

Diplomarbeit im Studiengang Medieninformatik
im Bereich Softwareentwicklung

Remote Konfigurationsmanagement

Konzeption und Entwicklung einer Service
Anwendung für Haushaltsgeräte auf Basis von OSGi

1. Prüfer: Prof. Dr. Matthias Hinkelmann
2. Prüfer: Dr. Stephan Rupp

Vorgelegt von Philip Matthias Alb
am 15. April 2005
an der Fachhochschule Stuttgart
Hochschule der Medien (HdM)

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der Frage, wie moderne Haushaltsgeräte wie Waschmaschinen, Kühlschränke oder Gefriertruhen über ein Netzwerk gewartet werden können. Es wird ein Weg gesucht, Geräte zu analysieren und mit neuer Software zu versorgen. Im Laufe der Arbeit wird ein offenes und herstellerunabhängiges Konzept entwickelt, das aus mehreren Schlüsselkomponenten besteht. Eine allgemeine und verständliche Beschreibung aller an der Wartung beteiligter Entitäten in einem semantischen Modell soll einen Hersteller- und Technologie-unabhängigen Austausch von Wissen ermöglichen. Dadurch wird es neuen Anbietern erleichtert, das Konzept für sich zu nutzen. Durch den Einsatz konfigurationsfreier Netzkomponenten soll die Verwaltung vor allem aus Sicht der Benutzer vereinfacht werden. Aufgrund der großen Anzahl der in heutigen Heimnetzen anzutreffenden, unterschiedlichen Technologien bedarf es zudem einer Komponente, die diese Heterogenität berücksichtigt. Die Lösung soll möglichst offen für unterschiedliche Technologien sein. Deshalb sieht das Konzept den Einsatz eines Gateways als zentrale Steuerkomponente vor, das unterschiedliche Teilnetze des Heimes, transparent verbindet. Nach der Untersuchung geeigneter Technologien für die technische Umsetzung wird eine Anwendung entworfen, die das Konzept prototypisch umsetzt.

Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst zu haben. Alle für meine Recherchen benutzen Quellen finden sich im Quellenverzeichnis. Weiter versichere ich, dass diese Diplomarbeit niemals in Deutschland oder einem anderen Land veröffentlicht wurde.

Philip Alb
Stuttgart, den 15. April 2005

Danksagung

Die vorliegende Diplomarbeit wurde in der Firma Alcatel SEL AG in der Abteilung Zentrale Unternehmensentwicklung / Services durchgeführt.

Mein herzlicher Dank geht an meinen Professor Herrn Dr. Matthias Hinkelmann für die Betreuung der Arbeit und seine nützlichen Tipps zur Abfassung der vorliegenden Schrift.

Meinem Betreuer bei Alcatel Herrn Dr. Stephan Rupp danke ich sehr herzlich für die Ermöglichung dieser Arbeit, die umfangreichen, fachlichen Anregungen und sein großes Interesse am Fortgang der Arbeit.

Ebenfalls möchte ich mich bei Matthias Duspiva für Tipps und Anregungen, sowie bei Franz Josef Banet und Rodolfo Lopez- Aladros für die Organisation der Studenten Meetings bedanken.

Mein besonderer Dank gilt meinen Kollegen Thorsten Dobelmann, Lan Wu und Shafeer Hajamohideen für die tatkräftige Unterstützung bei der Recherche und Entwicklung.

Inhaltsverzeichnis

1	Einleitung	4
1.1	Ziel der Diplomarbeit	4
1.2	Aufbau der Diplomarbeit	4
1.3	Anmerkung	5
2	Einführung	6
2.1	Chancen und Herausforderungen	6
2.2	Begriffsklärung Remote Konfigurationsmanagement	10
3	IT Landschaft heutiger Heimnetze	13
3.1	Anschlusstechnologien	15
3.1.1	IrDa	15
3.1.2	Bluetooth	15
3.1.3	Firewire	17
3.2	Übertragungstechnologien	18
3.2.1	Powerline	18
3.2.2	IEEE 802.11 / WLAN	19
3.3	Hausbussysteme	22
3.3.1	LON	23
3.3.2	EIB und KNX	23
3.4	Kommunikation auf der Anwendungsebene	24
3.4.1	UPnP	24
3.4.2	JINI	27
3.5	Neue Entwicklungen	29
3.5.1	CHAIN	29
3.5.2	Serve@Home	29
3.6	Entwicklung von der Sicherheit zum Komfort	30

4	Reparatur Szenario	32
4.1	Beschreibung	33
4.2	Beteiligte Entitäten	34
4.2.1	Residential Service Gateway	34
4.2.2	Heimnetz	35
4.2.3	Waschmaschine	36
4.2.4	Besitzer	36
4.2.5	Reparatur Service Provider (RSP)	36
4.2.6	Reparatur Service Mitarbeiter (RSA)	36
4.2.7	Identity Provider	37
4.3	Ablauf	37
5	Semantik	39
5.1	Ontologie	39
5.2	OWL	41
5.3	Protégé	45
5.4	Service Ontologie	47
5.5	Mapping	52
5.5.1	OWL/UML Umwandlung	53
6	OSGi	55
6.1	Service Plattform – Sparsamer Container für Dienste	56
6.2	Architektur	57
6.2.1	Lebenszyklus eines Bundles	58
6.2.2	Kooperation der Bundles durch Services	59
6.3	Gegenüberstellung von OSGi und UPnP	61
6.4	Zusammenfassung	61
7	Technische Umsetzung	63
7.1	Waschmaschine: Design und Implementierung	63
7.1.1	Design	64
7.1.2	Cyberlink	65
7.1.3	Control Nachrichten	65
7.1.4	Update Service	67

7.2	Gateway	69
7.2.1	Knopflerfish	70
7.2.2	Design	71
7.2.3	Base Driver	72
7.2.4	Ablauf	73
7.2.5	Implementierung	75
7.3	Weitere Entwicklung	76
8	Schlussbetrachtungen	78
8.1	Zusammenfassung und Fazit	78
8.2	Persönliches Fazit	79
9	Anhang	81
9.1	Abkürzungsverzeichnis	81
9.2	Quellenverzeichnis	83
9.3	Abbildungsverzeichnis	85
9.4	CD	86

1 Einleitung

Die Tatsache, dass es im privaten Wohnbereich immer mehr Geräte gibt, die einen Netzwerkanschluss besitzen und mit Softwarediensten erweiterbar sind, macht es erforderlich, dass man sich mit der Frage auseinandersetzt, wie und von wem solche Geräte verwaltet werden.

1.1 Ziel der Diplomarbeit

Ziel dieser Diplomarbeit ist die Entwicklung eines einfachen Konzepts zur Verwaltung und Wartung von intelligenten Haushaltsgeräten. Dies beinhaltet die Untersuchung vorhandener Konzepte und Lösungen, sowie eine Übersicht über die in heutigen Heimnetzen anzutreffenden Technologien. Es werden für die Umsetzung geeignete Technologien untersucht, die dann in der prototypischen Umsetzung eingesetzt werden. Dabei soll eine Service Anwendung einen konkreten Wartungsvorgang verdeutlichen. Die Ziele dieser Diplomarbeit lassen sich wie folgt definieren.

- Untersuchung vorhandener Konzepte und Lösungen für das Remote Konfigurations-Management
- Erstellung eines Konzepts für das Remote Konfigurations-Management
- Untersuchung geeigneter Technologien für die technische Umsetzung
- Prototypische Realisierung unter Einsatz der untersuchten Technologien

1.2 Aufbau der Diplomarbeit

Im zweiten Kapitel erfolgt eine Einführung in das Thema der Diplomarbeit. Hier wird der Bedarf des Konzepts und die damit verbundenen Herausforderungen gezeigt. In Kapitel 3 erhält der Leser Hintergrundinformationen über den aktuellen Stand der Technik in heutigen Heimnetzen. Eine Bestandsaufnahme über bestehende Lösungen und eingesetzte Technologien im Heimbereich

soll etwas Licht in das Thema bringen. Dabei soll auch die Entwicklung der Gebäudeautomation vom Sicherheitsbereich in Kraftwerken hinein in den Komfortbereich der Heimnetze gezeigt werden. Aktuelle Entwicklungen und Bestrebungen der Hersteller nach gemeinsamen Standards geben einen Ausblick auf die Zukunft der Heimnetze.

