell für intelligente Umgebungen mit mobilen Benutzern vorgestellt. Es ermöglicht die beidseitige Authentifizierung zwischen mobilen Benutzern und Dienstanbietern während der Suche nach Diensten. Durch den Einsatz einer *Trusted Computing Platform* [Pea02] wird die Privatsphäre der Benutzer bewahrt. Dieses Modell wurde zwar theoretisch beschrieben, aber nicht implementiert. Es beschreibt eine anonyme Authentifizierung bei einem Dienstanbieter. Es wird aber nicht darauf eingegangen, welche Dienste anschließend von den Benutzern genutzt werden dürfen und wie dabei die Zugriffskontrolle umgesetzt wird.

In [KLK+06, LKD+07] wird die Middleware *DISCREET* vorgestellt, die den Schutz der Privatsphäre in einer intelligenten Stadt gewährleisten soll. Für die Verwaltung von Benutzerdaten wird ein *geschützter Bereich* eingesetzt, der von einer vertrauenswürdigen Organisation administriert wird. Möchte eine Anwendung nun auf bestimmte Daten zugreifen, gibt die Middleware die Daten nicht raus. Stattdessen wird die Anwendungslogik in den geschützten Bereich migriert und dort ausgeführt. Dadurch kann unbefugter Zugriff auf Benutzerdaten vermieden werden. Ein Identitätsmanagementsystem unterstützt die Benutzer darin, die Menge der offenzulegenden Daten zu bestimmen. Anonymität wird auf der Netzwerkebene durch MIXe realisiert.

In [KS03] werden Benutzermodelle in Bezug auf Sicherheit und Privatsphäre untersucht. Dabei wird die Sicherheit als Voraussetzung für die Anwendung von Richtlinien zum Schutz der Privatsphäre gesehen. Die Benutzer werden in die Definition von Richtlinien einbezogen, sodass ein Benutzermodell die Sicherheitskriterien an die Bedürfnisse seiner Benutzer anpassen kann. Die selektive Zugriffskontrolle wird auf Basis des rollenbasierten Zugriffskontrollmodells realisiert. Hierfür wurde die *Knowledge Query and Manipulation Language* (KQML) um die MIX-Technik erweitert (*SKQML*), um den Nachrichtenaustausch zwischen Anwendungen, Benutzermodellen und Benutzern auf Basis von Pseudonymen zu anonymisieren. Der Nachrichtenaustausch wird dabei per SSL verschlüsselt.

In [Sta02] wird die *Resurrecting Duckling Security Policy* als Lösungsansatz für die Authentifizierung in Systemen, in denen kein dauerhaft verbundenes TTP existiert, vorgestellt. Die Idee hinter diesem Ansatz sieht die Kopplung von Geräten in intelligenten Umgebungen vor. Soll ein Gerät zum ersten Mal in einer Umgebung verwendet werden (*Geburt*), wird es mit einem sogenannten Mutterobjekt verknüpft. Durch diese Master-Slave-Beziehung wird sichergestellt, dass das neue Gerät nur mit dem Mutterobjekt interagiert. Beispielsweise könnte dieser Ansatz für Autoschlüssel angewendet werden. Ein anderes Beispiel bildet die Fernbedienung eines eHomes, der alle Geräte in dem eHome kontrollieren kann, aber nicht die der Nachbarn. Hierfür wird auch der Begriff *Secure Transient Association* verwendet. Der Fokus dieses Ansatzes ist somit auf die Authentifizierung von Geräten untereinander gerichtet, während in der vorliegenden Arbeit auch die Authentifizierung von Benutzern betrachtet wurde.

In [MJM01b] werden allgemeine Aufgaben und Möglichkeiten von Identitätsmanagementsystemen bzgl. der Nutzung im Internet beschrieben. Auf Basis der dort identifizierten Kriterien wurde ein kontextbasiertes und situationsbezogenes Identitätsmanagementsystem für mobile Geräte für den Einsatz in intelligenten Umgebungen entwickelt, der in [JKZ02] vorgestellt wird.

Kapitel 10

Schlussbemerkungen

Diese Arbeit ist im Anschluss eines Forschungsprojekts entstanden, das sich die Entwicklung von Konzepten und Werkzeugen zum Ziel gesetzt hat, die die kostengünstige softwareseitige Einrichtung und den Betrieb von eHomes auf Basis des SCD-Prozesses ermöglichen. Ausgehend von den Ergebnissen dieses Projekts wurden zwei Fragestellungen betrachtet, die bisher nicht behandelt wurden, aber eine wesentliche Rolle für eine weitverbreitete Akzeptanz von eHomes spielen.

Die erste Fragestellung betrifft die Unterstützung mobiler Benutzer. Hierunter werden Benutzer verstanden, die häufig zwischen verschiedenen eHomes wechseln (Inter-eHome-Mobilität). Sie möchten dabei, dass ihre persönlichen Präferenzen in allen eHomes ohne mehrfachen Konfigurierungsaufwand soweit wie möglich umgesetzt werden.

Die zweite Fragestellung bezieht sich auf die Gewährleistung der wechselseitigen Sicherheit im Rahmen dieser Mobilität. Konkret wurden Ansätze untersucht, die den Schutz der Privatsphäre auf der einen Seite und den Schutz von eHome-Diensten vor Zugriff unbefugter Benutzer oder Dienste auf der anderen Seite sicherstellen. Eine besondere Schwierigkeit hierbei lag darin, die einander widerstrebenden Anforderungen „Schutz der Privatsphäre“ und „Komfort durch automatische Personalisierung“ miteinander in Einklang zu bringen.

In diesem Kapitel werden die im Rahmen dieser Arbeit entwickelten Konzepte zusammenfassend wiedergegeben. Anschließend werden mögliche Ansatzpunkte für weiterführende Forschungsarbeiten beschrieben.

10.1 Zusammenfassung

Sowohl für die Personalisierung als auch beim Schutz der Privatsphäre in eHomes spielen Benutzerdaten eine zentrale Rolle. Die Architektur des existierenden eHome-Prototyps aus dem Vorgängerprojekt wies einige Probleme in Bezug auf die Verwaltung dieser Daten auf. Um diese Probleme zu beheben, wurde im Rahmen dieser Arbeit die Architektur so umstrukturiert, dass die Benutzerdaten vom Rest des Systems entkoppelt wurden. Das neu entwickelte *Benutzermodell* ist nun allein für die Verwaltung von Benutzerdaten zuständig.

Das Benutzermodell stellt nun sicher, dass in eHomes keine Redundanzen von Benutzerdaten auftreten können. Durch die lose Kopplung mit Diensten und den übrigen

Bausteinen des Prototyps konnten Qualitätseigenschaften, wie etwa die Wartbarkeit, von eHomes verbessert werden. Darüber hinaus kann das Benutzermodell jetzt auch Benutzer handhaben, die ein eHome erst nach dessen Inbetriebnahme betreten. Bisher mussten die Benutzer dem eHome hingegen schon vor der Inbetriebnahme des eHomes bekannt sein, damit ihre Präferenzen berücksichtigt werden konnten. Des Weiteren unterstützt das Benutzermodell mehrere Typen personalisierbarer Dienste, wie beispielsweise personen- oder auch raumgebundene Dienste, die im selben eHome ausgeführt werden können.

Anschließend wurde in dieser Arbeit untersucht, wie Benutzer darin unterstützt werden können, Dienste auch über ihre eigenen vier Wände hinaus in anderen eHomes zu nutzen. Insbesondere wurde in diesem Zusammenhang die *Personalisierung* von eHome-Diensten betrachtet. Es stellte sich heraus, dass in jedem eHome Benutzerdaten anfallen, wenn in diesem personalisierbare Dienste genutzt werden sollen. Der existierende Prototyp unterstützte die Benutzer nicht darin, ihre Daten mehreren eHomes zur Verfügung zu stellen. Daher mussten Benutzerdaten in jedem eHome redundant gespeichert werden.

Für die Lösung dieses Problem wurde im Rahmen dieser Arbeit ein *mobiles Benutzermodell* entwickelt, das die Benutzer auf ihren Handhelds stets bei sich tragen und ihre Daten unterwegs freigeben können. Die Modellierung der Benutzerdaten auf Grundlage der Benutzermodellontologie GUMO stellt dabei die Interoperabilität zwischen den Handhelds und den eHomes sicher. Ferner wurde ein Synchronisationsmechanismus entwickelt, der dafür sorgt, dass sowohl auf den Handhelds als auch in den eHomes stets die aktuellen Daten verwendet werden. Insgesamt wurde dadurch der Aufwand zur Bereitstellung von Benutzerdaten für mehrere eHomes reduziert.

Das Handheld wurde außerdem als Grundlage für eine neue Form der Personalisierung genutzt. Es wurde ein Konzept entwickelt, das die *Ausführung persönlicher Dienste auf dem Handheld* ermöglicht, die mit passenden Unterdiensten im eHome kommunizieren können. Auf diese Weise sind mobile Benutzer nun imstande, gewohnte Funktionalitäten auch dann zu nutzen, wenn das besuchte eHome selbst keinen Dienst mit ähnlichen Funktionalitäten anbietet. Hierfür wurde der SCD-Prozess um Verteilungsaspekte erweitert. Zusätzlich wurde ein Konzept entwickelt, der den Benutzern erlaubt, über ihr Handheld mit eHome-Diensten zu interagieren. Hierfür erhält jeder Dienst eine für seine Funktionalitäten zugeschnittene Benutzeroberfläche, die auf dem Handheld dargestellt wird. Dadurch kann das Handheld als eine Art zentrales Bedienelement für eHomes eingesetzt werden.

Als Nächstes wurde die Fragestellung untersucht, wie die *Privatsphäre* der Benutzer geschützt werden kann. Es wurde insbesondere das Ziel verfolgt, die einander widerstrebenden Anforderungen *informationelle Selbstbestimmung* und *Komfort durch automatische Personalisierung* miteinander in Einklang zu bringen. Um dieses Ziel zu erreichen, wurden im Rahmen dieser Arbeit auf zwei Ebenen Konzepte zur Datensparsamkeit und Unverkettbarkeit offengelegter Daten entwickelt.

Auf der ersten Ebene, der *eHome-Ebene*, wurden persönliche Daten betrachtet, die einem eHome mindestens offengelegt werden müssen, um in diesem personalisierbare Dienste verwenden. Dabei wurde festgestellt, dass von eHome zu eHome unterschiedliche Daten erforderlich sind. Aus diesem Grund wurde ein Ansatz zum *selbstgesteuerten Identitätsmanagement* realisiert, der mobilen Benutzern ermöglicht, jedem eHome gegenüber mit einer

anderen Identität entgegenzutreten. Dadurch sind die Benutzer nun in der Lage, bestimmen zu können, wann welches eHome welche Daten erhalten darf.

Die von einem eHome benötigten Daten hängen dabei von den Diensten ab, die ein Benutzer in diesem nutzt. Daher wurde das Identitätsmanagement um einen *Aushandlungsprozess* erweitert. Dieser unterstützt die Benutzer darin, mit dem eHome flexibel die Menge der zu nutzenden Dienste und damit verbunden auch die Menge der preiszugebenden Daten auszuhandeln. Dadurch behält der Benutzer die Kontrolle über seine Daten und kann erwirken, dass die Menge der freigegebenen Daten auf ein notwendiges Minimum reduziert wird. Zusätzlich trägt auch die Ausführung von Diensten auf dem Handheld der Reduzierung offengelegter Daten bei, weil die von ihnen benötigten Daten auf dem Handheld verbleiben. Die entwickelten Konzepte wurden insgesamt unter dem Begriff der *Datensparsamkeit* zusammengefasst.

Zusätzlich zur Datensparsamkeit wurde auf Basis anonymer Credentials ein Authentifizierungsmechanismus entwickelt, der die *Verkettbarkeit* von Benutzerdaten über mehrere Sitzungen, also Interaktionen zwischen einem Benutzer und verschiedenen eHomes, hinweg unterbindet. Dies ist wichtig, da durch die Kombination der preisgegebenen Benutzerdaten in verschiedenen eHomes möglicherweise Informationen abgeleitet werden können, die der jeweilige Benutzer jedoch schützen möchte.

Auf der zweiten Ebene, der *Dienstebene*, wurden persönliche Daten betrachtet, die einem eHome-Dienst zur Verfügung gestellt werden müssen, um ihn zu personalisieren. Im Hinblick auf den Schutz persönlicher Daten ist es sinnvoll, nicht allen Diensten eines eHomes die gleichen, ausgehandelten Daten vollumfänglich zu offerieren. Daher wurde im Rahmen dieser Arbeit das Ziel verfolgt, jedem Dienst nur die Daten zur Verfügung zu stellen, die er zur Realisierung seiner Funktionalitäten benötigt. Hierfür wurde das Benutzermodell um eine *selektive Zugriffskontrolle* erweitert. Der Benutzer kann bei Bedarf interaktiv mitbestimmen, welcher Dienst auf welche Daten zugreifen darf. Ferner vermeidet das Benutzermodell durch den Einsatz von *Pseudonymen* eine mögliche Verkettbarkeit der Daten durch die Dienste untereinander. Insgesamt übernimmt das Benutzermodell somit über die eigentliche Datenhaltung hinaus auch eine aktive Rolle beim Schutz der Privatsphäre.

Während die Wahrung der Privatsphäre die Schutzziele von Benutzern abdeckt, wurden im Rahmen dieser Arbeit auch die Schutzziele von eHomes betrachtet. In diesem Zusammenhang wurde festgestellt, dass eHome-Dienste vor unbefugtem Zugriff geschützt werden müssen. Dabei wurde unterschieden zwischen unbefugten Zugriffen durch Benutzer und unbefugten Zugriffen durch Dienste.

Für den Schutz von eHome-Diensten vor dem Zugriff unbefugter *Benutzer* wurden im Rahmen der vorliegenden Arbeit mehrere alternative Authentifizierungsmechanismen entwickelt, um die widersprüchlichen Anforderungen nach Sicherheit und Privatsphäre miteinander in Einklang zu bringen. Offensichtlich wird ein eHome einem Benutzer nur dann Zugriff auf seine Dienste gewähren, wenn sich dieser authentifizieren lässt, also gewisse Eigenschaften von sich preis gibt. Umgekehrt möchte der Benutzer jedoch möglichst wenig über sich offenbaren, um seine Privatsphäre zu schützen.

Diese Authentifizierungsmechanismen bieten den Benutzern jeweils unterschiedliche Anonymitätsstufen. Die stärkste Anonymität kann ein Benutzer erlangen, indem er sich

anhand anonymer Credentials (TTP-Credentials) bei einem eHome authentifiziert und als Zugriffsberechtigung für die ausgehandelten Dienste ein weiteres anonymes Credential erhält (Session-Credential). Den Besitz des Letzteren muss er bei jedem Zugriff auf einen Dienst nachweisen. Dadurch sind weder seine Sitzungen noch seine Zugriffe auf Dienste während einer Sitzung miteinander verkettbar. Diese zweistufige Anwendung anonymer Credentials ist ein besonderer Beitrag dieser Arbeit. Dadurch können Zugriffsrechte, falls gewünscht, für jede Sitzung flexibel neu bestimmt werden.

Falls die Anonymität innerhalb einer Sitzung nicht gewünscht ist, kann sich der Benutzer dafür entscheiden, kein Session-Credential zu erhalten. Stattdessen werden ihm ein Pseudonym und eine Rolle zugewiesen. Inspiriert durch das rollenbasierte Zugriffskontrollmodell enthält die dem Benutzer zugewiesene Rolle Zugriffsrechte genau für die Dienste, die der Benutzer zuvor mit dem eHome ausgehandelt hat. Schließlich kann sich ein Benutzer auch ganz ohne anonyme Credentials authentifizieren, beispielsweise anhand eines Pseudonyms (genauer: Beziehungspseudonym) in Form eines Benutzernamens und einem Passwort oder anhand eines biometrischen Merkmals. Die Zugriffskontrolle während der Sitzung findet wiederum rollenbasiert statt. Dieser Ansatz ist jedoch nicht so flexibel und sicher wie mit anonymen Credentials, weil in diesem Fall der Benutzer dem eHome zuvor bekannt sein muss und über mehrere Sitzungen hinweg wiedererkannt werden kann.