Als Grundlage für das Konfigurationskonzept wird im vierten Kapitel ein Szenario für den Einsatz einer Service Anwendung entworfen. Eine moderne Waschmaschine wird über ein Service Gateway gewartet und mit neuer Software versorgt. Das Service Gateway dient dabei als integrierender Teil über die unterschiedlichen Technologien im Heimnetz und soll die Komplexität des Gesamtsystems vor dem Nutzer verbergen.

Ausgehend von diesem Szenario wird im fünften Kapitel ein semantisches Modell, eine sog. Ontologie entworfen, die eine Schlüsselkomponente des Konzepts darstellt. Der Leser erhält eine Übersicht über Theorie, Entwicklung und Einsatz von Ontologien bevor die eigentliche Service Ontologie entwickelt wird. Als Basistechnologie für die Service Anwendung wird das relativ neue OSGi eingesetzt, welches in Kapitel 6 im Detail beschrieben wird, bevor in Kapitel 7 auf die technische Umsetzung eingegangen wird. Nachdem Entwurf des Software-Designs und der Beschreibung von weiteren eingesetzten Technologien erfolgt ein Ausblick auf die weitere technische Umsetzung. Im abschließenden letzten Kapitel erfolgt eine Zusammenfassung der Arbeit inkl. eines allgemeinen und eines persönlichen Fazits.

1.3 Anmerkung

Diese Diplomarbeit entstand bei der Firma Alcatel in Stuttgart Zuffenhausen in der Abteilung Zentrale Unternehmensentwicklung / Services. Die Arbeit ist eingegliedert in das von der EU geförderte Forschungsprojekt Magnet. Im Anschluss an diese Diplomarbeit werden die Implementierungsarbeiten voraussichtlich fortgesetzt. Zum Thema Konfigurationsmanagement für Haushaltsgeräte gab es gleichzeitig eine parallele Arbeit mit dem Schwerpunkt der Erstellung eines Sicherheitskonzeptes, für das Thorsten Dobelmann verantwortlich war und auf dessen Arbeit an einigen Stellen verwiesen wird.

2 Einführung

Das heutige Heim wird immer moderner. Nach dem phänomenalen Erfolg des Internets im privaten Wohnbereich in den letzten Jahren hält die Kommunikationstechnik nun auch Einzug in andere Bereiche des Heimes. Immer mehr Audio, Video und sogar Haushaltsgeräte werden vernetzt. Der Trend zum vernetzten Heim scheint unaufhaltbar. Schlagworte wie Pervasive Computing¹ und das intelligente Heim sind in aller Munde. Hauptgrund für diese Entwicklung ist die zunehmende Intelligenz der Geräte, die über immer mehr Funktionen verfügen. Neben der im Heimbereich fest etablierten PC-Vernetzung ist auch die Multimedia-Vernetzung nichts Exotisches mehr. Dadurch sind neue Anwendungen wie z. B. „Movies-On-Demand“, bei denen Filme bei Bedarf über das Internet bestellt, geliefert und abgerechnet werden, möglich. Neu ist, dass inzwischen, neben AV Geräten, Mobiltelefonen und Personal Data Assistants (PDA) auch klassische Haushaltsgeräte wie Kühl- oder Gefrierschränke netzfähig sind. Dies ermöglicht auch hier neue Anwendungen. Der Kühlschrank meldet selbstständig, dass die Milch abgelaufen ist, und die Waschmaschine kann mit dem neuesten, umweltschonenden Waschprogramm aktualisiert werden. Abbildung 1 zeigt die Integration unterschiedlicher Geräte im Heimnetz.

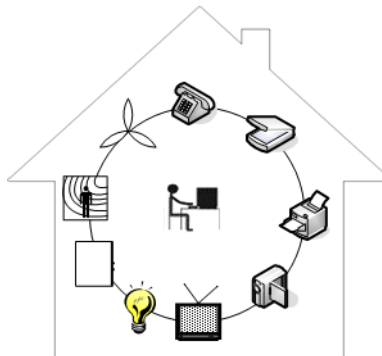


Abbildung 1 Anwendungsbereiche im Heimnetz

2.1 Chancen und Herausforderungen

Die Entwicklung im Heimbereich bietet große Chancen für Nutzer, Gerätehersteller und Service Provider, die Softwaredienste für diese Geräte anbieten. Service Provider und Gerätehersteller sind durch

¹ Pervasive Computing bezeichnet die totale Vernetzung „intelligenter“ Alltagsgegenstände

neue Angebote in der Lage ihre Kunden zu erreichen, und für Kunden bietet die neue Technik mehr Komfort zuhause. Beispielsweise leitet eine Funktion eines PC-basierten Systems für das intelligente Heim der Firma Tobit die Stimme eines Besuchers von der Türsprechanlage an das Mobiltelefon des Besitzers weiter. Dies geschieht erst, nachdem Bewegungssensoren überprüft haben, dass der Besitzer sich weder im Haus noch im Garten befindet. [2.01] Auch denkbar ist der Informationsaustausch eines stromhungrigen Gerätes mit einem Service Provider zur Ermittlung des günstigsten Stromtarifs.

Allerdings birgt diese Entwicklung auch große Herausforderungen. Lassen sich neue Geräte in bestehende Lösungen integrieren, und sind sie untereinander kompatibel? Kann der Nutzer die Geräte selbst anschließen und wie groß ist der administrative Aufwand? Und was ist im Falle eines Fehlers? Muss der Nutzer selbst neue Firmware oder Software auf das Gerät spielen oder das Gerät gar zur Reparatur bringen?² Gerade die Frage von Software Updates ist dabei auch für Gerätehersteller interessant, insbesondere wenn ein Fehler noch innerhalb des Garantiezeitraumes auftaucht. Durch die Möglichkeit eines Remote Software Updates ließe sich womöglich so manche teure Rückrufaktion vermeiden.

Obwohl das intelligente Heim schon seit Jahren propagiert wird, lässt der große Durchbruch zum Leid vieler Hersteller noch auf sich warten. Laut einer Studie von Harris Interactive sind bisher erst ein Fünftel aller US-amerikanischen Haushalte vernetzt. [2.02] Dies liegt zum einen an einem unzureichenden Angebot sinnvoller Anwendungen. Dazu kommt aber, dass die Kunden durch zu viele unterschiedliche Technologien, Protokolle und Standards im Heimbereich abgeschreckt werden. Das Fehlen von wirklich offenen Standards verhindert auch das Sinken der Preise und schmälert damit die Attraktivität von Heimvernetzungs-Technologien für den Massenmarkt.

² Als Firmware bezeichnet Software, die in ein Gerät eingebettet ist und für elementare Steueraufgaben verantwortlich ist. Sie wird vom Hersteller geliefert und ist für den Benutzer normalerweise nicht sichtbar. Obwohl Firmware permanent gespeichert wird kann sie bei Verwendung spezieller, programmierbarer Speicherbausteine aktualisiert werden. Dies kann genutzt werden um Fehler zu beheben oder die Funktionalität des Gerätes zu erweitern.