Neben dem unmittelbaren Zugriff durch unbefugte Benutzer müssen Dienste auch vor dem mittelbaren unbefugten Zugriff über weitere *Dienste* geschützt werden. Hierfür wurde auch eine rollenbasierte Zugriffskontrolle zwischen verschiedenen Diensten umgesetzt. Im Rahmen dieses Ansatzes wird jedem Dienst eine Rolle zugewiesen, die Zugriffsrechte nur auf die Dienste enthält, die dieser Dienst benutzen darf. Die dafür benötigten Informationen werden aus der Dienstkomposition gewonnen, die ein Ergebnis des oben erwähnten SCD-Prozesses ist. Die Generierung der Dienstrollen und die dynamische Anpassung der Zugriffsrechte an eine sich ändernde Dienstkomposition findet dabei völlig automatisch statt.

Eine im Rahmen dieser Arbeit entwickeltes Werkzeug für Handhelds unterstützt mobile Benutzer bei der Durchführung vielfältiger Aufgaben. Es bietet eine Benutzeroberfläche für Aufgaben wie die Verwaltung eigener Identitäten, die Authentifizierung gegenüber eHomes, die Auswahl von eHome-Diensten und die Interaktion mit ihnen. Ein weiteres im Rahmen dieser Arbeit entwickeltes Werkzeug, der eHomeAdministrator, bietet eHome-Betreibern hingegen eine Benutzeroberfläche für die Verwaltung von Benutzern sowie deren Identitäten und Zugriffsrechten.

Für die Evaluierung der entwickelten Konzepte und Werkzeuge wurden unterschiedliche Demonstratoren für eHomes herangezogen. Ferner wurden mehrere Testszenarien mithilfe dieser Demonstratoren ausgeführt. Zwei dieser Demonstratoren wurden im Rahmen dieser Arbeit entwickelt und sind vollständig in Software realisiert. Es handelt sich dabei um den eHomeSimulator mit einer 2D-Oberfläche und um den OpenSim/Second-Life-Demonstrator mit einer 3D-Oberfläche. Letzterer bietet dabei die meisten Funktionalitäten, und auch die nötige Flexibilität, anhand der insbesondere auch die Mobilität von Benutzern zwischen mehreren eHomes simuliert werden konnte. Als Handhelds wurden mehrere PDAs eingesetzt. Zusammengenommen konnte die Anwendbarkeit des Gesamtkonzepts gezeigt werden.

10.2 Ausblick

Im Folgenden wird ein Ausblick auf mögliche Erweiterungen der in dieser Arbeit vorgestellten Konzepte gegeben.

Falls mehrere Benutzer in einem eHome bzw. Raum die gleichen Dienste nutzen, können aufgrund gegensätzlicher Präferenzen *Konflikte* auftreten. In dieser Arbeit wurden solche Konflikte in den Diensten selbst erkannt und durch einfache Strategien gelöst. Sinnvoller ist es jedoch, die Konflikterkennung und -lösung als Querschnittsfunktionalitäten nicht in den Diensten selbst durchzuführen. Stattdessen sollten diese Maßnahmen in den Verantwortungsbereich des Benutzermodells gestellt werden, weil dort alle Daten verfügbar sind, die für die Konfliktlösung benötigt werden. Dies dient auch dem Schutz der Privatsphäre der Benutzer, weil auf diese Weise nur noch die Informationen herausgegeben werden, die das Ergebnis der Konfliktlösung sind. Ein noch stärkerer Schutz der Privatsphäre könnte dadurch erzielt werden, indem persönliche Daten zunächst auf den Handhelds verbleiben und ohne den Umweg über das Benutzermodell im eHome direkt an die Dienste übergeben werden. Auf diese Art könnte in einem Konfliktfall anonym ausgehandelt werden, wessen Präferenzen ausgewählt werden. Dieser Gedanke ist inspiriert durch Verfahren zur anonymen Durchführung von Auktionen, wie etwa in [CND09]. Es ist jedoch nicht absehbar, ob solch ein Ansatz tatsächlich für eHomes anwendbar ist. Es bedarf daher einer genauen Untersuchung der Anwendbarkeit.

Im Rahmen dieser Arbeit wurde angenommen, dass das Benutzermodell im eHome von einem vertrauenswürdigen Hersteller stammt. Daher können die Benutzer darauf vertrauen, dass das Benutzermodell ihre Daten nicht an unbefugte Dienste weitergibt oder anderweitig missbraucht. Garantieren kann man ihnen das jedoch nicht. Es besteht daher Bedarf an Konzepten, die den Benutzern ermöglichen, erfahren zu können, wie ihre Daten in eHomes tatsächlich verwendet werden. Die Verwendung von Richtlinien, wie beispielsweise von P3P vorgesehen, ist hierfür nicht ausreichend, selbst bei Zertifizierung durch eine dritte Stelle. In diesem Fall würde das eHome nur behaupten, dass er sich an die Regeln hält, überprüfbar ist dies jedoch nicht. Die Umsetzung von sogenannten Monitoringkonzepten könnte hier eine Abhilfe schaffen. Das bedeutet, die Benutzer werden darüber informiert, welche personenbezogenen Daten über sie gesammelt und wie diese Informationen verwendet werden.

Dieser Aspekt ist insbesondere auch für solche Daten wichtig, deren Sammlung ein Benutzer nicht verhindern kann. Der hier vorgestellte Ansatz ermöglicht den Benutzern zwar, die Kontrolle über die Freigabe ihrer persönlichen Daten zu behalten. Sie kann jedoch nicht verhindern, dass ein eHome über spezielle Sensoren oder Kameras zusätzliche Informationen sammelt, die nicht der Kontrolle des Benutzers unterliegen. Diese Problematik rein technisch zu lösen, dürfte schwierig bis unmöglich sein. Wie auch in [Roß07] gefordert, müssen technische Konzepte daher mit einer Anpassung des normativen Schutzprogramms des *Datenschutzrechts* kombiniert werden.

Es wurde erwähnt, dass ein mobiler Benutzer mit eHome-Diensten über spezifische *Benutzeroberflächen* auf seinem Handheld interagieren kann. Dabei wurde angenommen, dass es einen Standard gibt, an den sich die Dienstentwickler beim Entwurf dieser Oberflächen

halten. Dadurch kann sichergestellt werden, dass mobile Benutzer in unterschiedlichen eHomes die gleichen Benutzeroberflächen für die gleichen Funktionalitäten vorfinden. Alternativ könnten Benutzeroberflächen auch auf Basis eines solchen Standards durch die Generierung sogenannter Adapter an die Präferenzen der Benutzer angepasst werden. Somit müssten die Hersteller nicht die gleichen Oberflächen entwickeln, stattdessen bestimmt das Handheld des Benutzers anhand der Funktionalitäten der Dienste, wie die Oberfläche aussehen soll. In beiden Fällen würde die Akzeptanz von eHomes erhöht werden. Einen solchen Standard gibt es momentan jedoch noch nicht. Daher besteht der Bedarf an der Entwicklung eines solchen Standards, möglicherweise auf Basis der von Norbisrath vorgeschlagenen Funktionalitätshierarchie [Nor07], die auch für die Dienstkomposition herangezogen wird. Die Entwicklung solcher Benutzeroberflächen sollte mit psychologischen Untersuchungen kombiniert werden, um eine möglichst hohe Akzeptanzquote zu erreichen.

Schließlich würde eine weiterführende *Studie* der im Rahmen dieser Arbeit realisierten Konzepte und Werkzeuge mit realen Testpersonen eine weiterführende Evaluierung ermöglichen. Insbesondere ist es interessant herauszufinden, inwieweit die Benutzer sich mit der anonymen Authentifizierung und Dienstnutzung in eHomes anfreunden werden. Werden sie einen möglichen Komfortverlust zugunsten eines besseren Schutzes der Privatsphäre hinnehmen? Ferner könnte eine weiterführende Analyse durchgeführt werden, um herauszufinden, welche Dienste von den Menschen angenommen werden und welche eher nicht. Die Antwort auf diese Fragen wird die Informatik allein nicht geben können. Es wird sich nach einer breiten Verfügbarkeit von eHomes zeigen, welche Konzepte und Anwendungen sich im freien Markt durchsetzen werden.

10.3 Fazit

Im Rahmen dieser Arbeit wurden Konzepte und Werkzeuge entwickelt, die eHome-Betreiber und insbesondere mobile Benutzer im Rahmen der Inter-eHome-Mobilität unterstützen. Das verfolgte Hauptziel, den Schutz der Privatsphäre und die Personalisierung von eHomes miteinander in Einklang zu bringen, wurde erreicht. Die hierfür eingesetzten anonymen Credentials bieten eine angemessene Kombination von Anonymität und Zurechenbarkeit. Die Anwendbarkeit der erzielten Ergebnisse wurde anhand unterschiedlicher Testszenarien und Demonstratoren gezeigt.

Damit leistet diese Arbeit einen wichtigen Beitrag für die Akzeptanz von eHomes und ebnet den Weg für einen zukünftigen Massenmarkt. Dieser Weg ist jedoch noch lang. Bis dahin wird und muss sich auf der technischen Ebene noch viel tun. Gleichzeitig muss aber auch die Gesellschaft erkennen, dass die informationelle Selbstbestimmung ein wichtiges und schützenswertes Gut ist. Denn ein kompetenter und aufgeklärter Benutzer stellt den wirksamsten Schutz vor unerwünschten Eingriffen in seine Privatsphäre dar.

Ausgewählte Veröffentlichungen

[AE08] Armaç, Ibrahim ; Evers, Daniel: Client Side Personalization of Smart Environments. In: *SAM '08: Proceedings of the International 1st Workshop on Software Architectures and Mobility at ICSE 2008 (SAM 2008)*, ACM, 2008. – ISBN 978–1–60558–022–7, S. 57–59

[AK06] Armaç, Ibrahim ; Kirchhof, Michael: Process Support in eHome Systems: Empowering Providers to Handle a Future Mass Market. In: Latour, Thibaud (Hrsg.) ; Petit, Michael (Hrsg.): *Proceedings of the CAiSE'06 Workshops and Doctoral Consortium*, Press Universitares de Namur, 2006. – ISBN 978–2–87037–525–9, S. 999–1013

[AKM06] Armaç, Ibrahim ; Kirchhof, Michael ; Manolescu, Liviana: Modeling and Analysis of Functionality in eHome Systems: Dynamic Rule-based Conflict Detection. In: *ECBS '06: Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2546–6, S. 219–228

[AKR07] Armaç, Ibrahim ; Kirchhof, Michael ; Richterich, Claus: Frictionless Service Interaction in Protected Areas: Collaboration in eHomes. In: Pernici, Barbara (Hrsg.) ; Gulla, Jon A. (Hrsg.): *Proceedings of the CAiSE'07 Workshops and Doctoral Consortium, Vol. 2*, Tapir Academic Press, 2007. – ISBN 978–82–519–2246–3, S. 515–529

[APPR09] Armaç, Ibrahim ; Panchenko, Andriy ; Pettau, Marcel ; Retkowitz, Daniel: Privacy-Friendly Smart Environments. In: Al-Begain, Khalid (Hrsg.): *Third International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies (NGMAST 2009)*, IEEE Computer Society, 2009. – ISBN 978–0–7695–3786–3, S. 425–431

[AR07] Armaç, Ibrahim ; Retkowitz, Daniel: Simulation of Smart Environments. In: *Proceedings of the IEEE International Conference on Pervasive Services 2007 (ICPS'07)*, IEEE Press, 2007, S. 257–266

[AR08] Armaç, Ibrahim ; Rose, Daniel: Privacy-friendly user modelling for smart environments. In: *Mobiquitous '08: Proceedings of the 5th Annual International*

Conference on Mobile and Ubiquitous Systems, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. – ISBN 978–963–9799–27–1, S. 1–6

[MA09a] Mengi, Cem ; Armaç, Ibrahim: Ein Klassifikationsansatz zur Variabilitätsmodellierung in E/E-Entwicklungsprozessen. In: *Software Engineering 2009 Workshop: Produkt-Variabilität im gesamten Lebenszyklus*, Gesellschaft für Informatik (GI), 2009 (Lecture Notes in Informatic (LNI))

[MA09b] Mengi, Cem ; Armaç, Ibrahim: Functional Variant Modeling for Adaptable Functional Networks. In: Benavides, David (Hrsg.) ; Metzger, Andreas (Hrsg.) ; Eisenecker, Ulrich W. (Hrsg.): *Third International Workshop on Variability Modelling of Software-Intensive Systems* Bd. 29, Universität Duisburg-Essen, 2009 (ICB Research Report). – ISSN 1860–2770, S. 83–92

[NARS06] Norbisrath, Ulrich ; Armaç, Ibrahim ; Retkowitz, Daniel ; Salumaa, Priit: Modeling eHome Systems. In: Terzis, Sotirios (Hrsg.): *Proceedings of the 4th Intl. Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006)*. New York, NY, USA : ACM Press, 2006. – ISBN 1–59593–421–9, S. 1–6

[NMA06] Norbisrath, Ulrich ; Mosler, Christof ; Armaç, Ibrahim: The eHomeConfigurator Tool Suite. In: Meersman, Robert (Hrsg.) ; Tari, Zahir (Hrsg.) ; Herrero, Pilar (Hrsg.): *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, Part II, 1st International Workshop on Pervasive Systems (PerSys 2006)*, Springer, 2006 (LNCS 4278). – ISBN 3–540–48273–3, S. 1315–1324

[RAN09] Retkowitz, Daniel ; Armaç, Ibrahim ; Nagl, Manfred: Towards Mobility Support in Smart Environments. In: *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009)*. 3420 Main Street, Skokie, Illinois 60076, USA : Knowledge Systems Institute Graduate School, 2009. – ISBN 1–891706–24–1, S. 603–608

Literaturverzeichnis

[ACKK06] Assad, Mark ; Carmichael, David J. ; Kay, Judy ; Kummerfeld, Bob: Active Models for Context-Aware Services / School of Information Technologies, University of Sydney. 2006 (594). – Forschungsbericht

[ACKK07] Assad, Mark ; Carmichael, David J. ; Kay, Judy ; Kummerfeld, Bob: PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In: LaMarca, Anthony (Hrsg.) ; Langheinrich, Marc (Hrsg.) ; Truong, Khai N. (Hrsg.): *Pervasive 2007* Bd. 4480, Springer, 2007 (LNCS). – ISBN 978–3–540–72036–2, S. 55–72

[ADB+99] Abowd, Gregory D. ; Dey, Anind K. ; Brown, Peter J. ; Davies, Nigel ; Smith, Mark ; Steggles, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *HUC '99: Proceedings of the* 1st *International Symposium on Handheld and Ubiquitous Computing*, Springer, 1999. – ISBN 3–540–66550–1, S. 304–307

[AGGH06] Arroyo, Roberto F. ; Gea, Miguel ; Garrido, José L. ; Haya, Pablo A.: A Task-Driven Design Model for Collaborative AmI Systems. In: Latour, Thibaud (Hrsg.) ; Petit, Michaël (Hrsg.): *Proceedings of the CAiSE'06 Workshops and Doctoral Consortium*, Presses Universitaires de Namur, 2006. – ISBN 2–87037–525–5, S. 969–983. – Workshop: UMICS'06

[AK06] Armaç, Ibrahim ; Kirchhof, Michael: Process Support in eHome Systems: Empowering Providers to Handle a Future Mass Market. In: Latour, Thibaud (Hrsg.) ; Petit, Michael (Hrsg.): *Proceedings of the CAiSE'06 Workshops and Doctoral Consortium*, Press Universitares de Namur, 2006. – ISBN 978–2–87037–525–9, S. 999–1013

[AKR07] Armaç, Ibrahim ; Kirchhof, Michael ; Richterich, Claus: Frictionless Service Interaction in Protected Areas: Collaboration in eHomes. In: Pernici, Barbara (Hrsg.) ; Gulla, Jon A. (Hrsg.): *Proceedings of the CAiSE'07 Workshops and Doctoral Consortium, Vol. 2*, Tapir Academic Press, 2007. – ISBN 978–82–519–2246–3, S. 515–529

[All79] Allen, James F.: *A plan-based approach to speech act recognition*, University of Toronto, Canada, Diss., 1979

[All03] Allman, Mark: An Evaluation of XML-RPC. In: *SIGMETRICS Perform. Eval. Rev.* 30 (2003), Nr. 4, S. 2–11. – ISSN 0163–5999