Einheitliche Standards

Es gibt eine Vielzahl unterschiedlicher Technologien, die im Heimnetz eingesetzt werden können. Von der Auswahl des Internetzugangs, der Anschlussart von Unterhaltungsgeräten bis hin zur Wahl des Übertragungsmediums ist die Auswahl riesig. Will man gar Licht und Heizung in das Netz integrieren kommt eine Vielzahl möglicher Hausbussysteme hinzu. Inzwischen gibt es Bestrebungen der Hersteller gemeinsame Standards zu schaffen. So arbeitet der Verband europäischer Haushaltsgerätehersteller an einem gemeinsamen Standard für ein Kommunikationsprotokoll für Haushaltsgeräte.³ Auch gibt es Bemühungen für eine Kommunikation von unterschiedlichen Geräten auf Anwendungsebene, wie etwa das Universal Plug And Play (UPnP) von Microsoft.⁴

Verwaltungsaufwand

Obwohl das Wachstum nicht so schnell voranschreitet wie ursprünglich prognostiziert, zeichnet sich bereits eine Konsequenz aus der zunehmenden Heimvernetzung ab. Durch die wachsende Komplexität der Heimsysteme steigen die Anforderungen an die Verwaltung der Geräte. Unterschiedliche Informationen müssen abrufbar sein, was die Speicherung in einer Datenbank voraussetzt. Es müssen z. B. nicht nur das Gerätemodell und die Seriennummer bekannt sein, sondern auch der aktuelle Softwarestand des Gerätes. Nur so kann ein Fehler, der vielleicht nur auf einer inkompatiblen Anwendung beruht, schnell und kostengünstig behoben werden. Als negatives Beispiel für zu wenig Information kann man sich einen modernen Mittelklassewagen vorstellen. Angenommen in dem Fahrzeug arbeiten etwa 100 Prozessoren, von denen alle in einer zentralen Datenbank registriert sind. Wenn jetzt die Softwarestände nicht ebenfalls hinterlegt sind, muss im Falle eines Defekts die Firmware aller Prozessoren komplett neu geladen werden. Dabei wird der ursprüngliche Softwarestand wiederhergestellt, was einen unnötigen Kostenaufwand darstellt. Software Updates von Drittanbietern wie beispielsweise Car Tuning Unternehmen gehen verloren. Auch ist der neueste Softwarestand für jedes Element nicht unbedingt immer die beste Lösung. Das Aufspielen der neusten Software Versionen ohne Berücksichtigung vorhandener Softwarestände kann zu unerwünschten Nebeneffekten führen. Es

³ Vgl. Kapitel 3.5.1 CHAIN

⁴ Vgl. Kapitel 3.4.1 UPnP

können Inkompatibilitäten mit installierten Softwarediensten von Drittanbietern auftreten. Die Verwaltung von Geräteprofilen würde die Wartung erheblich erleichtern.

Den Bedarf einer sinnvollen Geräteverwaltung für mobile Endgeräte hat die Firma Nokia in einem 2002 veröffentlichten White Paper dokumentiert. [2.03] Bei Einführung neuer Dienste für Mobiltelefone zur Anzeige reduzierter Internetinhalte oder zum Versenden von Multimedia Nachrichten über den Multimedia Message Service (MMS) waren einige Konfigurationsarbeiten erforderlich. So mancher Mobilfunk-Kunde war dazu nicht in der Lage, und somit von der Nutzung der neuen Dienste ausgeschlossen. Dies allein wäre schon schlimm genug. Nokia äußerte zudem die Befürchtung, dass viele Kunden den Hersteller verantwortlich machen, da dieser keine benutzerfreundlichere Möglichkeit der Konfiguration anbietet. Im schlimmsten Fall wechseln sie später zu einem anderen Anbieter. Nokia gelangt deshalb zu der Einsicht, dass Konfigurationsinformationen vom Hersteller oder einem Drittanbieter, einem Service Provider, verwaltet werden sollten und bei Bedarf auf das Gerät geladen werden. Für Mobiltelefone existieren bereits Lösungen für das Geräte Management. Einen solchen Bedarf wird es in Zukunft auch für Haushaltsgeräte geben. Denn auch Besitzer von Haushaltsgeräten wollen die Geräte nicht selbst verwalten, und in den allermeisten Fällen verfügen sie auch nicht über das technische Know-how. Vielmehr müssen die Hersteller selbst für die Verwaltung der Geräte sorgen. Es bedarf daher ebenfalls für Haushaltsgeräte eines Ansatzes zur zentralen und intelligenten Verwaltung der Geräteinformationen.

Beschreibung der Geräteinformationen

Für die Verwaltung der Geräte werden unterschiedliche Geräteinformationen benötigt. Dazu gehören z. B. die Seriennummer, das Modell und wie bereits erwähnt die derzeitige laufende Software. Man kann aufgrund der unterschiedlichen Technologien nicht davon ausgehen, dass alle Hersteller die „gleiche technische Sprache“ sprechen. Deshalb ist es von Vorteil, wenn alle benötigten Informationen in einer allgemeinen Form vorliegen. Dazu dient ein semantisches Modell, welches die Informationen in einer allgemeinen, maschinenlesbaren Sprache beschreibt. Das Modell erleichtert dann den automatischen Informationsaustausch auch zwischen Geräten, die unterschiedliche Protokolle unterstützen. Ein semantisches Modell ist

somit ein wesentlicher Bestandteil der Geräteverwaltung. Die Beschreibung der Geräte und Netzkomponenten in einer allgemeinen und maschinenlesbaren Form kann als gemeinsame Sprache gesehen werden, welche die Integration von unterschiedlichen Technologien unterstützt. Zusätzlich kann ein semantisches Modell dem Aufbau einer Wissens Basis dienen, mit deren Hilfe neue Anbieter leichter in der Lage sind, eigene Lösungen entwickeln. Dies erhöht die Qualität und senkt die Preise, was wichtig für den großen Durchbruch im Heimbereich ist.

2.2 Begriffsklärung Remote Konfigurationsmanagement

Unter Remote Konfigurationsmanagement von Haushaltsgeräten wird in dieser Diplomarbeit die Möglichkeit verstanden, diese Geräte entfernt, also über ein Netzwerk, zu warten. Dazu gehört die Verwaltung von Geräteinformationen an einer zentralen Stelle. Unterschiedliche Informationen über ein Gerät müssen bekannt sein. Beispielsweise die Art des Gerätes, der Hersteller und das Modell, das Baujahr und die Dauer der Garantie. Aber auch Informationen über den Zustand des Gerätes wie zum Beispiel der aktuelle Softwarestand und die auf dem Gerät laufenden Dienste. Wenn die letzte Information nicht bekannt ist, bleibt im Falle eines Fehlers nur das Zurücksetzen des Gerätes in den Ursprungszustand. Alle später vorgenommenen Änderungen wie etwa ein Zusatzprogramm zur Nutzung des Nachtstroms, gehen verloren.

Geräte zu warten bedeutet, sich über den aktuellen Status zu informieren und gegebenenfalls neue Software aufzuspielen. Dafür müssen die Geräte im Netzwerk kommunizieren, Auskunft über ihren Zustand geben können und die Möglichkeit eines Remote Software Updates bieten, also die Möglichkeit, die Software entfernt über ein Netzwerk auf das Gerät zu spielen. Ein Gerät soll beispielsweise Auskunft über seine Seriennummer, sein Modell und die aktuelle Software geben können. Diese Informationen können dann mit Informationen abgeglichen werden, die von einem Service Provider verwaltet werden. Dieser ermittelt zur aktuellen Software kompatible neue Softwaredienste. Dafür ist es allerdings notwendig, dass die beteiligten Geräte über eine gewisse minimale Grundfunktionalität verfügen, die es ihnen ermöglicht in einem Netzwerk zu kommunizieren und Remote Software Updates durchzuführen.

Dieses Verständnis des Remote Konfigurationsmanagements, erinnert an die Grundzüge der klassischen Softwareverteilung. Auch dort will man Software über ein Netzwerk auf Geräte aufspielen, und dies mit so wenig Benutzerinteraktion wie möglich.

Bezug zur klassischen Softwareverteilung

Bei der Softwareverteilung geht es darum Software (Betriebssystem und Anwendungen) auf Anwender PCs zu installieren und zu konfigurieren. Dies muss automatisiert geschehen und ohne dass die Benutzer in den Prozess mit einbezogen werden. Man geht davon aus, dass die Installation und Konfiguration von PCs den Anwender überfordert und zudem Sicherheitsprobleme mit sich bringt. Deshalb wird der PC des Anwenders von einem Experten vorbereitet. Da dieser oft sehr viele PCs (10.000!) einrichten muss, sollte dies automatisch und entfernt geschehen. Ziel ist damit eine automatische Installation, Konfiguration und Wartung einer großen Anzahl von Computer mit minimalem Aufwand und ein störungsarmer und sicherer Betrieb der betreuten Umgebung. [2.04]

Die PCs müssen über eine Netzwerkkarte am Netz angeschlossen sein und über eine minimale Funktionalität verfügen. Oft reicht es schon aus, wenn sie ein Bootprotokoll wie BOOTP oder PXE verstehen. Über dieses können sie von einem Bootserver automatisch gestartet werden, bevor die eigentlichen Softwarepakete installiert werden. Außer den Strom einzuschalten muss der Benutzer nichts machen. Wenn die Netzwerkkarte es unterstützt, kann sogar das Einschalten automatisch geschehen.

Bei Haushaltsgeräten hat man ähnliche Probleme. Es handelt sich um sehr viele Geräte, die fernwartbar sein sollen. Der Besitzer verfügt nicht über das Wissen, die Geräte zu warten. Und er will sich auch nicht mit solchen technischen Details belasten. Die Konfiguration soll also ebenfalls automatisch geschehen, und die Geräte müssen ein Software Update unterstützen.

Es gibt allerdings auch große Unterschiede zwischen der Softwareverteilung und dem Remote Konfigurationsmanagement. Zum einen hat ein Software-Verteilungssystem ausreichende Rechte im Netz. Es handelt sich in der Regel um ein privates Firmen LAN.⁵

⁵ LAN: Local Area Network.

Bei der Wartung eines Haushaltsgerätes befindet man sich in der Regel nicht in seinem eigenen LAN. Man will ein Gerät konfigurieren, das sich in einem fremden Netz, dem Netz des Benutzers, befindet, welches vor unbefugtem Zugriff zu schützen ist. Dies stellt große Anforderungen an die Sicherheit.

Dann ist die Topologie des Netzes nicht bekannt. Es muss also Mechanismen geben, um herauszufinden, wo sich die zu wartenden Geräte befinden, und welche Dienste sie unterstützen. Eine Lösung dieses Problems ist der Einsatz von Self-Organizing Network Komponenten, durch den Geräte und Dienste selbstständig gefunden werden.⁶

Weiter ist die klassische Softwareverteilung auf PCs ausgelegt, die in höchstem Maße standardisiert sind. Für Haushaltsgeräte fehlen die Standards bislang. Es ist bisher noch völlig unklar, wie eine Waschmaschine oder eine Gefriertruhe mit ihrer Netzumgebung kommuniziert. Das Finden von gemeinsamen Standards und Protokollen ist deshalb ein großer Wunsch vieler Hersteller. Es gibt in der Tat einige Bestrebungen in diese Richtung, von denen einige in Kapitel 3 aufgezeigt werden.

⁶ Self-Organizing Networks sind Netze, in denen die Netztopologie nicht zentral verwaltet wird. Dies erfordert eine höhere Intelligenz der Geräte, die Mechanismen zum Service Discovery benötigen

3 IT Landschaft heutiger Heimnetze

Die wachsende Vernetzung im Heimbereich mit ihrem Anstieg netzwerkfähiger Haushaltsgeräte führt zu einer großen Anzahl unterschiedlicher Protokolle und Technologien. Heimsysteme werden immer heterogener und komplexer, was dem großen Durchbruch in diesem Bereich eher hinderlich ist. Der Kunde muss sich bei der Anschaffung zwischen vielen, untereinander oft inkompatiblen Systemen entscheiden. Neben der Art des Internetzugangs wie Modem, ISDN oder DSL und den physikalischen Übertragungsmedien Kabel, Luft und Strom stehen unterschiedliche Hausbussysteme für Licht, Heiz- und Sicherheits-Technik zur Auswahl. Auch auf der Anwendungsebene gibt es unterschiedliche Technologien für die Kommunikation zwischen den Geräten, wie beispielsweise JINI oder UPnP. Eine noch größere Auswahl herrscht im Bereich der Geräteanschlüsse. Firewire, USB, Infrarot und Bluetooth sind nur einige Namen, die hier eine Rolle spielen.

Anwendungsgebiete im Heimnetzwerk

Die Anwendungsbereiche im Heimbereich sind vielfältig. Man kann grob zwischen den vier folgenden Feldern unterscheiden. [3.01]

- **Information und Kommunikation**
Für Recherche, E-Commerce, Bankgeschäfte, Heimarbeit und Kommunikation genügt ein PC samt Internetanschluss.
- **Unterhaltung**
Mit digital vernetzten Audio und Video Geräten, die über Kabel oder Satellit an einen Service Provider angeschlossen sind, ist es möglich, Filme und Musik herunterzuladen oder an Online Spielen teilzunehmen.
- **Sicherheit**
Neben der Hausüberwachung und der Einbruchmeldung geht es hier hauptsächlich um die Geräteüberwachung. Geräte werden über Sensoren überwacht, die über das Telefonnetz oder entsprechende Funktechnologie kommunizieren. Ferner ist die medizinische Überwachung oder die Kontrolle des Fernsehprogramms durch die Eltern denkbar.

- Komfort
Die Kontrolle von Haushaltsgeräten wie Kühlschrank, Herd und Waschmaschine soll vor allem eine Zeitersparnis bei der täglichen Hausarbeit bringen.

Für diese Anwendungsbereiche gibt es eine Vielzahl möglicher technischer Umsetzungen. Diese Übersicht soll etwas Licht in das Wirrwarr der Namen und Abkürzungen bringen, die in einem heutigen Heimnetz anzutreffen sind. Abbildung 2 zeigt einige der möglichen Technologien.

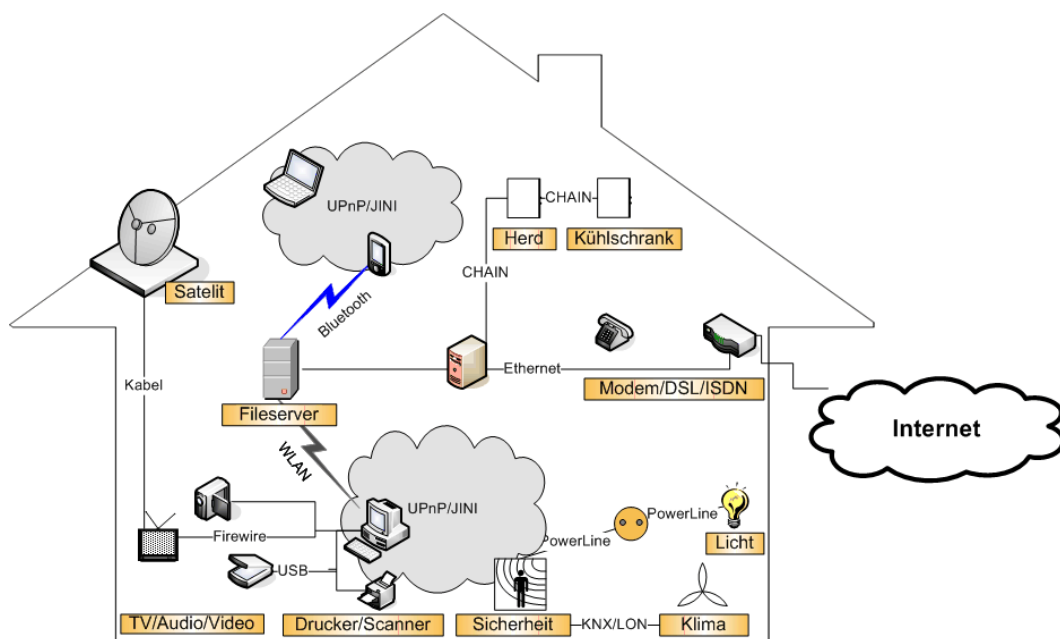


Abbildung 2 Technologien und Anwendungsbereiche im Heimnetz

Die Technologien lassen sich in die Bereiche Zugang, Geräteanschluss, Übertragung, Kommunikationsprotokolle und Hausbussysteme unterteilen. Eine kleine Übersicht soll ein allgemeines Verständnis über Eigenschaften und Einsatz der jeweiligen Technologie vermitteln.

Zugangstechnologien

Die Verbindung zur Außenwelt ist inzwischen fast überall gleich. Für den Anschluss an das Internet gewinnt neben analogen 56k Modems und digitalen ISDN Zugängen zunehmend der Breitbandzugang über DSL an Bedeutung. Eine Alternative stellt zudem der Internetzugang über das TV-Kabel dar, der allerdings nicht flächendeckend verfügbar

ist. Die Internetanbindung über das Stromnetz spielt dagegen kaum noch eine Rolle.