[AMCK+02] Al-Muhtadi, Jala ; Campbell, Roy ; Kapadia, Apu ; Mickunas, Dennis ; Yi, Seung: Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In: *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, IEEE Computer Society, 2002. – ISBN 0–7695–1585–1, S. 74–83

[AMW99] Anderson, Mark ; Martin, Karl ; Walsh, Tom: *The Residential Gateway: Expanding the Horizons of Home Networking.* `http://citeseer.ist.psu.edu/419782.html`, 1999

[AN08] Assaf, Ali ; Noyé, Jacques: Dynamic AspectJ. In: *DLS '08: Proceedings of the 2008 Symposium on Dynamic Languages.* New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–270–2, S. 1–12

[AR07] Armaç, Ibrahim ; Retkowitz, Daniel: Simulation of Smart Environments. In: *Proceedings of the IEEE International Conference on Pervasive Services 2007 (ICPS'07)*, IEEE Press, 2007, S. 257–266

[AT99] Adomavicius, Gediminas ; Tuzhilin, Alexander: User Profiling in Personalization Applications through Rule Discovery and Validation. In: *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 1999. – ISBN 1–58113–143–7, S. 377–381

[AZA08] Arning, Kathrin ; Ziefle, Martina ; Arning, Jan: Comparing Apples and Oranges? Exploring users' acceptance of ICT and eHealth applications. In: *International Conference on Health Care Systems, Ergonomics, and Patient Safety (HEPS 2009)*, 2008

[Bay10] Bayerischer Forschungsverbund für Situierung, Individualisierung und Personalisierung in der Mensch-Maschine-Interaktion: *FORSIP.* 2010

[BB96] Bellotti, Victoria ; Bly, Sara: Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team. In: *CSCW '96: Proceedings of the first ACM Conference on Computer Supported Cooperative Work*, ACM, 1996. – ISBN 0–89791–765–0, S. 209–218

[BB02] Banavar, Guruduth ; Bernstein, Abraham: Software infrastructure and design challenges for ubiquitous computing applications. In: *Commun. ACM* 45 (2002), Dezember, Nr. 12, S. 92–96. – ISSN 0001–0782

[BBC08] Bauer, David ; Blough, Douglas ; Cash, David: Minimal Information Disclosure with Efficiently Verifiable Credentials. In: *Proceedings of the 4th*

ACM workshop on Digital identity management, ACM, 2008. – ISBN 978–1–60558–294–8, S. 15–24

[BDD07] Brands, Stefan ; Demuynck, Liesje ; Decker, Bart D.: A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users. In: Pieprzyk, Josef (Hrsg.) ; Ghodosi, Hossein (Hrsg.) ; Dawson, Ed (Hrsg.): *Proceedings of the 12th Australasian Conference on Information Security and Privacy (ACISP 2007)*, Springer, 2007 (LNCS 4586). – ISBN 978–3–540–73457–4, S. 400–415

[BDP07] Blazic, Aleksej J. ; Dolinar, Kajetan ; Porekar, Jan: Enabling Privacy in Pervasive Computing Using Fusion of Privacy Negotiation, Identity Management and Trust Management Techniques. In: *Proceedings of the First International Conference on the Digital Society*, IEEE Computer Society, 2007. – ISBN 0–7695–2760–4, S. 30–30

[BGKS02] Brookshier, Daniel ; Govoni, Darren ; Krishnan, Navaneeth ; Soto, Juan C.: *JXTA: Java P2P Programming*. Sams Publishing, 2002

[Bis02] Biskup, Joachim: Credential-basierte Zugriffskontrolle: Wurzeln und ein Ausblick. In: *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI)*, GI, 2002. – ISBN 3–88579–348–2, S. 423–428

[BKN06] Bellare, Mihir ; Kohno, Tadayoshi ; Namprempre, Chanathip: *The Secure Shell (SSH) Transport Layer Encryption Modes*. `http://tools.ietf.org/html/rfc4344`, Januar 2006

[BLHL01] Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In: *Scientific American* (2001), Mai

[Boe06] Boehm, Oliver: *Aspektorientierte Programmierung mit AspectJ 5*. dpunkt-Verlag, 2006. – ISBN 978–3898643306

[Bon04] Bonér, Jonas: What are the key issues for commercial AOP use – how does AspectWerkz address them? In: Murphy, Gail C. (Hrsg.) ; Lieberherr, Karl J. (Hrsg.): *AOSD '04: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development*. New York, NY, USA : ACM, 2004. – ISBN 1–58113–842–3, S. 5–6

[Bra00] Brands, Stefan A.: *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. Cambridge, MA, USA : MIT Press, 2000. – ISBN 0262024918

[Bra02] Brands, Stefan A.: A Technical Overview of Digital Credentials / Credentica. 2002. – Forschungsbericht

[Bra04] Brands, Stefan: Non-Intrusive Cross-Domain Digital Identity Management. In: *Proceedings of the 3rd Annual PKI R&D Workshop in Gaithersburg*, 2004

[BSS05] Brusilovsky, Peter ; Sosnovsky, Sergey A. ; Shcherbinina, Olena: User Modeling in a Distributed E-Learning Architecture. In: Ardissono, Liliana (Hrsg.) ; Brna, Paul (Hrsg.) ; Mitrovic, Antonija (Hrsg.): *User Modeling 2005, 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005, Proceedings* Bd. 3538, Springer, 2005 (LNCS). – ISBN 3–540–27885–0, S. 387–391

[BT94] Brajnik, Giorgio ; Tasso, Carlo: A shell for developing non-monotonic user modeling systems. In: *Int. J. Hum.-Comput. Stud.* 40 (1994), Nr. 1, S. 31–62. – ISSN 1071–5819

[Bun05] Bunt, Andrea: User Modelling to Support User Customization. In: Ardissono, Liliana (Hrsg.) ; Brna, Paul (Hrsg.) ; Mitrovic, Antonija (Hrsg.): *User Modeling 2005* Bd. 3538, Springer, 2005 (LNCS). – ISBN 978–3–540–27885–6, S. 499–501

[CAMN+03] Campbell, Roy ; Al-Muhtadi, Jala ; Naldurg, Prasad ; Sampemane, Geetanjali ; Mickunas, Dennis: Towards Security and Privacy for Pervasive Computing. In: Okada, Mitsuhiro (Hrsg.) ; Pierce, Benjamin C. (Hrsg.) ; Scedrov, Andre (Hrsg.) ; Yonezawa, Akinori (Hrsg.): *ISSS'02: Proceedings of the 2002 Mext-NSF-JSPS International Conference on Software Security*, Springer, 2003 (LNCS 2609). – ISBN 3–540–00708–3, S. 1–15

[CCP04] Cozzolongo, Giovanni ; Carolis, Berardina D. ; Pizzutilo, Sebastiano: A Personal Agent Supporting Ubiquitous Interaction. In: Baldoni, Matteo (Hrsg.) ; Paoli, Flavio D. (Hrsg.) ; Martelli, Alberto (Hrsg.) ; Omicini, Andrea (Hrsg.): *Workshop "From Objects to Agents": Complex Systems and Rational Agents (WOA 2004)*, Pitagora Editrice Bologna, 2004. – ISBN 88–371–1533–4, S. 55–61

[CD04] *Kapitel* 12. In: Cook, Diane J. (Hrsg.) ; Das, Sayal K. (Hrsg.): *Lessons from an Adaptive Home.* Wiley, 2004, S. 273–298

[CG01] Chen, Kirk ; Gong, Li: *Programming Open Service Gateways with Java Embedded Server Technology.* Addison-Wesley Professional, 2001. – 480 S. – ISBN 0–201–71102–8

[CGRZ04] Creese, Sadie ; Goldsmith, Michael ; Roscoe, Bill ; Zakiuddin, Irfan: Authentication for Pervasive Computing. In: Hutter, Dieter (Hrsg.) ; Müller, Günter (Hrsg.) ; Stephan, Werner (Hrsg.) ; Ullmann, Markus (Hrsg.): *Proceedings of the first International Conference on Security in Pervasive Computing*, Springer, 2004 (LNCS 2802). – ISBN 978–3–540–20887–7, S. 116–129

[CH01] Councill, William T. ; Heineman, George T.: *Component Based Software Engineering: Putting the Pieces Together*. Addison-Wesley, 2001. – ISBN 0201704854

[CH02] Camenisch, Jan ; Herreweghen, Els V.: Design and Implementation of the idemix Anonymous Credential System. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*. New York, NY, USA : ACM Press, 2002. – ISBN 1–58113–612–9, S. 21–30

[Cha81] Chaum, David L.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. In: *Commun. ACM* 24 (1981), Nr. 2, S. 84–90. – ISSN 0001–0782

[Cha85] Chaum, David: Security without identification: transaction systems to make big brother obsolete. In: *Communications of the ACM* 28 (1985), Nr. 10, S. 1030–1044. – ISSN 0001–0782

[Cha90] Chaum, David: Showing Credentials without Identification: Transferring Signatures between Unconditionally Unlinkable Pseudonyms. In: Seberry, Jennifer (Hrsg.) ; Pieprzyk, Josef (Hrsg.): *AUSCRYPT '90: Proceedings of the international conference on cryptology on Advances in cryptology*, Springer, 1990 (LNCS 453). – ISBN 978–3–540–53000–8, S. 246–264

[Cha92] Chaum, David: Achieving Electronic Privacy. In: *Scientific American* 267 (1992), Nr. 2, S. 96–101. – ISSN 0036–8733

[Chi89] Chin, David N.: KNOME: Modeling what the User Knows in UC. In: Kobsa, Alfred (Hrsg.) ; Wahlster, Wolfgang (Hrsg.): *User Models in Dialog Systems*, Springer, 1989, S. 74–107

[Chi93] Chin, David N.: Acquiring user models. In: *Artificial Intelligence Review* 7 (1993), Nr. 3-4, S. 185–197

[Chr04] Christoph, Auerbach N.: *Anonymous Digital Identity in e-Government*. 2004. – 274 S. – Dissertation an der Universität Zürich

[CI07] Cardoso, Roberto S. ; Issarny, Valeroe: Architecting Pervasive Computing Systems for Privacy: A Survey. In: *WICSA '07: Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, IEEE Computer Society, 2007. – ISBN 0–7695–2744–2, S. 26

[CKK05] Carmichael, David J. ; Kay, Judy ; Kummerfeld, Bob: Consistent Modelling of Users, Devices and Sensors in a Ubiquitous Computing Environment. In: *User Modeling and User-Adapted Interaction* 15 (2005), August, Nr. 3-4, S. 197–234. – ISSN 0924–1868

[CL01] Camenisch, Jan ; Lysyanskaya, Anna: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation.

In: Pfitzmann, Birgit (Hrsg.): *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, Springer, 2001 (LNCS 2045). – ISBN 3–540–42070–3, S. 93–118

[CL02] Cranor, Lorrie F. ; Lessig, Lawrence: *Web Privacy with P3P*. Sebastopol, CA, USA : O'Reilly & Associates, Inc., 2002. – ISBN 0596003714

[CLMR06] Cranor, Lorrie ; Langheinrich, Marc ; Marchiori, Massimo ; Reagle, Joseph: *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*. `http://www.w3.org/TR/2006/NOTE-P3P11-20061113`, November 2006

[CND09] Chanda, Prasit B. ; Naessens, Vincent ; Decker, Bart D.: Anonymous, Yet Trustworthy Auctions. In: Godart, Claude (Hrsg.) ; Gronau, Norbert (Hrsg.) ; Sharma, Sushil K. (Hrsg.) ; Canals, Gérôme (Hrsg.): *Software Services for e-Business and e-Society, 9th IFIP WG 6.1 Conference on e-Business, e-Services and e-Society, I3E 2009* Bd. 305, Springer, 2009 (IFIP). – ISBN 978–3–642–04279–9, S. 225–239

[CNM83] Card, Stuart K. ; Newell, Allen ; Moran, Thomas P.: *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., 1983

[CPFJ04] Chen, Harry ; Perich, Filip ; Finin, Timothy W. ; Joshi, Anupam: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In: *1st Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2004), Networking and Services*, IEEE Computer Society, August 2004. – ISBN 0–7695–2208–4, S. 258–267

[CRI07a] Cardoso, Roberto S. ; Raverdy, Pierre-Guillaume ; Issarny, Valerny: A Privacy-Aware Service Discovery Middleware for Pervasive Environments. In: Etalle, Sandro (Hrsg.) ; Marsh, Stephen (Hrsg.): *Proceedings of IFIPTM 2007: Joint iTrust and PST Conferences on Privacy, Trust Management and Security* Bd. 238, Springer, 2007 (IFIP International Federation for Information Processing). – ISBN 978–0–387–73654–9, S. 59–74

[CRI07b] Cardoso, Roberto S. ; Raverdy, Pierre-Guillaume ; Issarny, Valérie: A Privacy-Aware Service Discovery Middleware for Pervasive Environments. In: Etalle, Sandro (Hrsg.) ; Marsh, Stephen (Hrsg.): *Proceedings of IFIPTM 2007: Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, Springer, 2007 (IFIP Advances in Information and Communication Technology 238). – ISBN 978–0–387–73654–9, S. 59–74

[CSS+05] Camenisch, Jan ; Shelat, Abhi ; Sommer, Dieter ; Fischer-Hübner, Simone ; Hansen, Marit ; Krasemann, Henry ; Lacoste, Gérard ; Leenes, Ronald ; Tseng, Jimmy: Privacy and Identity Management for Everyone. In: *DIM '05: Proceedings of the 2005 Workshop on Digital Identity Management*, ACM, 2005. – ISBN 1–59593–232–1, S. 20–27

[CUH+01] Cerqueira, Renato ; Ururahy, Cristina ; Hess, Christopher ; Carvalho, Dulcineia ; Roman, Manuel ; Rodriguez, Noemi ; Campbell, Roy: Support for Mobility in Active Spaces. In: *Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001)*, 2001

[DAS01] Dey, Anind K. ; Abowd, Gregory D. ; Salber, Daniel: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In: *Human - Computer Interaction* 16 (2001), Nr. 2, 3 & 4, S. 97–166. – ISSN 07370024

[Dij72] Dijkstra, Edsger W.: The humble programmer. In: *Commun. ACM* 15 (1972), Nr. 10, S. 859–866. – Turing Award lecture

[DR02] Daemen, Joan ; Rijmen, Vincent: *The Design of Rijndael: AES–the Advanced Encryption Standard*. Springer, 2002. – ISBN 978–3540425809

[DR06] Daemen, Thomas ; Rubinstein, Ira: The Identity Metasystem: Towards a Privacy-Compliant Solution to the Challenges of Digital Identity / Microsoft. 2006. – Forschungsbericht

[DTA10] Deutsche Telekom AG, T-Com Z.: *T-Com Haus – Homepage*. `http://t-com-haus.idmedia.com/`, 2010

[DVFS04] Damiani, Ernesto ; Vimercati, Sabrina De C. ; Fugazza, C ; Samarati, Pierangela: Semantics-aware Privacy and Access Control: Motivation and Preliminary Results. In: *Proceedings of the first Italian Semantic Web Workshop: Semantic Web Applications an Presepectives (SWAP)*, 2004

[Ecl08a] Eclipse Foundation: *embedded Rich Client Platform (eRCP)*. `http://www.eclipse.org/ercp/`, 2008

[Ecl08b] Eclipse Foundation: *SWT: The Standard Widget Toolkit*. `http://www.eclipse.org/swt/`, 2008

[Ecl10a] Eclipse Foundation: *About the Eclipse Foundation*. `http://www.eclipse.org/org/`, 2010

[Ecl10b] Eclipse Foundation: *AJDT: AspectJ Development Tools*. `http://www.eclipse.org/ajdt/`, April 2010

[Eve07] Evers, Daniel: *Mobilität personalisierter Dienste in eHome-Umgebungen*, RWTH Aachen University, Diplomarbeit, Mai 2007

[FD86] Finin, Tim ; Drager, David: GUMS: A General User Modeling System. In: *HLT '86: Proceedings of the workshop on Strategic computing natural language*. Morristown, NJ, USA : Association for Computational Linguistics, 1986, S. 224–230

[FD09] Farshchian, Babak A. ; Divitini, Monica: Collaboration Support for Mobile Users in Ubiquitous Computing Environments. In: Nakashima, Hideyuki (Hrsg.) ; Aghajan, Hamid (Hrsg.) ; Augusto, Juan C. (Hrsg.): *Handbook of Ambient Intelligence and Smart Environments*, Springer, 2009. – ISBN 0387938079, 9780387938073, S. 173–199

[FFSS01] Feigenbaum, Joan ; Freedman, Michael J. ; Sander, Tomas ; Shostack, Adam: Privacy Engineering for Digital Rights Management Systems. In: *DRM '01: Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management*, Springer, 2001. – ISBN 3–540–43677–4, S. 76–105

[FHMO04] Ferscha, Alois ; Hechinger, Manfred ; Mayrhofer, Rene ; Oberhauser, Roy: A Light-Weight Component Model for Peer-to-Peer Applications. In: *ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04)*, IEEE Computer Society, 2004. – ISBN 0–7695–2087–1, S. 520–527

[FJMP97] Federrath, Hannes ; Jerichow, Anja ; Müller, Jan ; Pfitzmann, Andreas: Unbeobachtbarkeit in Kommunikationsnetzen. In: Müller, Günter (Hrsg.): *Verlässliche IT-Systeme: zwischen Key Escrow und elektronischem Geld (VIS'97)*, Vieweg, 1997. – ISBN 3–528–05594–4, S. 191–210

[FKJ97] Fink, Josef ; Kobsa, Alfred ; Jaceniak, Igor: Individualisierung von Benutzerschnittstellen mit Hilfe von Datenchips für Personalisierungsinformation. In: *Der GMD-Spiegel* 1 (1997), S. 16–17. – ISSN 0724–4339

[FM95] Flinn, Bill ; Maurer, Hermann: Levels of Anonymity. In: *Journal of Universal Computer Science* 1 (1995), Nr. 1, S. 35–47

[FNTZ98] Fischer, Thorsten ; Niere, Jörg ; Torunski, Lars ; Zündorf, Albert: Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In: Ehrig, Hartmut (Hrsg.) ; Engels, Gregor (Hrsg.) ; Kreowski, Hans-Jörg (Hrsg.) ; Rozenberg, Grzegorz (Hrsg.): *Theory and Application of Graph Transformations* Bd. 1764, Springer, 1998 (LNCS). – ISBN 3–540–67203–6, S. 157–167

[Fow04] Fowler, Martin: *Inversion of Control Containers and the Dependency Injection pattern.* http://martinfowler.com/articles/injection.html, 2004

[FP01] Federrath, Hannes ; Pfitzmann, Andreas: Neues Datenschutzrecht und die Technik. In: Kubicek, Herbert (Hrsg.): *Internet at Future, Jahrbuch Telekommunikation und Gesellschaft*, Hüthig Verlag, 2001. – ISBN 3–7785–3935–3, S. 252–259

[Gab04] Gabriel, Peter: *Interaktive Benutzeroberflächen für eHome-Systeme*, Rheinisch-Westfälische Technische Hochschule Aachen, Diplomarbeit, 2004

[GEB99] Gladman, Brian ; Ellison, Carl ; Bohm, Nicholas: *Digital Signatures, Certificates and Electronic Commerce.* `http://jya.com/bg/digsig.pdf`, 1999

[GHJV08] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 2008. – ISBN 978–0–201–63361–0

[GHL08] Ganser, Andreas ; Hurtz, Stefan ; Lichter, Horst: A Server Side SOA Meta Model for Assigning Aspect Services. In: *Proceedings of the Workshop on Modeling Security (MODSEC08) at MODELS08*, 2008. – ISSN 1613–0073 Vol–413

[Gli06] Gliewe, Olaf: *Konzepte zur Dienstmigration in Virtuellen eHome-Umgebungen*, RWTH Aachen University, Diplomarbeit, März 2006

[Gol07] Goldberg, Ian ; Acquisti, Alessandro (Hrsg.) ; Gritzalis, Stefanos (Hrsg.) ; Lambrinoudakis, Costos (Hrsg.) ; Vimercati, Sabrina di (Hrsg.): *Privacy Enhancing Technologies for the Internet II: Ten Years Later.* 2007

[Gon01a] Gong, Li: A Software Architecture for Open Service Gateways. In: *IEEE Internet Computing* 5 (2001), Nr. 1, S. 64–70. – ISSN 1089–7801

[Gon01b] Gong, Li: JXTA: A Network Programming Environment. In: *IEEE Internet Computing* 5 (2001), S. 88–95. – ISSN 1089–7801

[Gon01c] Gong, Li: Project JXTA: A Technology Overview / Sun Microsystems, Inc. 2001. – Forschungsbericht

[Grü09] Grüssner, Roland: *Ausführung von eHome-Diensten auf mobilen Geräten*, RWTH Aachen, Diplomarbeit, 2009

[Gri98] Griffel, Frank: *Componentware.* dpunkt, 1998. – ISBN 3–932588–02–9

[GS08] Grinewitschus, Viktor ; Scherer, Klaus: inHaus-2: Ein neues Konzept für die kooperative Entwicklung von Lösungen für das Betreute Wohnen / Fraunhofer Publica [http://publica.fraunhofer.de/oai.har] (Germany). 2008. – Forschungsbericht

[GT00] Gruhn, Volker ; Thiel, Andreas: *Komponentenmodelle DCOM, Javabeans, Enterprise Java Beans, CORBA.* Addison-Wesley, 2000. – ISBN 3–8273–1724–X

[Gua98] Guarino, Nicola: Formal Ontology and Information Systems. In: *Proceedings of the First International Conference on Formal Ontology and Information Systems (FOIS 1998)*, IOS Press, 1998

[Gut99] Guttman, Erik: Service Location Protocol: Automatic Discovery of IP Network Services. In: *IEEE Internet Computing* 3 (1999), Juli, Nr. 4, S. 71–80. – ISSN 1089–7801

[HB03] Hansen, Marit ; Berlich, Peter: Identity Management Systems: Gateway and Guardian for Virtual Residences. In: *EMTEL Conference: New Media, Technology and Everyday Life in Europe Conference*, LSE - The London School Of Economics And Political Science, 2003

[HBCDF06] Heydt-Benjamin, Thomas S. ; Chae, Hee-Jin ; Defend, Benessa ; Fu, Kevin: Privacy for Public Transportation. In: Danezis, George (Hrsg.) ; Golle, Philippe (Hrsg.): *Privacy Enhancing Technologies* Bd. 4258, Springer, 2006 (LNCS), S. 1–19

[HBPP05] Hansen, Marit ; Borcea-Pfitzmann, Katrin ; Pfitzmann, Andreas: PRIME - Ein europäisches Projekt für nutzerbestimmtes Identitätsmanagement. In: *it - Information Technology* 47 (2005), Nr. 6, S. 352–359

[Hec01] Heckmann, Dominik: Ubiquitous User Modeling for Situated Interaction. In: Bauer, Mathias (Hrsg.) ; Gmytrasiewicz, Piotr J. (Hrsg.) ; Vassileva, Julita (Hrsg.): *UM '01: Proceedings of the 8th International Conference on User Modeling 2001* Bd. 2109, Springer, 2001 (LNCS). – ISBN 3–540–42325–7, S. 280–282

[Hec03a] Heckmann, Dominik: A Specialized Representation for Ubiquitous Computing and User Modeling. In: *Proceeding of the First Workshop on User Modeling for Ubiquitous Computing, UM 2003*, 2003

[Hec03b] Heckmann, Dominik: Introducing Situational Statements as an integrating Data Structure for User Modeling, Context-Awareness and Resource-Adaptive Computing. In: *LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, GI, 2003

[Hec06] Heckmann, Dominik: *Ubiquitous User Modeling.* IOS Press, 2006. – 296 S. – Dissertation an der Universität des Saarlandes. – ISBN 978–1–58603–608–9

[Hei09] Heise Online: *Über 1 Million Datensätze bei SchülerVZ abgesaugt.* `http://www.heise.de/newsticker/meldung/Ueber-1-Million-Datensaetze-bei-SchuelerVZ-abgesaugt-832232.html`, Oktober 2009

[HKRG03] Hansen, Marit ; Krasemann, Henry ; Rost, Martin ; Genghini, Riccardo: Datenschutzaspekte von Identitätsmanagementsystemen - Recht und Praxis in Europa. In: *Datenschutz und Datensicherheit* 7 (2003), Nr. 9

[HR03] Hansen, Marit ; Rost, Martin: Nutzerkontrollierte Verkettung - Pseudonyme, Credentials, Protokolle für Identitätsmanagement. In: *Datenschutz und Datensicherheit* 27 (2003), Nr. 5

[HSB+05] Heckmann, Dominik ; Schwartz, Tim ; Brandherm, Boris ; Schmitz, Michael ; Wilamowitz-Moellendorff, Margeritta von: GUMO — The General User Model Ontology. In: Ardissono, Liliana (Hrsg.) ; Brna, Paul (Hrsg.) ; Mitrovic, Antonija (Hrsg.): *User Modeling 2005, 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005, Proceedings* Bd. 3538, Springer, 2005 (LNCS). – ISBN 3–540–27885–0, S. 428–432

[HSBK05] Heckmann, Dominik ; Schwartz, Tim ; Brandherm, Boris ; Kröner, Alexander: Decentralized User Modeling with UserML and GUMO. In: Dolog, Peter (Hrsg.) ; Vassileva, Julita (Hrsg.): *Proceedings of the Workshop on Decentralized, Agent Based and Social Approaches to User Modeling (DASUM-05) in the 9th International Conference on User Modelling (UM2005)*, 2005, S. 61–66

[Hur08] Hurtz, Stefan: *Modellierung von Sicherheitsaspekten in einer serviceorientierten und modellgetriebenen Anwendungslandschaft*, RWTH Aachen University, Diplomarbeit, September 2008

[HVJS07] Hossain, A.K.M. M. ; Van, Hien N. ; Jin, Yunye ; Soh, Wee-Seng: Indoor Localization Using Multiple Wireless Technologies. In: *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, 2007. – ISBN 978–1–4244–1455–0, S. 1–8

[HWRB05] Hämmerle, Simone ; Wimmer, Matthias ; Radig, Bernd ; Beetz, Michael: Sensor-based Situated, Individualized, and Personalized Interaction in Smart Environments. In: Cremers, Armin B. (Hrsg.) ; Manthey, Rainer (Hrsg.) ; Martini, Peter (Hrsg.) ; Steinhage, Volker (Hrsg.): *INFORMATIK 2005 - Informatik LIVE! Band 1, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e. V.* Bd. 67. Bonn, Germany : GI, September 2005 (LNI). – ISBN 3–88579–396–2, S. 261–265

[IBM08] IBM: *Identity Mixer Language Specification.* 2008

[IBM10] IBM Corporation: *WebSphere Everyplace Micro Environment.* `http://www.ibm.com/software/wireless/weme/`, 2010

[ISO94] ISO (International Standard Organisation): *Basic Reference Model for Open Systems Interconnection.* `http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=18824&ICS1=35&ICS2=100&ICS3=1&scopelist=`, 1994. – Specification ISO/IEC 10731:1994, ICS 35.100.01

[IST01] IST Advisory Group (ISTAG), European Commission: *Scenarios for Ambient Intelligence in 2010*. 2001. – Final Report. Compiled by K. Ducatel, et al. Feb-2001. EC. Brussels, 2001. ISBN 9289407352.

[JKZ02] Jendricke, Uwe ; Kreutzer, Michael ; Zugenmaier, Alf: Pervasive Privacy with Identity Management. In: *Proceedings of the Workshop on Security in Ubiquitous Computing at UbiComp 2002*, 2002

[JM01] Jendricke, Uwe ; Markotten, Daniela G.: Identitätsmanagement: Einheiten und Systemarchitektur. In: Fox, Dirk (Hrsg.) ; Köhntopp, Marit (Hrsg.) ; Pfitzmann, Andreas (Hrsg.): *Verlässliche IT-Systeme – Sicherheit in komplexen Infrastrukturen*, Vieweg, September 2001. – ISBN 3–528–05782–3, S. 77–85

[jxta] *JXTA JXME*. `https://jxta-jxme.dev.java.net/`

[jxtb] *jxta-rmi: Project Home*. `http://jxta-rmi.jxta.org/`. – Webseite inzwischen nicht mehr verfügbar

[jxtc] *JXTA SOAP bindings*. `http://soap.jxta.org/`

[jxtd] *XML-RPC for JXTA*. `http://xmlrpc.jxta.org/`

[Kay94] Kay, Judy: The um toolkit for cooperative user modelling. In: *User Modeling and User-Adapted Interaction* 4 (1994), Nr. 3, S. 149–196

[Kes99] Kesdogan, Dogan: *Privacy im Internet*. Vieweg Verlag, 1999. – ISBN 3–528–05731–9

[KF06] Kobsa, Alfred ; Fink, Josef: An LDAP-Based User Modeling Server and its Evaluation. In: *User Modeling and User-Adapted Interaction* 16 (2006), Nr. 2, S. 129–169. – ISSN 0924–1868

[Köh99] Köhntopp, Marit: Identitätsmanagement - Anforderungen aus Nutzersicht. In: Die Landesbeauftragte für den Datenschutz Nordrhein-Westfalen (Hrsg.): *Datenschutz und Anonymität*, Vieweg, 1999

[KHH+01] Kiczales, Gregor ; Hilsdale, Erik ; Hugunin, Jim ; Kersten, Mik ; Palm, Jeffrey ; Griswold, William G.: An Overview of AspectJ. In: *ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming*, Springer, 2001. – ISBN 3–540–42206–4, S. 327–353

[KHS01] Kirste, Thomas ; Herfet, Thorsten ; Schnaider, Michael: EMBASSI: Multimodal Assistance for Universal Access to Infotainment and Service Infrastructures. In: *WUAUC'01: Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing*, ACM, 2001. – ISBN 1–58113–424–X, S. 41–50

[Kir05] Kirchhof, Michael: *Integrierte Low-Cost eHome-Systeme – Prozesse und Infrastrukturen.* Shaker Verlag GmbH, 2005. – 346 S. – Dissertation an der RWTH Aachen. – ISBN 978–3–8322–4776–8

[KK06] Kay, Judy ; Kummerfeld, Bob: Scrutability, User Control and Privacy for Distributed Personalization. In: Kobsa, Alfred (Hrsg.) ; Chellappa, Ramnath K. (Hrsg.) ; Spiekermann, Sarah (Hrsg.): *Online Proceedings of PEP 2006, CHI 2006 Workshop on Privacy-Enhanced Personalization*, 2006

[KKL02] Kay, Judy ; Kummerfeld, Bob ; Lauder, Piers: Personis: A Server for User Models. In: Bra, Paul D. (Hrsg.) ; Brusilovsky, Peter (Hrsg.) ; Conejo, Ricardo (Hrsg.): *AH '02: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* Bd. 2347, Springer, 2002 (LNCS). – ISBN 3–540–43737–1, S. 203–212

[KKL03] Kay, Judy ; Kummerfeld, R.J. ; Lauder, Piers: Managing private user models and shared personas. In: *Proceeding of the First Workshop on User Modeling for Ubiquitous Computing at UM 2003*, Springer, Juni 2003 (LNCS 2702). – ISBN 978–3–540–40381–4

[KKR07] Küster, Ulrich ; König-Ries, Birgitta: Semantic Service Discovery with DIANE Service Descriptions. In: *Web Intelligence/IAT Workshops*, IEEE Computer Society, 2007. – ISBN 978–0–7695–3028–4, S. 152–156

[KKR09] Küster, Ulrich ; König-Ries, Birgitta: Supporting Dynamics in Service Descriptions - The Key to Automatic Service Usage. In: Krämer, Bernd J. (Hrsg.) ; Lin, Kwei-Jay (Hrsg.) ; Narasimhan, Priya (Hrsg.): *Service-Oriented Computing - ICSOC 2007*, Springer, 2009 (LNCS 4749). – ISBN 978–3–540–74973–8, S. 220–232

[KKRSK07] Küster, Ulrich ; König-Ries, Birgitta ; Stern, Mirco ; Klein, Michael: DIANE: An Integrated Approach to Automated Service Discovery, Matchmaking and Composition. In: *WWW '07: Proceedings of the 16th international conference on World Wide Web*, ACM, 2007. – ISBN 978–1–59593–654–7, S. 1033–1042