3.1 Anschlusstechnologien

3.1.1 IrDa

Die Infrared Data Association (IrDa) hat im März 1994 einen Funkstandard zur Datenkommunikation veröffentlicht. Der Standard definiert neben den Übertragungsprotokollen auch die Geräte. Hauptsächlich wird IrDa eingesetzt um verschiedene mobile Geräte zu verbinden. Es werden Datenraten von bis zu 115,2kbit/s erreicht, neuere Generationen sogar bis zu 4Mbit/s. Die Reichweite ist mit 1m begrenzt und für eine Verbindung ist Sichtkontakt erforderlich. [3.02]

Zusammenfassung der Merkmale

- Drahtlos
- Sehr kurze Distanzen (1m)
- Sicht notwendig
- Billig

3.1.2 Bluetooth

Bluetooth ist eine Funktechnologie, um drahtlose Geräte auf kurze Distanz zu vernetzen. Die Entwicklung wurde bereits 1994 von der Firma Ericsson initiiert, die einen Weg suchte, Kabel zwischen Telefonen und Zusatzgeräten zu ersetzen. Zusammen mit weiteren namhaften Herstellern wie Nokia, IBM, Motorola und Microsoft entstand 1998 die Bluetooth Special Interest Group (BSIG), die 1999 die erste Version der Bluetooth Spezifikation veröffentlichte. Heute zählt die BSIG mehr als 3200 Mitglieder, die meisten davon als Lizenznehmer. [3.03] [3.04]

Der Einsatz von Bluetooth ist äußerst vielfältig, da Bluetooth sowohl Sprach- als auch Datenübertragung unterstützt. So kann man einen PC mit Eingabegeräten wie Maus oder Tastatur verbinden, oder über ein drahtlos an ein Mobiltelefon angeschlossenes Headset telefonieren. Bluetooth unterstützt Transferraten von bis zu 721kBit/s. Hinsichtlich der Sendeleistung und der daraus resultierenden Reichweite unterscheidet man die drei folgenden Klassen.

Klasse	Sendeleistung	Reichweite
1	1 mW	3m
2	2,5 mW	10m
3	100 mW	100m

Tabelle 1 Bluetooth Reichweite

Bluetooth ist kostengünstig, energiesparend und benutzerfreundlich. Ebenso wie WLAN nutzt Bluetooth das kostenfreie ISM-Frequenzband von 2,4 GHz für die Übertragung, so dass keine Lizenzkosten entstehen.⁷ Um Konflikte mit anderen Geräten zu vermeiden, setzt Bluetooth ein Frequenzsprungverfahren ein, bei dem die Frequenz 1.600-mal pro Sekunde innerhalb von 79 Kanälen gewechselt wird.

Der niedrige Energieverbrauch wird durch einen effizienten Energiemechanismus erreicht, der die Stärke der Funkübertragung des Senders an die tatsächlichen Erfordernisse anpasst. Umgekehrt teilt es ein Empfänger dem Sender mit, wenn er ein übermäßig starkes Signal von diesem bekommt. Der Sender kann dann seine Signalstärke senken. [3.06]

Benutzerfreundlich ist Bluetooth in mehrerlei Hinsicht. Dass Bluetooth ohne Kabel auskommt, liegt natürlich in der Natur der Sache. Im Gegensatz zu Infrarot ist aber zusätzlich keine Sichtverbindung zwischen den Geräten erforderlich, da Funkwellen im Bereich von 2,4 GHz weder vor Jackentaschen noch vor Türen oder Wänden halt machen. Man kann das Mobiltelefon also in der Tasche lassen, wenn man mit einem Bluetooth Headset telefonieren will.

Schließlich muss der Nutzer relativ wenige Einstellungen vornehmen, bevor eine Verbindung zustande kommen kann. Die Geräte erkennen sich gegenseitig, ohne dass sie dafür konfiguriert werden müssen. Ein Bluetooth Telefon erkennt ein Headset z. B. automatisch als solches. Dies wird durch sog. Profile erreicht. Diese Profile legen auch fest, welche Bluetooth Protokolle von welchen Geräteklassen implementiert werden müssen. Im Unterschied zu anderen Funktechnologien wie Wireless LAN definiert Bluetooth nämlich einen ganzen Satz von

⁷ ISM: Die Industrial, Scientific and Medical Frequenzbänder sind für die nicht kommerzielle und lizenzfreie Nutzung reserviert. Durch die Nutzung entstehen keinerlei Kosten. Die ISM Bänder werden von einer Vielzahl von Geräten wie beispielsweise Mikrowellen, schnurlosen Telefonen, Fernbedienungen, Garagenöffnern oder Funk-Kopfhörern genutzt, wodurch es immer häufiger zu Störungen kommt. [3.05]

Protokollen, die das komplette OSI Modell abdecken, von der Bitübertragungsschicht bis hinauf zur Anwendungsschicht.⁸ Nicht alle Geräte müssen alle Protokolle umsetzen.

Bluetooth Geräte sind in der Lage, spontan und selbstständig Netze zu bilden. Ein sogenanntes Piconet besteht aus bis zu 8 aktiven Geräten, von denen eines die Rolle des Masters übernimmt, und die übrigen Slaves sind. Bis zu 255 zusätzliche Geräte können sich in einem inaktiven Parkmodus befinden. Der Master ist dasjenige Gerät, das die Verbindung initiiert. Er hat die Aufgabe, die Geräte zu synchronisieren, was aufgrund des Frequenzsprungverfahrens erforderlich ist. Ein PC, der über Bluetooth mit einer Tastatur, einer Maus und einem Drucker verbunden ist, wäre beispielsweise der Master. Ein Master kann selbst Slave in einem weiteren Piconet sein, dann spricht man von einem Scatternetz.

Zusammenfassung der Merkmale

- Drahtlos
- Keine Sicht erforderlich
- Kostengünstig durch ISM Band Nutzung
- Energiesparend
- Reichweiten von 3m bis 100m
- Kaum Konfiguration erforderlich
- Sicher

3.1.3 Firewire

Firewire ist Apples Name für den schnellen Kommunikationsstandard IEEE1394.⁹ Dieser wird meist für die Datenübertragung zwischen Multimediageräten eingesetzt, z. B. um Videodaten zwischen Camcorder und PC zu übertragen. Firewire ist wie USB Hot Plug-fähig, d. h. es können Geräte bei laufendem Betrieb angeschlossen werden, was früher nicht selbstverständlich war. Dank der hohen Bandbreite von bis zu 400Mbit/s lässt sich Firewire aber auch als

⁸ OSI steht für Open Systems Interconnection. Das OSI Modell beschreibt den vollständigen Weg von Daten zwischen zwei Anwendungen. Die Kommunikation wird in 7 Schichten unterteilt, die ihre jeweilige Aufgabe unabhängig von den anderen erledigen können. Obwohl schon seit den 70ern entwickelt, dient das Modell auch heute noch als Basis für viele Netzprotokolle.

⁹ IEEE: Das Institute of Electrical and Electronics Engineers ist ein 1969 aus zwei Organisationen hervorgegangener Berufsverband von Ingenieuren, der Gremien für die Standardisierung von Technologien, Hardware und Software bildet. [3.07]

Alternative zu Ethernet nutzen. [3.08] Eine Weiterentwicklung erhöht die Übertragungsrates erst auf 800Mbit/s, später dann sogar auf 1.600 und 3.200Mbit/s. Zudem können neben Shielded Twisted Pair (STP) Kabeln auch alternative Übertragungsmedien wie Unshielded Twisted Pair (UTP) Kabel oder Glasfaser eingesetzt werden. Über einen Protocol Adaption Layer (PAL) ist es auch möglich, den Funkstandard IEEE 802.15.3 umzusetzen und damit ganz auf Kabel zu verzichten. Man spricht dann von Wireless Firewire.

Zusammenfassung der Merkmale

- Sehr schnell
- Hot Plug fähig
- Integrierte Stromversorgung der Geräte möglich
- STP Kabel mit 4 oder 6 Adern, zudem UTP, Glasfaser und Funk
- Max. Entfernung von 4,5 m zwischen den Geräten (bei 400Mbit/s)

3.2 Übertragungstechnologien

Übertragungstechnologien sind den ersten beiden OSI Schichten zuzuordnen. Diese umfassen die Bitübertragungsschicht, die das eigentliche Übertragungsmedium definiert und die Sicherungsschicht, in welcher der Medienzugriff geregelt wird.

3.2.1 Powerline

Bei Powerline Communications (PLC) handelt es sich um eine Technik, bei der Daten über das Stromnetz übertragen werden. Die Technik an sich ist schon mehrere Jahrzehnte alt und wird immer noch in vielen Bereichen eingesetzt, z. B. bei der Rundfunkübertragung oder zur Fernsteuerung von Straßenlaternen. Auch kennen viele sicherlich das Babyfon.

Im Heimbereich unterscheidet man heute hauptsächlich zwischen den beiden PLC Disziplinen Internetzugang und Inhouse Vernetzung. Die Übertragungsrates des Internetzugangs liegen derzeit bei 14Mbit/s zwischen der Trafostation und den Anschlüssen. Diesen Zugang teilen sich bis zu 150 Teilnehmer. Durch den Einsatz von Multiplexern kann dies aber abgefangen werden. Ein Vorteil ist, dass man immer online ist. Allerdings verliert der Internetzugang über das Stromnetz seit

seiner Markteinführung im Jahre 2001 zunehmend an Bedeutung.
[3.09]

Etwas anders sieht es bei der Inhouse Vernetzung aus. PLC kann zur Vernetzung von intelligenten Geräten wie PCs eingesetzt werden. Ebenso können weitere intelligente Geräte wie Waschmaschinen oder Kühlschränke vernetzt werden. Im Unterschied zu Ethernet Systemen sind keine zusätzlichen Kabel notwendig. Jede Steckdose ist sozusagen die Schnittstelle ins Datennetz. Zwar bietet diesen Vorteil auch WLAN, allerdings ist die Funktechnik nicht in Gebäuden geeignet, in denen Stahlbeton in den Wänden verbaut wurde.

Unter dem Namen HomePlug gibt es einen Standard für die Inhouse Vernetzung über das Stromnetz. [3.10] Es ist theoretisch möglich Daten mit 14Mbit/s zu übertragen. Ein Nachteil ist die hohe Abstrahlung, da Stromkabel keine Abschirmung für die hohen PCL Frequenzen von 1 bis 30MHz besitzen. Die Kabel werden somit zu Antennen, die eine Reichweite von bis zu 100km! haben. Die starke Abstrahlung kann insbesondere den Polizei-, Militär- und Amateurfunk stören. Wegen der großen Reichweite sind PCL Systeme auch nicht abhörsicher. Zwar verschlüsseln HomePlug konforme Systeme die Daten, allerdings nur mit 56bit. Die Erfahrungen mit WLAN haben gezeigt, dass dies keinen wirklichen Schutz gegen Hacker bietet. [3.11] Kaum eine Rolle spielt die Abstrahlung bei der Nutzung von Powerline in der Gebäudeautomation, da zur Übertragung der Steuersignale nur geringe Datenmengen übertragen werden.

Zusammenfassung der Merkmale

- Keine zusätzliche Verkabelung notwendig.
- Hoher Frequenzbereich von 1 bis 30 MHz
- Datenraten: theoretisch 14Mbit/s effektive 5 bis 10Mbit/s
- Distanz bis zu 200m
- Starke Abstrahlung bis 100km
- 56bit Verschlüsselung nicht abhörsicher
- Große Auswahl an USB und Ethernet Adaptern

3.2.2 IEEE 802.11 / WLAN

IEEE 802.11, besser bekannt als Wireless Lan oder WLAN, ist ein Standard für den Aufbau von drahtlosen Datennetzen. Die erste

Version wurde 1997 von der IEEE veröffentlicht. Ursprünglich war WLAN als reine Unternehmenstechnik konzipiert. Die erste WLAN Version war sehr teuer, die Leistung mit Übertragungsraten von lediglich 2Mbit/s dafür ziemlich dürftig. Inzwischen erlebt die Technik aber einen beispiellosen Siegeszug. Keine andere Netztechnik wurde ähnlich schnell verbreitet. Durch einen massiven Preisverfall und der starken Verbreitung im Heimbereich wurde die Technik auch für ein kommerzielles Umfeld wieder interessant. [3.12]

Die ursprüngliche Version nutzt ebenso wie Bluetooth das lizenzfreie ISM-Frequenzband von 2,4 GHz. Mit Übertragungsraten von 1-2Mbit/s ist sie heute allerdings nicht mehr verbreitet. Inzwischen gibt es mehrere Protokollerweiterungen. Die am weitesten verbreitete Version IEEE 802.11b arbeitet ebenfalls im Bereich von 2,4 GHz und erreicht Datenraten von 11Mbit/s. Bei Apple läuft diese Version unter dem Namen AirPort. IEEE 802.11a erreicht sogar 54Mbit/s, nutzt aber das 5 GHz Band, welches in Deutschland nicht lizenzfrei ist, was den Einsatz stark einschränkt.

Im Prinzip sind WLANs verteilte Systeme, die sich aus einzelnen Zellen zusammensetzen, sog. Basic Service Sets (BSS). Diese werden von einem Access Point (AP) verwaltet. Der AP kann über ein Backbone mit anderen APs zu einem Extended Service Set (EES) erweitert werden.¹⁰ Für Protokolle der höheren Schichten befinden sich aber alle Geräte eines solchen verteilten Systems in einem Netz. In einem Ad-Hoc Modus können Geräte aber auch direkt miteinander kommunizieren. Abb. 3 zeigt eine typische WLAN Infrastruktur.

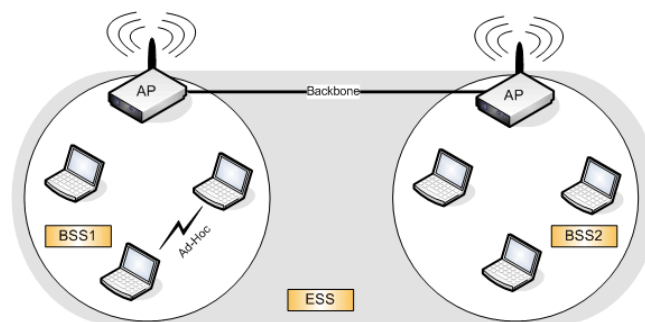


Abbildung 3 WLAN Netzstruktur

¹⁰ Als Backbone bezeichnet man in diesem Kontext einen breitbandigen Teil des Netzes, der die WLAN Teilnetze verbindet.

WLAN deckt die Medium Access Layer (MAC) und die Bitübertragungsschicht des OSI Referenzmodells ab. Die MAC Schicht arbeitet ähnlich wie Ethernet, musste allerdings an die Erfordernisse von Funknetzen angepasst werden. Das Zugriffsverfahren CSMA/CD wurde durch CSMA/CA ersetzt.¹¹ Dafür gibt es zwei Gründe. Zum einen können Geräte in Funknetzen nicht sicher sagen, ob es eine Kollision bei der Datenübertragung gab, weil nicht immer alle Geräte von allen Geräten erreicht werden können. Man spricht dabei von Hidden Nodes oder Hidden Terminals. Zum anderen ist für CSMA/CD eine Duplex Übertragung erforderlich, also die Möglichkeit gleichzeitig zu senden und zu empfangen. Diese Technik hätte WLAN aber zu teuer gemacht. WLAN Adapter arbeiten deshalb nur im Halbduplex Modus.

Typischerweise sind in WLANs Entfernungen von etwa 100m zum AP möglich. Allerdings verringern große Entfernungen den effektiven Datendurchsatz des Gesamtsystems, weil die Übertragungszeiten länger werden und Geräte länger auf Empfangsbestätigungen warten müssen. [3.13] Die Reichweite bringt zudem ein großes Sicherheitsproblem mit sich. WLANs können leicht abgehört werden. Die ursprünglichen im Standard vorgesehenen Sicherheitsmechanismen bieten keinen wirklichen Schutz. Es wird daher empfohlen, auf den höheren OSI-Schichten zu verschlüsseln, also beispielsweise mit SSL.