[KLK+06] Koutsoloukas, Eleftherios ; Lioudakis, Georgios V. ; Kapellaki, Sofia ; Dellas, Nikolaos L. ; Katsigiannis, Christos ; Prezerakos, George N. ; Kaklamani, Dimitra I. ; Venieris, I.S.: A Middleware Architecture for Privacy Protection in Smart Environments. In: *Proceedings of the 15th IST Mobile and Wireless Communications Summit*, 2006

[KLM+97] Kiczales, Gregor ; Lamping, John ; Mendhekar, Anurag ; Maeda, Chris ; Lopes, Cristina ; Loingtier, Jean-Marc ; Irwin, John: Aspect-Oriented Programming. In: Akşit, Mehmet (Hrsg.) ; Matsuoka, Satoshi (Hrsg.):

ECOOP'97 – Object-Oriented Programming Bd. 1241. Springer, 1997, S. 220–242. – ISBN 3–540–63089–9

[KOA+99] Kidd, Cory D. ; Orr, Robert ; Abowd, Gregory D. ; Atkeson, Christopher G. ; Essa, Irfan A. ; MacIntyre, Blair ; Mynatt, Elizabeth D. ; Starner, Thad ; Newstetter, Wendy: The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In: *CoBuild '99: Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture.* London, UK : Springer, 1999. – ISBN 3–540–66596–X, S. 191–198

[Kob04] Kobsa, Alfred: Adaptive Verfahren – Benutzermodellierung. In: Kuhlen, Rainer (Hrsg.) ; Seeger, Thomas (Hrsg.) ; Strauch, Dietmar (Hrsg.): *Grundlagen der Information und Dokumentation.* 5th. K. G. Saur, 2004

[Kob07] Kobsa, Alfred: Generic User Modeling Systems. In: Brusilovsky, Peter (Hrsg.) ; Kobsa, Alfred (Hrsg.) ; Nejdl, Wolfgang (Hrsg.): *The Adaptive Web: Methods and Strategies of Web Personalization (LNCS 4321).* Berlin : Springer, 2007, S. 136–154. – ISBN 978–3–540–72078–2

[KP95] Kobsa, Alfred ; Pohl, Wolfgang: The User Modeling Shell System BGP-MS. In: *User Modeling and User-Adapted Interaction* 4 (1995), Juni, Nr. 2, S. 59–106

[KP03] Köpsell, Stefan ; Pfitzmann, Andreas: Wie viel Anonymität verträgt unsere Gesellschaft? In: Knop, Jan von (Hrsg.) ; Haverkamp, Wilhelm (Hrsg.) ; Jessen, Eike (Hrsg.): *Security, E-Learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze, Düsseldorf, 2003* Bd. 44, GI, 2003 (LNI). – ISBN 3–88579–373–3, S. 13–26

[KS00] Kuflik, Tsvi ; Shoval, Peretz: Generation of User Profiles for Information Filtering — Research Agenda. In: *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in InformationRretrieval.* New York, NY, USA : ACM, 2000. – ISBN 1–58113–226–3, S. 313–315

[KS03] Kobsa, Alfred ; Schreck, Jörg: Privacy Through Pseudonymity in User-Adaptive Systems. In: *ACM Transactions on Internet Technology (TOIT)* 3 (2003), Nr. 2, S. 149–183. – ISSN 1533–5399

[KSW03] Karjoth, Günter ; Schunter, Matthias ; Waidner, Michael: Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In: *PET'02: Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, Springer, 2003. – ISBN 3–540–00565–X, S. 69–84

[KW89] Kobsa, Alfred ; Wahlster, Wolfgang: User Models in Dialog Systems. In: Kobsa, Alfred (Hrsg.) ; Wahlster, Wolfgang (Hrsg.): *User Models in Dialog Systems*. Berlin : Springer, 1989, S. 4–34. – ISBN 3540183809

[LAB+99] Lesser, Victor ; Atighetchi, Michael ; Benyo, Brett ; Horling, Bryan ; Raja, Anita ; Vincent, Regis ; Wagner, Thomas ; Ping, Xuan ; Zhang, Shelley X.: The Intelligent Home Testbed. In: *Proceedings of the Autonomy Control Software Workshop (Autonomous Agent Workshop)*, 1999

[LABW92] Lampson, Butler ; Abadi, Martín ; Burrows, Michael ; Wobber, Edward: Authentication in Distributed Systems: Theory and Practice. In: *ACM Transactions on Computer Systems* 10 (1992), Nr. 4, S. 265–310. – ISSN 0734–2071

[Lam74] Lampson, Butler W.: Protection. In: *ACM SIGOPS Operating Systems Review* 8 (1974), Nr. 1, S. 18–24. – ISSN 0163–5980

[Lan01a] Langheinrich, Marc: P3P – Ein neuer Standard für Datenschutz im Internet. In: *digma – Zeitschrift für Datenrecht und Informationssicherheit* 1 (2001), S. 32–34

[Lan01b] Langheinrich, Marc: Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems. In: Abowd, Gregory D. (Hrsg.) ; Brumitt, Barry (Hrsg.) ; Shafer, Steven A. (Hrsg.): *Proceedings of the Third International Conference on Ubiquitous Computing (UbiComp 2001)*, Springer, 2001 (LNCS 2201), S. 273–291

[Lan02] Langheinrich, Marc: A Privacy Awareness System for Ubiquitous Computing Environments. In: Borriello, Gaetano (Hrsg.) ; Holmquist, Lars E. (Hrsg.): *4th International Conference on Ubiquitous Computing (Ubicomp 2002)*, Springer, 2002 (LNCS 2498). – ISBN 978–3–540–44267–7, S. 237–245

[Lan05] Langheinrich, Marc: Die Privatsphäre im Ubiquitous Computing – Datenschutzaspekte der RFID-Technologie. In: Elgar Fleisch and Friedemann Mattern (Hrsg.): *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer, 2005, S. 329–362

[Lap08] Lapon, Jorn: PetAnon: A Fair and Privacy-Preserving Petition System. In: *FIDIS/IFIP Internet Security & Privacy Summer School*, 2008

[LCM02] Langheinrich, Marc ; Cranor, Lorrie ; Marchiori, Massimo: *APPEL: A P3P Preference Exchange Language*. `http://www.w3.org/TR/P3P-preferences/`, 2002

[Leg10] Legion of the Bouncy Castle: *Bouncy Castle Crypto APIs*. `http://www.bouncycastle.org/`, 2010

[LKD+07] Lioudakis, Georgios V. ; Koutsoloukas, Eleftherios A. ; Dellas, Nikolaos L. ; Tselikas, Nikolaos ; Kapellaki, Sofia ; Prezerakos, George N. ; Kaklamani, Dimitra I. ; Venieris, Iakovos S.: A middleware architecture for privacy protection. In: *Computer Networks* 51 (2007), Nr. 16, S. 4679–4696. – ISSN 1389–1286

[LM07] Leung, Adrian ; Mitchell, Chris J.: Ninja: Non Identity Based, Privacy Preserving Authentication for Ubiquitous Environments. In: Krumm, John (Hrsg.) ; Abowd, Gregory D. (Hrsg.) ; Seneviratne, Aruna (Hrsg.) ; Strang, Thomas (Hrsg.): *Proceedings of 9th International Conference on Ubiquitous Computing (UbiComp 2007)*, Springer, 2007 (LNCS 4717). – ISBN 978–3–540–74852–6, S. 73–90

[LMBE03] Loeser, Chris ; Mueller, Wolfgang ; Berger, Frank ; Eikerling, Heinz-Josef: Peer-to-Peer Networks for Virtual Home Environments. In: *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, IEEE Computer Society, 2003. – ISBN 0–7695–1874–5, S. 9 pp.

[Lor05] Lorenz, Andreas: A Specification for Agent-Based Distributed User Modelling in Ubiquitous Computing. In: Dolog, Peter (Hrsg.) ; Vassileva, Julita (Hrsg.): *Proceedings of the Workshop on Decentralized, Agent Based and Social Approaches to User Modeling (DASUM2005) at UM2005*, 2005, S. 31–40

[LSB09] Lüder, Marian ; Salomon, Ralf ; Bieber, Gerald: StairMaster: Ein neues Gerät zur online Erkennung von Stürzen. In: VDE (Hrsg.) ; AAL (Hrsg.) ; BMBF (Hrsg.): *Ambient Assisted Living 2009 – 2. Deutscher AAL-Kongress*, VDE Verlag, 2009. – ISBN 978–3–8007–3138–1

[LSM03] Loeser, Chris ; Schaefer, Robbie ; Mueller, Wolfgang: JXTA for Virtual Home Environments. In: *Proceedings of JXTA Workshop - Potenziale, Konzepte, Anwendungen*, 2003. – ISBN 3–9367–7115–4

[LW05] Lau, Kung-Kiu ; Wang, Zheng: A Taxonomy of Software Component Models. In: *EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE Computer Society, 2005. – ISBN 0–7695–2431–1, S. 88–95

[Mat08] Mattern, Friedemann: Allgegenwärtige Datenverarbeitung – Trends, Visionen, Auswirkungen. In: Rossnagel, Alexander (Hrsg.) ; Sommerlatte, Tom (Hrsg.) ; Winand, Udo (Hrsg.): *Digitale Visionen – Zur Gestaltung allgegenwärtiger Informationstechnologien*, Springer, 2008, S. 3–29

[MCF+94] Mitchell, Tom M. ; Caruana, Rich ; Freitag, Dayne ; McDermott, John ; Zabowski, David: Experience with a Learning Personal Assistant. In: *Communications of the ACM* 37 (1994), Nr. 7, S. 80–91. – ISSN 0001–0782

[Mei10] MEICHSNER, Julian: *Development and Evaluation of an eHome Simulator using Second Life/OpenSimulator*, RWTH Aachen University, Bachelorarbeit, März 2010

[Met10] METZ-WERKE GMBH: *mecaHome+*. `http://www.metz.de/de/tv-erlebnis/mecahome.html`, 2010

[Mic] MICROSYSTEMS, Sun: *Dynamic Proxy Classes*. `http://java.sun.com/j2se/1.4.2/docs/guide/reflection/proxy.html`

[Mic10] MICROSOFT: *U-Prove Documentation (March 2010)*. `https://connect.microsoft.com/site642/Downloads/DownloadDetails.aspx?DownloadID=26953`, 2010

[MJ01] MARKOTTEN, Daniela G. ; JENDRICKE, Uwe: Identitätsmanagement im E-Commerce. In: *it+ti Informationstechnik und Technische Informatik* 43 (2001), Nr. 5, S. 236–245

[MJM01a] MARKOTTEN, Daniela G. ; JENDRICKE, Uwe ; MÜLLER, Günter: Benutzbare Sicherheit – Der Identitätsmanager als universelles Sicherheitswerkzeug. In: MÜLLER, Günter (Hrsg.) ; REICHENBACH, Martin (Hrsg.): *Sicherheitskonzepte für das Internet*. Springer, 2001, S. 135–146. – ISBN 3–540–41703–6

[MJM01b] MARKOTTEN, Daniela G. ; JENDRICKE, Uwe ; MÜLLER, Günter: Benutzbare Sicherheit – Der Identitätsmanager als universelles Sicherheitswerkzeug. In: MÜLLER, Günter (Hrsg.) ; REICHENBACH, Martin (Hrsg.): *Sicherheitskonzepte für das Internet*. Springer, 2001, Kapitel 7, S. 135–146. – ISBN 3–540–41703–6

[ML05] MCAFFER, Jeff ; LEMIEUX, Jean-Michel: *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications*. Addison-Wesley Professional, 2005 (The Eclipse Series). – 552 S. – ISBN 978–0321334619

[MMS+07] MARIN, Andres ; MUELLER, Wolfgang ; SCHAEFER, Robbie ; ALMENAREZ, Florian ; DIAZ, Daniel ; ZIEGLER, Max: Middleware for Secure Home Access and Control. In: *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, 2007. – ISBN 0–7695–2788–4, S. 489–494

[Moo65] MOORE, Gordon E.: Cramming More Components onto Integrated Circuits. In: *Electronics* 38 (1965), April, Nr. 8, S. 114–117

[Moz98] MOZER, Michael C.: The Neural Network House: An Environment that Adapts to its Inhabitants. In: COEN, Michael H. (Hrsg.): *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, AAAI Press, 1998, S. 110–114

[MS09] Modersheim, Sebastian ; Sommer, Dieter: A Formal Model of Identity Mixer / IBM Zurich Research Lab. 2009. – Forschungsbericht. RZ 3749, Computer Science, 18 pages

[Nag90] Nagl, Manfred: *Softwaretechik: Methodisches Programmieren im Großen.* Springer, 1990. – ISBN 3–540–52705–2

[NARS06] Norbisrath, Ulrich ; Armaç, Ibrahim ; Retkowitz, Daniel ; Salumaa, Priit: Modeling eHome Systems. In: Terzis, Sotirios (Hrsg.): *Proceedings of the 4th Intl. Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006).* New York, NY, USA : ACM Press, 2006. – ISBN 1–59593–421–9, S. 1–6

[Nor07] Norbisrath, Ulrich: *Konfigurierung von eHome-Systemen.* Tartu University Press, 2007. – 293 S. – Dissertation an der RWTH Aachen. – ISBN 978–3–8322–4776–8

[NT94] Neuman, Clifford ; Ts'o, Theodore: Kerberos: An Authentication Service for Computer Networks. In: *IEEE Communications Magazine* 32 (1994), Nr. 9, S. 33–38

[Nus04] Nuseibeh, Bashar: Crosscutting Requirements. In: Murphy, Gail C. (Hrsg.) ; Lieberherr, Karl J. (Hrsg.): *AOSD '04: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development*, ACM, 2004. – ISBN 1–58113–842–3, S. 3–4

[NWET04] *Kapitel* Smart Environments: Technology, Protocols and Applications. In: Nixon, Paddy ; Wagealla, Waleed ; English, Colin ; Terzis, Sotirios: *Privacy, Security, and Trust Issues in Smart Environments.* Wiley, 2004, S. 220–240

[oe209] oe24.at: *Handy-Verbreitung – 2009 wird 4 Milliarden-Marke erreicht.* `http://www.oe24.at/digital/2009-wird-4-Milliarden-Marke-erreicht-0507136.ece`, August 2009

[Oel07] Oelmann, Guido: *Verhandlungsbasiertes und Privatsphäre schützendes Identitätsmanagement und Zugriffskontrolle für mobile Benutzer in eHomes*, RWTH Aachen University, Diplomarbeit, November 2007

[OG02] Oaks, Scott ; Gong, Li: *JXTA in a Nutshell.* O'Reilly & Associates, Inc., 2002. – ISBN 059600236X

[OKJ+01] Os, Els den ; Koning, Nicole D. ; Jongebloed, Hans ; Boves, Lou ; Nijmegen, Psycholinguistics: Usability of a Speech Centric Multimodal Directory Assistance Service. In: *Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue*, 2001, S. 65–69. – Online Proceedings

[ope10] openSimulator.org: *OpenSimulator*. http://opensimulator.org/wiki/Main_Page, 2010

[Orw94] Orwant, Jon: Heterogeneous learning in the Doppelgänger user modeling system. In: *User Modeling and User-Adapted Interaction* 4 (1994), Juni, Nr. 2, S. 107–130

[OSG09a] OSGi Alliance: *OSGi Service Platform Core Specification*. http://www.osgi.org/Download/File?url=/download/r4v42/r4.core.pdf, Juni 2009. – Relaese 4, Version 4.2

[OSG09b] OSGi Alliance: *OSGi Service Platform Specifications*. http://www.osgi.org/Download/Release4V42, September 2009. – Release 4.2

[Pad10] Paderborn, SmartHome: *SmartHome Paderborn – Homepage*. http://www.smarthomepaderborn.de/, 2010

[PAN05] Price, Blaine A. ; Adam, Karim ; Nuseibeh, Bashar: Keeping Ubiquitous Computing to Yourself: a practical model for user control of privacy. In: *International Journal of Human-Computer Studies* 63 (2005), Nr. 1-2, S. 228–253. – ISSN 1071–5819

[PBP06] Pfitzmann, Andreas ; Borcea-Pfitzmann, Katrin: Identitätsmanagement und informationelle Selbstbestimmung. In: *Allgegenwärtige Identifizierung? Neue Identitätsinfrastrukturen und ihre rechtliche Gestaltung; Schriftenreihe des Instituts für Europäisches Medienrecht (EMR)* 33 (2006), S. 83–91. ISBN 3832921273

[Pea02] Pearson, Siani: *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, 2002. – ISBN 978–0130092205

[Pet09] Pettau, Marcel: *Schutz der Privatsphäre in eHomes*, RWTH Aachen University, Diplomarbeit, Mai 2009

[PF06] Parrend, Pierre ; Frenot, Stephane: Secure Component Deployment in the OSGi(tm) Release 4 Platform / INRIA. 2006 (RT-0323). – Forschungsbericht

[Pfi06a] Pfitzmann, Andreas: Biometrie - wie einsetzen und wie keinesfalls? In: *Informatik Spektrum* 29 (2006), Nr. 5, S. 353–356

[Pfi06b] Pfitzmann, Andreas: Mehrseitige Sicherheit und Datenschutz. In: *Datenschutzkongress 2006 des Berufsverbandes der Datenschutzbeauftragten Deutschlands (BvD)*, 2006

[PGP07] PGP Corporation: *OpenPGP Message Format*. http://tools.ietf.org/html/rfc4880, November 2007

[PH09] Pfitzmann, Andreas ; Hansen, Marit: *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management.* 2009. – v0.32

[PM01] Pitt, Esmond ; McNiff, Kathy: *Java.rmi: The Remote Method Invocation Guide.* Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2001. – 320 S. – ISBN 0201700433

[PP09] Paar, Christof ; Pelzl, Jan: *Understanding Cryptography – A Textbook for Students and Practitioners.* Springer, 2009. – ISBN 978–3–642–04100–6

[PRI10a] PRIME Project: *PRIME - Privacy and Identity Management for Europe.* `https://www.prime-project.eu/`, 2010

[Pri10b] PrimeLife: *PrimeLife - Privacy and Identity Management in Europe for Life.* `http://www.primelife.eu/`, 2010

[PS95] Paiva, Ana ; Self, John A.: TAGUS — A User and Learner Modeling Workbench. In: *User Modeling and User-Adapted Interaction* 4 (1995), Nr. 3, S. 197–226. `http://dx.doi.org/http://dx.doi.org/10.1007/BF01100244`. – DOI http://dx.doi.org/10.1007/BF01100244. – ISSN 0924–1868

[PSK99] Pohl, Wolfgang ; Schwab, Ingo ; Koychev, Ivan: Learning About the User: A General Approach and Its Application. In: *Proceedings of IJCAI'99 Workshop: Learning About Users*, 1999. – Online Proceedings

[RAR07] Rellermeyer, Jan S. ; Alonso, Gustavo ; Roscoe, Timothy: R-OSGi: Distributed Applications Through Software Modularization. In: Cerqueira, Renato (Hrsg.) ; Campbell, Roy H. (Hrsg.): *Middleware '07: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware* Bd. 4834. New York, NY, USA : Springer, 2007 (LNCS). – ISBN 978–3–540–76777–0, S. 1–20

[RAW+08] Rymaszewski, Michael ; Au, Wagner J. ; Wallace, Mark ; Winters, Cathrine ; Batstone-Cunningham, Cory Ondrejka B.: *Second Life: The Official Guide.* 2. Indianapolis, IN : Wiley, 2008. – ISBN 978–0–470–22775–6

[RBLS07] Rukzio, Enrico ; Broll, Gregor ; Leichtenstern, Karin ; Schmidt, Albrecht: Mobile Interaction with the Real World: An Evaluation and Comparison of Physical Mobile Interaction Techniques. In: Schiele, Bernt (Hrsg.) ; Dey, Anind K. (Hrsg.) ; Gellersen, Hans (Hrsg.) ; Ruyter, Boris de (Hrsg.) ; Tscheligi, Manfred (Hrsg.) ; Wichert, Reiner (Hrsg.) ; Aarts, Emile (Hrsg.) ; Buchmann, Alejandro (Hrsg.): *European Conference on Ambient Intelligence (AmI 2007)* Bd. 4794, Springer, 2007 (LNCS). – ISBN 978–3–540–76651–3, S. 1–18

[Res03] Research, Philips: 365 days' Ambient Intelligent research in HomeLab. 2003. – Forschungsbericht. Technischer Bericht

[Ret10] Retkowitz, Daniel: *Softwareunterstützung für adaptive eHome-Systeme.* Shaker Verlag GmbH, 2010. – Dissertation an der RWTH Aachen, wird veröffentlicht

[RHC02] Roman, Manuel ; Ho, Herbert ; Campbell, Roy: Application Mobility in Active Spaces. In: *1st International Conference on Mobile and Ubiquitous Multimedia*, 2002

[Ric79] Rich, Elaine: User Modeling via Stereotypes. In: *Cognitive Science* 3 (1979), Nr. 4, S. 329–354. ISBN 1–55860–444–8

[RICLC06] Raverdy, Pierre-Guillaume ; Issarny, Valerie ; Chibout, Rafik ; La Chapelle, Agnes de: A Multi-protocol Approach to Service Discovery and Access in Pervasive Environments. In: *Proceedings of MOBIQUITOUS - The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services*, 2006

[RK05] Rivest, Ronald L. ; Kaliski, Burt: RSA Problem. In: *Encyclopedia of Cryptography and Security* (2005)

[RK09] Retkowitz, Daniel ; Kulle, Sven: Dependency Management in Smart Homes. In: Senivongse, Twittie (Hrsg.) ; Oliveira, Rui (Hrsg.): *Distributed Applications and Interoperable Systems, 9th IFIP WG 6.1 International Conference (DAIS 2009)* Bd. 5523, Springer, 2009 (LNCS), S. 143–156

[RL07] Ren, Kui ; Lou, Wenjing: Privacy-enhanced, Attack-resilient Access Control in Pervasive Computing Environments with Optional Context Authentication Capability. In: *Mobile Networks and Applications* 12 (2007), Nr. 1, S. 79–92. – ISSN 1383–469X

[Roß07] Rossnagel, Alexander: Informationelle Selbstbestimmung in der Welt des Ubiquitous Computing. In: Mattern, Friedemann (Hrsg.): *Die Informatisierung des Alltags – Leben in smarten Umgebungen*, Springer, 2007, S. 265–289

[Rod03] Roduner, Christof: *Citizen Controlled Data Protection in a SmartWorld (Personal Relationship Management with Credential Technology)*, Universität Zürich, Diplomarbeit, Mai 2003

[Ros03] Rost, Martin: Zur gesellschaftlichen Funktion von Anonymität - Anonymität im soziologischen Kontext. In: *Datenschutz und Datensicherheit* 27 (2003), Nr. 3

[Ros04] Rost, Martin: Verkettbarkeit als Grundbegriff des Datenschutzes? Identitätsmanagement soziologisch beobachtet. In: *Innovativer Datenschutz: für Helmut Bäumler*, 2004, S. 315–334

[Ros08] Rose, Daniel: *Modeling Mobile Users in Privacy-Protected eHomes*, RWTH Aachen, Diplomarbeit, 2008

[RP08] Retkowitz, Daniel ; Pienkos, Monika: Ontology-based Configuration of Adaptive Smart Homes. In: Taïani, François (Hrsg.) ; Cerqueira, Renato (Hrsg.): *Proceedings of the 7th Workshop on Reflective and Adaptive Middleware (ARM'08) held at the 9th International Middleware Conference*, ACM Press, 2008, S. 11–16

[RS06] Raub, Dominik ; Steinwandt, Rainer: An Algebra for Enterprise Privacy Policies closed unter Composition and Conjunction. (2006), Nr. 3995, S. 130–144. ISBN 978–3–540–34640–1

[RS08] Retkowitz, Daniel ; Stegelmann, Mark: Dynamic Adaptability for Smart Environments. In: Meier, René (Hrsg.) ; Terzis, Sotirios (Hrsg.): *Distributed Applications and Interoperable Systems, 8th IFIP WG 6.1 International Conference (DAIS 2008)* Bd. 5053, Springer, 2008 (LNCS), S. 154–167

[Rum04] Rumpe, Bernhard: *Modellierung mit UML*. Springer, 2004. – ISBN 3540209042

[SBJ03] Seigneur, Jean-Marc ; Biegel, Gregory ; Jensen, Christian D.: P2P with JXTA-Java pipes. In: *PPPJ '03: Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java*, Computer Science Press, Inc., 2003. – ISBN 0–9544145–1–9, S. 207–212

[SCFY96] Sandhu, Ravi S. ; Coyne, Edward J. ; Feinstein, Hal L. ; Youman, Charles E.: Role-Based Access Control Models. In: *Computer* 29 (1996), Nr. 2, S. 38–47. – ISSN 0018–9162

[Sch01] Schreck, Jörg: *Security and Privacy in User Modeling*. Kluwer Academic Publishers, 2001. – ISBN 140201130X

[SDF+04] Shavor, Sherry ; D'Anjou, Jim ; Fairbrother, Scott ; Kehn, Dan ; Kellerman, John ; McCarthy, Pat: *The Java Developer's Guide to Eclipse*. 2. Addison-Wesley Professional, 2004. – ISBN 978–0321305022

[Shi06] Shirehjinianlad, Ali Asghar N.: Context-Awarness: The DynAMITE Environment Model. In: Arbanowski, Stefan (Hrsg.) ; Kellerer, Wolfgang (Hrsg.) ; Steglich, Stephananlad (Hrsg.): *Proceedings of the 6th International Workshop on Applications and Services in Wireless Networks (ASWN 2006)*, Fraunhofer IRB Verlag, 2006. – ISBN 978–3–8167–7111–1, S. 26–32

[SLZ05] Specht, Marcus ; Lorenz, Andreas ; Zimmermann, Andreas: Towards a Framework for Distributed User Modelling for Ubiquitous Computing. In: Dolog, Peter (Hrsg.) ; Vassileva, Julita (Hrsg.): *Proceedings of the1st Workshop on Decentralized, Agent Based and Social Approaches to User Modeling (DASUM2005) at UM2005*, 2005, S. 80–85

[Sma10] Smart Home Systems, Inc.: *How X10 Works*. `http://www.smarthomeusa.com/info/x10theory/`, 2010

[SN99] Steimann, Friedrich ; Nejdl, Wolfgang: Modellierung und Ontologie / Universität Hannover. 1999. – Forschungsbericht

[Sou08] Sousa, Joao P.: Challenges and Architectural Approaches for Authenticating Mobile Users. In: *SAM '08: Proceedings of the 1st International Workshop on Software Architectures and Mobility*, ACM, 2008. – ISBN 978–1–60558–022–7, S. 15–20

[SS94] Sandhu, Ravi S. ; Samarati, Pierangela: Access Control: Principles and Practice. In: *IEEE Communications* 32 (1994), Nr. 9, S. 40–48

[SSA06] Sackmann, Stefan ; Strüker, Jens ; Accorsi, Rafael: Personalization in Privacy-Aware Highly Dynamic Systems. In: *Communications of the ACM* 49 (2006), Nr. 9, S. 32–38. – ISSN 0001–0782

[Sta02] Stajano, Frank: *Security for Ubiquitous Computing*. John Wiley and Sons, 2002. – ISBN 0–470–84493–0

[sue10] sueddeutsche.de: *Datenpanne, die zweite*. `http://www.sueddeutsche.de/finanzen/245/502478/text/`, Februar 2010

[Sun] Sun Microsystems: *The Java ME Platform*. `http://java.sun.com/javame/`

[SZM06] Schaefer, Robbie ; Ziegler, Max ; Mueller, Wolfgang: Securing Personal Data in Smart Home Environments. In: Kobsa, Alfred (Hrsg.) ; Chellappa, Ramnath K. (Hrsg.) ; (Hrsg.) ; Spiekermann, Sarah (Hrsg.): *Proceedings of the CHI2006 Workshop on Privacy-Enhanced Personalization*, 2006, S. 66–73

[TA90] Tay, B. H. ; Ananda, Akkihebbal L.: A Survey of Remote Procedure Calls. In: *SIGOPS Oper. Syst. Rev.* 24 (1990), Nr. 3, S. 68–79. – ISSN 0163–5980

[TP01] Tichy, Gunther ; Peissl, Walter: Beeinträchtigung der Privatsphäre in der Informationsgesellschaft - Detraction of Privacy in the Information Society / RePEc [http://oai.repec.openlib.org] (Germany). 2001. – Forschungsbericht

[Uni99] United States. National Institute of Standards and Technology: Data Encryption Standard (DES) / National Institute of Standards and Technology. 1999 (46-3). – FIPS pub. – 26 S

[Uni10] University of Illinois at Urbana-Champaign, Department of Computer Science: *Gaia Homepage*. http://gaia.cs.uiuc.edu/, 2010

[Vay05] Vaynerman, Igor: Service Personalization for User Support. In: Brass, Stefan (Hrsg.) ; Goldberg, Christian (Hrsg.): *17. GI-Workshop über Grundlagen von Datenbanken*, Institute of Computer Science, Martin-Luther-University, 2005, S. 143–147

[Wei91] Weiser, Mark: The Computer for the Twenty-First Century. In: *Scientific American* 265 (1991), September, Nr. 3, S. 94–104

[Wel09] Welt Online: *Datenpanne bei Internet-Buchhändler Libri weitet sich aus*. http://www.welt.de/die-welt/vermischtes/hamburg/article5035108/Datenpanne-bei-Internet-Buchhaendler-Libri-weitet-sich-aus.html, Oktober 2009

[WHFG92] Want, Roy ; Hopper, Andy ; Falcão, Veronica ; Gibbons, Jonathan: The active badge location system. In: *ACM Transactions on Information Systems* 10 (1992), Nr. 1, S. 91–102. – ISSN 1046–8188

[WHKL08] Wütherich, Gerd ; Hartmann, Nils ; Kolb, Bernd ; Lübken, Matthias: *Die OSGi Service Platform: Eine Einführung mit Eclipse Equinox*. Heidelberg : dpunkt, 2008

[WJM+03] Wohlgemuth, Sven ; Jendricke, Uwe ; Markotten, Daniela G. ; Dorner, Felix ; Müller, Günther: Sicherheit und Benutzbarkeit durch Identitätsmanagement. In: Spath, Dieter (Hrsg.) ; Haasis, Klaus (Hrsg.): *Aktuelle Trends in der Softwareforschung - Tagungsband zum doIT Software-Forschungstag 2003*. Stuttgart : IRB Verlag, 2003. – ISBN 3–8167–6453–3, S. 241–260

[WM06] Wohlgemuth, Sven ; Müller, Günter: Privatheit bei der Delegation von Rechten mit Identitätsmanagement. In: *Aktuelle Trends in der Softwareforschung, Tagungsband zum doIT Software-Forschungstag 2006* (2006), S. 9–27. ISBN 3–89864–413–8

[Wor10] World Wide Web Consortium (W3C): *P3P Public Overview*. http://www.w3.org/2000/07/p3p-brochure, 2010

[WP00] Wolf, Gritta ; Pfitzmann, Andreas: Properties of protection goals and their integration into a user interface. In: *Comput. Netw.* 32 (2000), Nr. 6, S. 685–699. – ISSN 1389–1286

[Wör03] Wörndl, Wolfgang: *Privatheit bei dezentraler Verwaltung von Benutzerprofilen*. 2003. – 169 S. – Dissertation an der Technischen Universität München

[YHC05] YOUNGBLOOD, G. M. ; HOLDER, Lawrence B. ; COOK, Diane J.: Managing Adaptive Versatile Environments. In: *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2299–8, S. 351–360

[ZO05] ZIMMERMAN, John ; OZEN, Kursat F.: Exploring Social Relationships Between Smart Homes and Their Occupants. In: *CHI 2005; First International Workshop on Social Implications of Ubiquitous Computing* (2005)

Lebenslauf

	Ibrahim Armaç Franzstr. 46 52064 Aachen
Geburtsdatum	20. September 1981
Geburtsort	Pertek/Türkei
Nationalität	deutsch
Schullaufbahn 09/1986 – 12/1989	Grundschule in Pertek/Türkei
01/1990 – 07/1995	Hauptschule in Niederzier
08/1995 – 06/1998	Gymnasium am Wirteltor in Düren Allgemeine Hochschulreife
Studium 10/1999 – 04/2005	Informatik an der RWTH Aachen Diplom-Informatiker
Promotion Seit 05/2005	RWTH Aachen Lehrstuhl Software Engineering Graduiertenkolleg „Software für mobile Kommunikationssysteme“

Related Interesting Work from the SE Group, RWTH Aachen

Agile Model Based Software Engineering

Agility and modeling in the same project? This question was raised in [Rum04]: "Using an executable, yet abstract and multi-view modeling language for modeling, designing and programming still allows to use an agile development process." Modeling will be used in development projects much more, if the benefits become evident early, e.g with executable UML [Rum02] and tests [Rum03]. In [GKRS06], for example, we concentrate on the integration of models and ordinary programming code. In [Rum12] and [Rum11], the UML/P, a variant of the UML especially designed for programming, refactoring and evolution, is defined. The language workbench MontiCore [GKR+06] is used to realize the UML/P [Sch12]. Links to further research, e.g., include a general discussion of how to manage and evolve models [LRSS10], a precise definition for model composition as well as model languages [HKR+09] and refactoring in various modeling and programming languages [PR03]. In [FHR08] we describe a set of general requirements for model quality. Finally [KRV06] discusses the additional roles and activities necessary in a DSL-based software development project.