Zusammenfassung der Merkmale

- Drahtlos
- Datenraten von 2, 11, 56Mbit/s
- Halbduplex
- CMSA/CA
- Teilweise lizenzfrei
- Reichweite:100m-300m mit ungerichteten Antennen
- Nicht geeignet für Bauten mit Stahlbeton

¹¹ CSMA/CD: Carrier Sense Multiple Access / Collision Avoidance ist ein in Funknetzen verbreitetes Verfahren für die Mediengriffsteuerung, bei dem versucht wird Kollisionen auf dem Übertragungsmedium zu vermeiden. Im Gegensatz dazu nimmt CSMA/CD Kollisionen in Kauf. Sie werden jedoch während der Datenübertragung erkannt (CA=Collision Detect), so dass die Daten erneut gesendet werden können.

3.3 Hausbussysteme

Einen wesentlichen Schritt in der Entwicklung von elektronischen Systemen für die Gebäudeautomation stellt die Einführung von Hausbus- bzw. Feldbus-Systemen dar. In einem solchen System wird eine große Anzahl sogenannter Sensoren und Aktoren über einen Bus miteinander verbunden. Während Sensoren die physikalischen und chemischen Eigenschaften der Umgebung überwachen, wandeln Aktoren empfangene Signale dann in Arbeit um. Beispielsweise erkennt ein Bewegungssensor eine Bewegung und sendet ein Signal, welches dann einen Aktor dazu veranlasst das Licht einzuschalten. Feldbusse werden in der Gebäudeautomation hauptsächlich für einfache Steueraufgaben eingesetzt. Der Anwendungsbereich in der Gebäudeautomation ist vielfältig. Der Einsatz reicht von der Licht-, Klima-, Heizung-, und Sicherheitstechnik über Aufzugsteuerung und Zutrittskontrolle bis hin zur Kontrolle von Feuersignalen. In der professionellen Gebäudetechnik werden Hausbussysteme seit langem eingesetzt. Im privaten Wohnungsbau hat sich dies bisher nicht gelohnt, da die Preise zu hoch waren. Zudem hat die Vielzahl an unterschiedlichen Spezifikationen eine breite Marktakzeptanz und das Wachstum im Heimbereich verhindert.

Die beiden ersten und wichtigsten Feldbus-Systeme wurden etwa zeitgleich entwickelt. Das Local Operation Network (LON) wurde von der kleinen US-amerikanischen Firma Echelon Corporation entwickelt, während in Europa der Europäische Installationsbus (EIB) unter Federführung von Siemens entstand. Folgende Liste zeigt einige der heute anzutreffenden Systeme.

- LON – LonWorks Local Operating Network von Echelon [3.14]
- EIB – European Installation Bus von EIBA [3.15]
- EHS – European Home System von EHS [3.16]
- BatiBUS von BCI [3.17]
- KNX – Konnex von Konnex Association [3.18]
- CEBus von CIC CEBus Industry Council [3.19]
- LCN – Local Control Network von Isendorf [3.20]

Während sich in der PC-Kommunikation mit Ethernet längst ein Kommunikationsstandard durchgesetzt hat, kann man dies in der Haustechnik leider noch nicht beobachten. Es existieren viele Bussysteme nebeneinander. Einige Firmen versuchen jedoch gemeinsame Standards zu schaffen. Die drei Organisationen BCI,

EHSA und EIBA haben sich inzwischen in der Konnex Association zusammengefunden und entwickeln den neuen gemeinsamen Standard KNX. EIB floss größtenteils unverändert in die Entwicklung mit ein und macht den Hauptbestandteil von KNX aus. Doch obwohl sich KNX als weltweit ersten offenen Standard für Hausbussysteme präsentiert sind allein für die Spezifikationen schon eine Gebühr von 1000 Euro zu zahlen. [3.21]

Technisch war die Entwicklung ein großer Fortschritt. Im Prinzip ermöglicht es ein Hausbussystem intelligenten Geräten miteinander zu kommunizieren und auf gemeinsame Daten zuzugreifen. Bis dahin wurden Befehle von einem zentralen Gerät ausgegeben. Ein Feldbus arbeitet dabei ähnlich wie Ethernet, allerdings kommt er mit einer wesentlich geringeren Bandbreite aus.

3.3.1 LON

Das Local Operating Network (LON) ist ein Feldbussystem, dass über eine dezentrale, flache Topologie verfügt. Für die Steuerung ist keine zentrale Instanz zuständig. Vielmehr übernehmen die Netzteilnehmer die Steuerung selbst und kommunizieren über Statusnachrichten, welche über das LON-eigene Kommunikationsprotokoll LonTalk übertragen werden. Neben Aktoren und Sensoren gibt es noch Controller, die jeweils für einen Bereich des Netzes verantwortlich sind. Ein Controller sorgt dafür, dass Informationen wenn möglich sofort lokal verarbeitet werden. Somit kann LON für sehr umfangreiche Installationen eingesetzt werden. LON Geräte verfügen über mehrere CPUs, die den gesamten Funktionsumfang der 7 Schichten des OSI Modells abdecken. Der Zugriff erfolgt über ein CSMA/CA Verfahren. Die Geräte können mit Standardanwendungen ausgestattet werden, die von der LonMark Organisation in sogenannten Funktionsprofilen definiert werden. Anwendungen laufen bei LON Geräten auf einer eigenen CPU. LON ist offen, das heißt es können Geräte unterschiedlicher Hersteller eingesetzt werden. [3.22]

3.3.2 EIB und KNX

EIB ist ebenfalls ein Bussystem. Der Einsatz des EIB Bussystems ist auf Licht-, Heizung- und Sonnenschutztechnik beschränkt. Durch höhere Netzprotokolle wie BACNet und HVAC kann der Einsatz

jedoch zum Beispiel auf die Anwendungen der Sicherheitstechnik erweitert werden. Trotz des beschränkten Einsatzes im Vergleich zu LON ist EIB vor allem in Deutschland weit verbreitet. Dies liegt unter anderem daran, dass es Teil der Ausbildung von Elektroinstallateuren ist. [3.23] Neben Installationen in der Gebäudeautomation findet man EIB inzwischen auch im privaten Bereich des gehobenen Wohnungsbaus. Die EIBA (EIB-Association) legt die einheitliche EIB-Spezifikation fest. Inzwischen ist EIB zusammen mit den Systemen EHS und BatiBUS im KNX Standard aufgegangen. Da EIB als Hauptbestandteil von KNX nicht verändert wurde, können EIB Geräte in KNX Installationen weiter verwendet werden. Als Medien sind sowohl bei Konnex als auch bei LonWorks Kabel, Strom und Funk möglich. Die Systeme unterscheiden sich aber erheblich im Datendurchsatz.

3.4 Kommunikation auf der Anwendungsebene

3.4.1 UPnP

Das Universal Plug and Play (UPnP) der Firma Microsoft stellt eine Erweiterung des Plug and Play (PnP) Konzepts dar. PnP ist Microsofts Name für ein Konzept, bei dem neu an einen Computer angeschlossene Geräte automatisch erkannt werden, ohne dass dafür spezielle Treiber installiert oder sonstige Konfigurationen vorgenommen werden müssen. Beispielsweise wird ein USB-Stick zur Speicherung von Daten von einem Windows XP System sofort selbstständig erkannt und als eigenes Laufwerk angeboten. UPnP überträgt diese Eigenschaft nun auf ein Netzwerk. UPnP fähige Geräte wie Scanner oder Drucker werden im Netz automatisch gefunden. Dies beinhaltet auch Softwaredienste, die auf den Geräten verfügbar sind. Es genügt, ein Gerät an das Netz anzuschließen. Weder der Ort des neuen Gerätes noch seine Funktion müssen irgendwo eingestellt werden. Denkbar ist z. B. Papas neuer Drucker, der nach dem Anschluss auch auf den PCs der Kinder verfügbar ist. UPnP Komponenten sind des sog. Zero-Configuration Komponenten.