Generative Software Engineering

The UML/P language family [Rum12, Rum11] is a simplified and semantically sound derivate of the UML designed for product and test code generation. [Sch12] describes a flexible generator for the UML/P based on the MontiCore language workbench [KRV10, GKR+06]. In [KRV06], we discuss additional roles necessary in a model-based software development project. In [GKRS06] we discuss mechanisms to keep generated and handwritten code separated. In [Wei12] we show how this looks like and how to systematically derive a transformation language in concrete syntax. To understand the implications of executability for UML, we discuss needs and advantages of executable modeling with UML in agile projects in [Rum04], how to apply UML for testing in [Rum03] and the advantages and perils of using modeling languages for programming in [Rum02].

Unified Modeling Language (UML)

Many of our contributions build on UML/P described in the two books [Rum11] and [Rum12] are implemented in [Sch12]. Semantic variation points of the UML are discussed in [GR11]. We discuss formal semantics for UML [BHP+98] and describe UML semantics using the "System Model" [BCGR09a], [BCGR09b], [BCR07b] and [BCR07a]. Semantic variation points have, e.g., been applied to define class diagram semantics [CGR08]. A precisely defined semantics for variations is applied, when checking variants of class diagrams [MRR11c] and objects diagrams [MRR11d] or the consistency of both kinds of diagrams [MRR11e]. We also apply these concepts to activity diagrams (ADs) [MRR11b] which allows us to check for semantic differences of activity diagrams [MRR11a]. We also discuss how to ensure and identify model quality [FHR08], how models, views and the system under development correlate to each other [BGH+98] and how to use modeling in agile development projects [Rum04], [Rum02] The question how to adapt and extend the UML in discussed in [PFR02] on product line annotations for UML and to more general discussions and insights on how to use meta-modeling for defining and adapting the UML [EFLR99], [SRVK10].

Domain Specific Languages (DSLs)

Computer science is about languages. Domain Specific Languages (DSLs) are better to use, but need appropriate tooling. The MontiCore language workbench [GKR+06], [KRV10], [Kra10] describes an integrated abstract and concrete syntax format [KRV07b] for easy development. New languages and tools

can be defined in modular forms [KRV08, Völ11] and can, thus, easily be reused. [Wei12] presents a tool that allows to create transformation rules tailored to an underlying DSL. Variability in DSL definitions has been examined in [GR11]. A successful application has been carried out in the Air Traffic Management domain [ZPK+11]. Based on the concepts described above, meta modeling, model analyses and model evolution have been examined in [LRSS10] and [SRVK10]. DSL quality [FHR08], instructions for defining views [GHK+07], guidelines to define DSLs [KKP+09] and Eclipse-based tooling for DSLs [KRV07a] complete the collection.

Modeling Software Architecture & the MontiArc Tool

Distributed interactive systems communicate via messages on a bus, discrete event signals, streams of telephone or video data, method invocation, or data structures passed between software services. We use streams, statemachines and components [BR07] as well as expressive forms of composition and refinement [PR99] for semantics. Furthermore, we built a concrete tooling infrastructure called MontiArc [HRR12] for architecture design and extensions for states [RRW13]. MontiArc was extended to describe variability [HRR+11] using deltas [HRRS11] and evolution on deltas [HRRS12]. [GHK+07] and [GHK+08] close the gap between the requirements and the logical architecture and [GKPR08] extends it to model variants. Co-evolution of architecture is discussed in [MMR10] and a modeling technique to describe dynamic architectures is shown in [HRR98].

Compositionality & Modularity of Models

[HKR+09] motivates the basic mechanisms for modularity and compositionality for modeling. The mechanisms for distributed systems are shown in [BR07] and algebraically underpinned in [HKR+07]. Semantic and methodical aspects of model composition [KRV08] led to the language workbench MontiCore [KRV10] that can even develop modeling tools in a compositional form. A set of DSL design guidelines incorporates reuse through this form of composition [KKP+09]. [Völ11] examines the composition of context conditions respectively the underlying infrastructure of the symbol table. Modular editor generation is discussed in [KRV07a].

Semantics of Modeling Languages

The meaning of semantics and its principles like underspecification, language precision and detailedness is discussed in [HR04]. We defined a semantic domain called “System Model” by using mathematical theory. [RKB95, BHP+98] and [GKR96, KRB96]. An extended version especially suited for the UML is given in [BCGR09b] and in [BCGR09a] its rationale is discussed. [BCR07a, BCR07b] contain detailed versions that are applied on class diagrams in [CGR08]. [MRR11a, MRR11b] encode a part of the semantics to handle semantic differences of activity diagrams and [MRR11e] compares class and object diagrams with regard to their semantics. In [BR07], a simplified mathematical model for distributed systems based on black-box behaviors of components is defined. Meta-modeling semantics is discussed in [EFLR99]. [BGH+97] discusses potential modeling languages for the description of an exemplary object interaction, today called sequence diagram. [BGH+98] discusses the relationships between a system, a view and a complete model in the context of the UML. [GR11] and [CGR09] discuss general requirements for a framework to describe semantic and syntactic variations of a modeling language. We apply these on class and object diagrams in [MRR11e] as well as activity diagrams in [GRR10]. [Rum12] embodies the semantics in a variety of code and test case generation, refactoring and evolution techniques. [LRSS10] discusses evolution and related issues in greater detail.

Evolution & Transformation of Models

Models are the central artifact in model driven development, but as code they are not initially correct and need to be changed, evolved and maintained over time. Model transformation is therefore essential to effectively deal with models. Many concrete model transformation problems are discussed: evolution [LRSS10, MMR10, Rum04], refinement [PR99, KPR97, PR94], refactoring [Rum12, PR03], translating models from one language into another [MRR11c, Rum12] and systematic model transformation language development [Wei12]. [Rum04] describes how comprehensible sets of such transformations support software development, maintenance and [LRSS10] technologies for evolving models within a language and across languages and linking architecture descriptions to their implementation [MMR10]. Automaton refinement is discussed in [PR94, KPR97], refining pipe-and-filter architectures is explained in [PR99]. Refactorings of models are important for model driven engineering as discussed in [PR03, Rum12]. Translation between languages, e.g., from class diagrams into Alloy [MRR11c] allows for comparing class diagrams on a semantic level.

Variability & Software Product Lines (SPL)

Many products exist in various variants, for example cars or mobile phones, where one manufacturer develops several products with many similarities but also many variations. Variants are managed in a Software Product Line (SPL) that captures the commonalities as well as the differences. Feature diagrams describe variability in a top down fashion, e.g., in the automotive domain [GHK+08] using 150% models. Reducing overhead and associated costs is discussed in [GRJA12]. Delta modeling is a bottom up technique starting with a small, but complete base variant. Features are added (that sometimes also modify the core). A set of applicable deltas configures a system variant. We discuss the application of this technique to Delta-MontiArc [HRR+11, HRR+11] and to Delta-Simulink [HKM+13]. Deltas can not only describe spacial variability but also temporal variability which allows for using them for software product line evolution [HRRS12]. [HHK+13] describes an approach to systematically derive delta languages. We also apply variability to modeling languages in order to describe syntactic and semantic variation points, e.g., in UML for frameworks [PFR02]. And we specified a systematic way to define variants of modeling languages [CGR09] and applied this as a semantic language refinement on Statecharts in [GR11].

Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) [KRS12] are complex, distributed systems which control physical entities. Contributions for individual aspects range from requirements [GRJA12], complete product lines [HRRW12], the improvement of engineering for distributed automotive systems [HRR12] and autonomous driving [BR12a] to processes and tools to improve the development as well as the product itself [BBR07]. In the aviation domain, a modeling language for uncertainty and safety events was developed, which is of interest for the European airspace [ZPK+11]. A component and connector architecture description language suitable for the specific challenges in robotics is discussed in [RRW13]. Monitoring for smart and energy efficient buildings is developed as Energy Navigator toolset [KPR12, FPPR12, KLPR12].

State Based Modeling (Automata)

Today, many computer science theories are based on state machines in various forms including Petri nets or temporal logics. Software engineering is particularly interested in using state machines for modeling systems. Our contributions to state based modeling can currently be split into three parts: (1) understanding how to model object-oriented and distributed software using statemachines resp. Statecharts

[GKR96, BCR07b, BCGR09b, BCGR09a], (2) understanding the refinement [PR94, RK96, Rum96] and composition [GR95] of statemachines, and (3) applying statemachines for modeling systems. In [Rum96] constructive transformation rules for refining automata behavior are given and proven correct. This theory is applied to features in [KPR97]. Statemachines are embedded in the composition and behavioral specifications concepts of Focus [BR07]. We apply these techniques, e.g., in MontiArcAutomaton [THR+13] as well as in building management systems [FLP+11].

Robotics

Robotics can be considered a special field within Cyber-Physical Systems which is defined by an inherent heterogeneity of involved domains, relevant platforms, and challenges. The engineering of robotics applications requires composition and interaction of diverse distributed software modules. This usually leads to complex monolithic software solutions hardly reusable, maintainable, and comprehensible, which hampers broad propagation of robotics applications. The MontiArcAutomaton language [RRW12] extends ADL MontiArc and integrates various implemented behavior modeling languages using MontiCore [RRW13] that perfectly fits Robotic architectural modelling. The LightRocks [THR+13] framework allows robotics experts and laymen to model robotic assembly tasks.

Automotive, Autonomic Driving & Intelligent Driver Assistance

Introducing and connecting sophisticated driver assistance, infotainment and communication systems as well as advanced active and passive safety-systems result in complex embedded systems. As these feature-driven subsystems may be arbitrarily combined by the customer, a huge amount of distinct variants needs to be managed, developed and tested. A consistent requirements management that connects requirements with features in all phases of the development for the automotive domain is described in [GRJA12]. The conceptual gap between requirements and the logical architecture of a car is closed in [GHK+07, GHK+08]. [HKM+13] describes a tool for delta modeling for Simulink [HKM+13]. [HRRW12] discusses means to extract a well-defined Software Product Line from a set of copy and paste variants. Quality assurance, especially of safety-related functions, is a highly important task. In the Carolo project [BR12a, BR12b], we developed a rigorous test infrastructure for intelligent, sensor-based functions through fully-automatic simulation [BBR07]. This technique allows a dramatic speedup in development and evolution of autonomous car functionality, and thus, enables us to develop software in an agile way [BR12a]. [MMR10] gives an overview of the current state-of-the-art in development and evolution on a more general level by considering any kind of critical system that relies on architectural descriptions. As tooling infrastructure, the SSElab storage, versioning and management services [HKR12] are essential for many projects.

Energy Management

In the past years, it became more and more evident that saving energy and reducing CO2 emissions is an important challenge. Thus, energy management in buildings as well as in neighbourhoods becomes equally important to efficiently use the generated energy. Within several research projects, we developed methodologies and solutions for integrating heterogeneous systems at different scales. During the design phase, the Energy Navigators Active Functional Specification (AFS) [FPPR12, KPR12] is used for technical specification of building services already. We adapted the well-known concept of statemachines to be able to describe different states of a facility and to validate it against the monitored values [FLP+11]. We show how our data model, the constraint rules and the evaluation approach to compare sensor data can be applied [KLPR12].

Cloud Computing & Enterprise Information Systems

The paradigm of Cloud Computing is arising out of a convergence of existing technologies for web-based application and service architectures with high complexity, criticality and new application domains. It promises to enable new business models, to lower the barrier for web-based innovations and to increase the efficiency and cost-effectiveness of web development. Application classes like Cyber-Physical Systems [KRS12], Big Data, App and Service Ecosystems bring attention to aspects like responsiveness, privacy and open platforms. Regardless of the application domain, developers of such systems are in need for robust methods and efficient, easy-to-use languages and tools. We tackle these challenges by perusing a model-based, generative approach [PR13]. The core of this approach are different modeling languages that describe different aspects of a cloud-based system in a concise and technology-agnostic way. Software architecture and infrastructure models describe the system and its physical distribution on a large scale. We apply cloud technology for the services we develop, e.g., the SSELab [HKR12] and the Energy Navigator [FPPR12, KPR12] but also for our tool demonstrators and our own development platforms. New services, e.g.,c collecting data from temperature, cars etc. are easily developed.

References

[BBR07] Christian Basarke, Christian Berger, and Bernhard Rumpe. Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence. *Journal of Aerospace Computing, Information, and Communication (JACIC)*, 4(12):1158–1174, October 2007.

[BCGR09a] Manfred Broy, Maria Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Considerations and Rationale for a UML System Model. In Kevin Lano, editor, *UML 2 Semantics and Applications*, pages 43–61. John Wiley & Sons, 2009.

[BCGR09b] Manfred Broy, Maria Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Definition of the UML System Model. In Kevin Lano, editor, *UML 2 Semantics and Applications*, pages 63–93. John Wiley & Sons, 2009.

[BCR07a] Manfred Broy, Maria Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 2: The Control Model. Technical Report TUM-I0710, TU Munich, February 2007.

[BCR07b] Manfred Broy, Maria Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 3: The State Machine Model. Technical Report TUM-I0711, TU Munich, February 2007.

[BGH+97] Ruth Breu, Radu Grosu, Christoph Hofmann, Franz Huber, Ingolf Krüger, Bernhard Rumpe, Monika Schmidt, and Wolfgang Schwerin. Exemplary and Complete Object Interaction Descriptions. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Proceedings OOPSLA'97 Workshop on Object-oriented Behavioral Semantics*, TUM-I9737, TU Munich, 1997.

[BGH+98] Ruth Breu, Radu Grosu, Franz Huber, Bernhard Rumpe, and Wolfgang Schwerin. Systems, Views and Models of UML. In M. Schader and A. Korthaus, editors, *Proceedings of the Unified Modeling Language, Technical Aspects and Applications*. Physica Verlag, Heidelberg, 1998.

[BHP+98] Manfred Broy, Franz Huber, Barbara Paech, Bernhard Rumpe, and Katharina Spies. Software and System Modeling Based on a Unified Formal Semantics. In M. Broy and B. Rumpe, editors, *RTSE '97: Proceedings of the International Workshop on Requirements Targeting Software and Systems Engineering*, LNCS 1526, pages 43–68, Bernried, Germany, October 1998. Springer.

[BR07] Manfred Broy and Bernhard Rumpe. Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik-Spektrum*, 30(1):3–18, Februar 2007.

[BR12a] Christian Berger and Bernhard Rumpe. Autonomous Driving - 5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In *Proceedings of the 10th Workshop on Automotive Software Engineering (ASE 2012)*, pages 789–798, Braunschweig, Germany, September 2012.

[BR12b] Christian Berger and Bernhard Rumpe. Engineering Autonomous Driving Software. In C. Rouff and M. Hinchey, editors, *Experience from the DARPA Urban Challenge*. Springer, 2012.

[CGR08] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. System Model Semantics of Class Diagrams. Informatik-Bericht 2008-05, CfG Fakultät, TU Braunschweig, 2008.

[CGR09] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Variability within Modeling Language Definitions. In *Model Driven Engineering Languages and Systems. Proceedings of MODELS 2009*, LNCS 5795, pages 670–684, Denver, Colorado, USA, October 2009.

[EFLR99] Andy Evans, Robert France, Kevin Lano, and Bernhard Rumpe. Meta-Modelling Semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*. Kluver Academic Publisher, 1999.

[FHR08] Florian Fieber, Michaela Huhn, and Bernhard Rumpe. Modellqualität als Indikator für Softwarequalität: eine Taxonomie. *Informatik-Spektrum*, 31(5):408–424, Oktober 2008.

[FLP+11] Norbert Fisch, Markus Look, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. State-Based Modeling of Buildings and Facilities. In *Proceedings of the 11th International Conference for Enhanced Building Operations (ICEBO' 11)*, New York City, USA, October 2011.

[FPPR12] Norbert Fisch, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. The Energy Navigator - A Web-Platform for Performance Design and Management. In *Proceedings of the 7th International Conference on Energy Efficiency in Commercial Buildings (IEECB)*, Frankfurt a. M., Germany, April 2012.

[GHK+07] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, and Bernhard Rumpe. View-based Modeling of Function Nets. In *Proceedings of the Object-oriented Modelling of Embedded Real-Time Systems (OMER4) Workshop*, Paderborn, Germany, October 2007.

[GHK+08] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, Lutz Rothhardt, and Bernhard Rumpe. Modelling Automotive Function Nets with Views for Features, Variants, and Modes. In *Proceedings of 4th European Congress ERTS - Embedded Real Time Software*, Toulouse, 2008.

[GKPR08] Hans Grönniger, Holger Krahn, Claas Pinkernell, and Bernhard Rumpe. Modeling Variants of Automotive Systems using Views. In *Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen (MBEFF)*, Informatik Bericht 2008-01, pages 76–89, CFG Fakultät, TU Braunschweig, March 2008.

[GKR96] Radu Grosu, Cornel Klein, and Bernhard Rumpe. Enhancing the SysLab System Model with State. Technical Report TUM-I9631, TUM, Munich, Germany, 1996.

[GKR+06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore 1.0 - Ein Framework zur Erstellung und Verarbeitung domänspezifischer Sprachen. Technical Report 2006-04, CfG Fakultät, TU Braunschweig, August 2006.

[GKRS06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, and Martin Schindler. Integration von Modellen in einen codebasierten Softwareentwicklungsprozess. In *Proceedings der Modellierung 2006*, Lecture Notes in Informatics LNI P-82, Innsbruck, März 2006. GI-Edition.

[GR95] Radu Grosu and Bernhard Rumpe. Concurrent Timed Port Automata. Technical Report TUM-I9533, TUM, Munich, Germany, 1995.

[GR11] Hans Grönniger and Bernhard Rumpe. Modeling Language Variability. In *Workshop on Modeling, Development and Verification of Adaptive Systems. 16th Monterey Workshop*, LNCS 6662, pages 17–32, Redmond, Microsoft Research, 2011. Springer.

[GRJA12] Tim Gülke, Bernhard Rumpe, Martin Jansen, and Joachim Axmann. High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A Problem Statement. In *Requirements Engineering: Foundation for Software Quality. 18th International Working Conference, Proceedings, REFSQ 2012*, Essen, Germany, March 2012.

[GRR10] Hans Grönniger, Dirk Reiß, and Bernhard Rumpe. Towards a Semantics of Activity Diagrams with Semantic Variation Points. In *Model Driven Engineering Languages and Systems, Proceedings of MODELS*, LNCS 6394, Oslo, Norway, 2010. Springer.

[HHK+13] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, and Ina Schaefer. Engineering Delta Modeling Languages. In *Proceedings of the 17th International Software Product Line Conference (SPLC), Tokyo*, pages 22–31. ACM, September 2013.

[HKM+13] Arne Haber, Carsten Kolassa, Peter Manhart, Pedram Mir Seyed Nazari, Bernhard Rumpe, and Ina Schaefer. First-Class Variability Modeling in Matlab / Simulink. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, pages 11–18, New York, NY, USA, 2013. ACM.

[HKR+07] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. An Algebraic View on the Semantics of Model Composition. In D. H. Akehurst, R. Vogel, and R. F. Paige, editors, *Proceedings of the Third European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2007), Haifa, Israel*, pages 99–113. Springer, 2007.

[HKR+09] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Scaling-Up Model-Based-Development for Large Heterogeneous Systems with Compositional Modeling. In H. Arabnia and H. Reza, editors, *Proceedings of the 2009 International Conference on Software Engineeering in Research and Practice*, Las Vegas, Nevada, USA, 2009.

[HKR12] Christoph Herrmann, Thomas Kurpick, and Bernhard Rumpe. SSELab: A Plug-In-Based Framework for Web-Based Project Portals. In *Proceedings of the 2nd International Workshop on Developing Tools as Plug-Ins (TOPI) at ICSE 2012*, pages 61–66, Zurich, Switzerland, June 2012. IEEE.

[HR04] David Harel and Bernhard Rumpe. Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer*, 37(10):64–72, Oct 2004.

[HRR98] Franz Huber, Andreas Rausch, and Bernhard Rumpe. Modeling Dynamic Component Interfaces. In Madhu Singh, Bertrand Meyer, Joseph Gil, and Richard Mitchell, editors, *TOOLS 26, Technology of Object-Oriented Languages and Systems*. IEEE Computer Society, 1998.

[HRR+11] Arne Haber, Holger Rendel, Bernhard Rumpe, Ina Schaefer, and Frank van der Linden. Hierarchical Variability Modeling for Software Architectures. In *Proceedings of International Software Product Lines Conference (SPLC 2011)*. IEEE Computer Society, August 2011.

[HRR12] Arne Haber, Jan Oliver Ringert, and Bernhard Rumpe. MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems. Technical Report AIB-2012-03, RWTH Aachen, February 2012.

[HRRS11] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta Modeling for Software Architectures. *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VII, fortiss GmbH*, February 2011.

[HRRS12] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Evolving Delta-oriented Software Product Line Architectures. In *Large-Scale Complex IT Systems. Development, Operation and Management, 17th Monterey Workshop 2012*, LNCS 7539, pages 183–208, Oxford, UK, March 2012. Springer.

[HRRW12] Christian Hopp, Holger Rendel, Bernhard Rumpe, and Fabian Wolf. Einführung eines Produktlinienansatzes in die automotive Softwareentwicklung am Beispiel von Steuergerätesoftware. In *Software Engineering 2012: Fachtagung des GI-Fachbereichs Softwaretechnik in Berlin*, Lecture Notes in Informatics LNI 198, pages 181–192, 27. Februar - 2. März 2012.

[KKP+09] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design Guidelines for Domain Specific Languages. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09), Sprinkle, J., Gray, J., Rossi, M., Tolvanen, J.-P., (eds.)*, Techreport B-108, Helsinki School of Economics, Orlando, Florida, USA, October 2009.

[KLPR12] Thomas Kurpick, Markus Look, Claas Pinkernell, and Bernhard Rumpe. Modeling Cyber-Physical Systems: Model-Driven Specification of Energy Efficient Buildings. In *Proceedings of the Modelling of the Physical World Workshop MOTPW'12, Innsbruck, October 2012*, pages 2:1–2:6. ACM Digital Library, October 2012.

[KPR97] Cornel Klein, Christian Prehofer, and Bernhard Rumpe. Feature Specification and Refinement with State Transition Diagrams. In *Fourth IEEE Workshop on Feature Interactions in Telecommunications Networks and Distributed Systems*. P. Dini, IOS-Press, 1997.

[KPR12] Thomas Kurpick, Claas Pinkernell, and Bernhard Rumpe. Der Energie Navigator. In *Entwicklung und Evolution von Forschungssoftware. Tagungsband, Rolduc, 10.-11.11.2011*, Aachener Informatik-Berichte, Software Engineering Band 14. Shaker Verlag Aachen, 2012.

[Kra10] Holger Krahn. *MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering*. Aachener Informatik-Berichte, Software Engineering Band 1. Shaker Verlag, Aachen, Germany, 2010.

[KRB96] Cornel Klein, Bernhard Rumpe, and Manfred Broy. A stream-based mathematical model for distributed information processing systems - SysLab system model. In *Proceedings of the first International Workshop on Formal Methods for Open Object-based Distributed Systems*, pages 323–338. Chapmann & Hall, 1996.

[KRS12] Stefan Kowalewski, Bernhard Rumpe, and Andre Stollenwerk. Cyber-Physical Systems - eine Herausforderung für die Automatisierungstechnik? In *Proceedings of Automation 2012, VDI Berichte 2012*, pages 113–116. VDI Verlag, 2012.

[KRV06] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Roles in Software Development using Domain Specific Modelling Languages. In J. Gray, J.-P. Tolvanen, and J. Sprinkle, editors, *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling 2006 (DSM'06), Portland, Oregon USA*, Technical Report TR-37, pages 150–158, Jyväskylä University, Finland, 2006.

[KRV07a] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Efficient Editor Generation for Compositional DSLs in Eclipse. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM' 07), Montreal, Quebec, Canada*, Technical Report TR-38, pages 8–10, Jyväskylä University, Finland, 2007.

[KRV07b] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Integrated Definition of Abstract and Concrete Syntax for Textual Languages. In G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, editors, *Proceedings of the ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MODELS 2007), Nashville, TN, USA, October 2007*, LNCS 4735. Springer, 2007.

[KRV08] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Monticore: Modular Development of Textual Domain Specific Languages. In R. F. Paige and B. Meyer, editors, *Proceedings of the 46th International Conference Objects, Models, Components, Patterns (TOOLS-Europe), Zurich, Switzerland, 2008*, Lecture Notes in Business Information Processing LNBIP 11, pages 297–315. Springer, 2008.

[KRV10] Holger Krahn, Bernhard Rumpe, and Stefen Völkel. MontiCore: a Framework for Compositional Development of Domain Specific Languages. *International Journal on Software Tools for Technology Transfer (STTT)*, 12(5):353–372, September 2010.

[LRSS10] Tihamer Levendovszky, Bernhard Rumpe, Bernhard Schätz, and Jonathan Sprinkle. Model Evolution and Management. In *MBEERTS: Model-Based Engineering of Embedded Real-Time Systems, International Dagstuhl Workshop, Dagstuhl Castle, Germany*, LNCS 6100, pages 241–270. Springer, October 2010.

[MMR10] Tom Mens, Jeff Magee, and Bernhard Rumpe. Evolving Software Architecture Descriptions of Critical Systems. *IEEE Computer*, 43(5):42–48, May 2010.

[MRR11a] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. ADDiff: Semantic Differencing for Activity Diagrams. In *Proc. Euro. Soft. Eng. Conf. and SIGSOFT Symp. on the Foundations of Soft. Eng. (ESEC/FSE'11)*, pages 179–189. ACM, 2011.

[MRR11b] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. An Operational Semantics for Activity Diagrams using SMV. Technical Report AIB-2011-07, RWTH Aachen University, Aachen, Germany, July 2011.

[MRR11c] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: Class Diagrams Analysis Using Alloy Revisited. In *Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand*, LNCS 6981, pages 592–607, 2011.

[MRR11d] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Modal Object Diagrams. In *Proc. 25th Euro. Conf. on Object Oriented Programming (ECOOP'11)*, LNCS 6813, pages 281–305. Springer, 2011.

[MRR11e] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Semantically Configurable Consistency Analysis for Class and Object Diagrams. In *Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand*, LNCS 6981, pages 153–167. Springer, 2011.

[PFR02] Wolfgang Pree, Marcus Fontoura, and Bernhard Rumpe. Product Line Annotations with UML-F. In G. J. Chastek, editor, *Software Product Lines - Second International Conference, SPLC 2*, LNCS 2379, pages 188–197, San Diego, 2002. Springer.

[PR94] Barbara Paech and Bernhard Rumpe. A new Concept of Refinement used for Behaviour Modelling with Automata. In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME'94: Industrial Benefit of Formal Methods*, LNCS 873. Springer, October 1994.

[PR99] Jan Philipps and Bernhard Rumpe. Refinement of Pipe-and-Filter Architectures. In J. Davies J. M. Wing, J. Woodcock, editor, *FM'99 - Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing System*, LNCS 1708, pages 96–115. Springer, 1999.

[PR03] Jan Philipps and Bernhard Rumpe. Refactoring of Programs and Specifications. In H. Kilov and K. Baclawski, editors, *Practical foundations of business and system specifications*, pages 281–297. Kluwer Academic Publishers, 2003.

[PR13] Antonio Navarro Perez and Bernhard Rumpe. Modeling Cloud Architectures as Interactive Systems. In I. Ober, A. S. Gokhale, J. H. Hill, J. Bruel, M. Felderer, D. Lugato, and A. Dabholka, editors, *Proc. of the 2nd International Workshop on Model-Driven Engineering for High Performance and Cloud Computing. Co-located with MODELS 2013, Miami*, Sun SITE Central Europe Workshop Proceedings CEUR 1118, pages 15–24. CEUR-WS.org, 2013.

[RK96] Bernhard Rumpe and Cornel Klein. Automata Describing Object Behavior. In H. Kilov and W. Harvey, editors, *Specification of Behavioral Semantics in Object-Oriented Information Modeling*, pages 265–286. Kluwer Academic Publishers, 1996.

[RKB95] Bernhard Rumpe, Cornel Klein, and Manfred Broy. Ein strombasiertes mathematisches Modell verteilter informationsverarbeitender Systeme - Syslab-Systemmodell. Technical Report TUM-I9510, Technische Universität München, 1995.

[RRW12] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. A Requirements Modeling Language for the Component Behavior of Cyber Physical Robotics Systems. In N. Seyff and A. Koziolek, editors, *Modelling and Quality in Requirements Engineering: Essays Dedicated to Martin Glinz on the Occasion of His 60th Birthday*. Monsenstein und Vannerdat, Münster, 2012.

[RRW13] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. MontiArcAutomaton: Modeling Architecture and Behavior of Robotic Systems. In *Workshops and Tutorials Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), May 6-10, 2013, Karlsruhe, Germany*, pages 10–12, 2013.

[Rum96] Bernhard Rumpe. *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, ISBN 3-89675-149-2, 1996.

[Rum02] Bernhard Rumpe. Executable Modeling with UML - A Vision or a Nightmare? In T. Clark and J. Warmer, editors, *Issues & Trends of Information Technology Management in Contemporary Associations, Seattle*, pages 697–701. Idea Group Publishing, Hershey, London, 2002.

[Rum03] Bernhard Rumpe. Model-Based Testing of Object-Oriented Systems. In F. de Boer, M. Bonsangue, S. Graf, W.-P. de Roever, editor, *Formal Methods for Components and Objects*, LNCS 2852, pages 380–402. Springer, November 2003.

[Rum04] Bernhard Rumpe. Agile Modeling with the UML. In M. Wirsing, A. Knapp, and S. Balsamo, editors, *Radical Innovations of Software and Systems Engineering in the Future. 9th International Workshop, RISSEF 2002. Venice, Italy, October 2002*, LNCS 2941. Springer, October 2004.

[Rum11] Bernhard Rumpe. *Modellierung mit UML*. Springer, second edition, September 2011.

[Rum12] Bernhard Rumpe. *Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring*. Springer, second edition, Juni 2012.

[Sch12] Martin Schindler. *Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P*. Aachener Informatik-Berichte, Software Engineering Band 11. Shaker Verlag, Aachen, Germany, 2012.

[SRVK10] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. Metamodelling: State of the Art and Research Challenges. In *MBEERTS: Model-Based Engineering of Embedded Real-Time Systems, International Dagstuhl Workshop, Dagstuhl Castle, Germany*, LNCS 6100, pages 57–76, October 2010.

[THR+13] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. A New Skill Based Robot Programming Language Using UML/P Statecharts. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 461–466, Karlsruhe, Germany, May 2013. IEEE.

[Völ11] Steven Völkel. *Kompositionale Entwicklung domänenspezifischer Sprachen*. Aachener Informatik-Berichte, Software Engineering Band 9. Shaker Verlag, Aachen, Germany, 2011.

[Wei12] Ingo Weisemöller. *Generierung domänenspezifischer Transformationssprachen*. Aachener Informatik-Berichte, Software Engineering Band 12. Shaker Verlag, Aachen, Germany, 2012.

[ZPK+11] Massimiliano Zanin, David Perez, Dimitrios S Kolovos, Richard F Paige, Kumardev Chatterjee, Andreas Horst, and Bernhard Rumpe. On Demand Data Analysis and Filtering for Inaccurate Flight Trajectories. In D. Schaefer, editor, *Proceedings of the SESAR Innovation Days*. EUROCONTROL, November 2011.