UPnP vereinfacht damit die Netzanbindung intelligenter Geräte und vermindert den Wartungsaufwand. Da für UPnP Geräte kein Administrator notwendig ist, eignet sich diese Technologie sehr gut für Heimnetze, bei deren Betreiber oft kein ausreichendes, technisches Wissen vorhanden ist.

UPnP ist eine Technologie der Anwendungsschicht des OSI Referenzmodells. Es hat eine offene Architektur und baut auf existierenden Protokollen und Industriestandards wie XML und SOAP auf.¹² Zudem wird es von sehr vielen Firmen unterstützt. Im UPnP Forum sind derzeit 732 Firmen vertreten. [3.24][3.25]

Technisch wird mithilfe der UPnP Protokolle ein Peer-to-Peer¹³ Netzwerk aufgebaut. Die Spezifikation definiert sowohl die Netzanbindung als auch die Steuerung der Geräte mittels über HTTP verschickter XML Nachrichten. Es gibt drei unterschiedliche Arten von UPnP Komponenten, die in Abbildung 4 dargestellt sind. Ein Dienst (*Service*) stellt die eigentliche Funktionalität über definierte Kommandos zur Verfügung. Ein Gerät (*Device*) kann als Container für Dienste gesehen werden. Zusätzlich kann es aber auch aus weiteren, eingebetteten Geräten und aus einem *Control Point* bestehen. Control Points sind die Clients in einem UPnP Netzwerk, welche die Geräte und Dienste steuern. Jedes Gerät wird eindeutig anhand einer UUID identifiziert.¹⁴

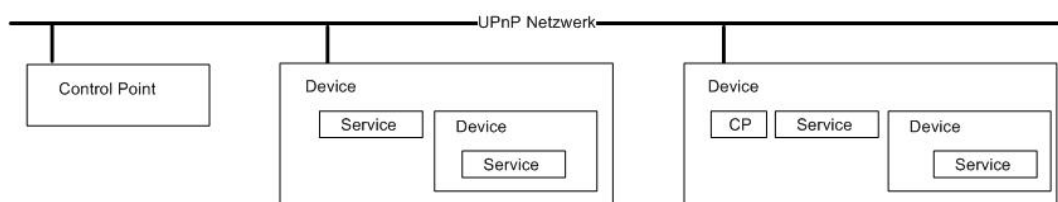


Abbildung 4 UPnP Netzstruktur

Die Kommunikation in einem UPnP Netzwerk lässt sich grob in die drei Bereiche *Location*, *Discovery* und *Control* einteilen. Zuerst muss ein Control Point das Gerät auffindig machen (*Control*). Dann findet er heraus, was das Gerät kann, und wie man es bedient (*Discovery*) um schließlich seine Dienste zu nutzen (*Control*).

¹² SOAP: Das Simple Object Access Protocol ist ein Nachrichtenprotokoll für die Kommunikation zwischen Anwendungen. Es verwendet meist HTTP als Transportprotokoll, wodurch die Kommunikation auch über Firewallgrenzen hinweg ermöglicht wird.

¹³ In einem Peer-To-Peer oder P2P Netzwerk sind alle Teilnehmer gleichberechtigt und können sowohl Dienste anbieten als auch nutzen. Im P2P Netz ist die aus dem Client-Server bekannte strikte Rollenverteilung somit aufgehoben.

¹⁴ UUID: Der Universal Unique Identifier ist eine global eindeutige 128-bit Adresse, die aus der Verbindung der eindeutigen Hardwareadresse der Netzwerkkarte mit einem Zeitstempel erzeugt wird.

Location

Sobald ein Gerät an ein Netz angeschlossen wird, schickt es eine Multicast Nachricht an die Control Points auf Basis des Simple Service Discovery Protocol (SSDP). Ebenso können Control Points nach Geräten suchen. Die Discovery Message enthält nur wenige Informationen wie Name und Gerätetyp. Vor allem aber enthält sie eine URL, unter der eine genaue Beschreibung des Gerätes zu finden ist.¹⁵

Discovery und Control

Sobald ein Control Point ein Gerät gefunden hat, liest er die Beschreibung des Gerätes. Diese liegt unter der vom Gerät angegebenen Adresse in Form eines XML Dokuments vor. Darin findet der Control Point Informationen über den Hersteller, die Seriennummer und vor allem Informationen zur Steuerung des Gerätes. Für jeden Dienst werden Kommandos samt Datentypen und einer Kommando URL beschrieben. Die Steuerung erfolgt anhand der Informationen aus der Beschreibungsdatei mittels Remote Procedure Calls (RPC) über SOAP.

Eventing

Für den Zugriff auf Geräte und deren Dienste ist es wichtig, sich über den aktuellen Zustand zu informieren. Ein Drucker sollte sich z. B. im Zustand *Bereit* befinden bevor man drucken kann. Ist er dagegen im Zustand *Druckend*, macht es keinen Sinn, ein Druckdokument zu schicken. Damit Control Points nicht ständig bei den Geräten nachfragen müssen, ob sich ihr Zustand geändert hat ist in UPnP ein Benachrichtigungsmechanismus eingebaut. Über diesen auf der General Event Notification Architecture (GENA) basierenden Mechanismus benachrichtigen Geräte die Control Points automatisch, sobald sich ihr Zustand ändert.

Standardgeräte

Damit UPnP Treiber bzw. Clients unabhängig von Geräteherstellern entwickelt werden können, werden in Arbeitsgruppen Standards für

¹⁵ URL: Der Uniform Resource Locator bezeichnet ein standardisiertes Adressierungsformat für Ressourcen in Computernetzwerken.

Geräteklassen definiert. Es gibt Standards für Drucker, Scanner, Media Server, Internet Gateways uvm.. Standards für Haushaltsgeräte wie Waschmaschinen und Kühlschränke sind in Arbeit aber noch nicht veröffentlicht. Derzeit gibt es ca. 100 bekannte und einige nicht bekannte, offiziell zertifizierte Geräte auf dem Markt. Bei diesen handelt es sich bisher fast ausschließlich um Internet Zugangsgeräte und AV Geräte. White Goods wie Kühlschränke oder Gefriertruhen sind noch nicht dabei.

Zusammenfassung der Merkmale

- Keine Konfiguration. Nachdem ein Gerät an das Netz angeschlossen wird, ist es sofort verfügbar.
- Basiert auf offenen Industriestandards wie TCP/IP, UDP, HTTP, XML und SOAP. Dadurch wird die Entwicklung neuer Geräte erheblich erleichtert.
- Plattformunabhängig. UPnP Geräte können in jeder Programmiersprache und für jedes Betriebssystem entwickelt werden.
- Unabhängig vom Übertragungsmedium. Einzige Voraussetzung ist ein vorhandener IP Protokollstack. Als Übertragungsmedium kommen u. a. Firewire, WLAN, Ethernet, Bluetooth, Powerline und Telefon in Frage.

Aufgrund der Benutzerfreundlichkeit von UPnP wird die Technologie bei der technischen Umsetzung des Remote Konfiguration Konzeptes eingesetzt.¹⁶

3.4.2 JINI

JINI wurde 1998 von der Firma SUN vorgestellt und steht für die Java Intelligent Network Infrastructure. Es definiert eine Architektur für den Bau verteilter Software Systeme. In einem verteilten System sind die Aufgaben auf unterschiedliche Netzwerkkomponenten verteilt. Einzelne Teilnehmer stellen ihre Funktionalität durch sogenannte Dienste dem Netzwerk zur Verfügung, was stark an UPnP erinnert. JINI will die Teilnahme an einem solchen System vereinfachen. Insbesondere sollen Dienste leicht und flexibel hinzugefügt oder entfernt werden können. Dadurch sind Benutzer des Netzwerks nicht an einen festen Ort gebunden, sondern können diesen leicht

¹⁶ Vgl. Kapitel 7.1

wechseln, was dem Netzwerk eine gewisse Dynamik gibt. Auch soll das Finden von Diensten erleichtert werden. In einem JINI System lassen sich folgende Rollen unterscheiden.

- *Service*
Ein Dienst ist eine Entität, die von einer Person, einem Programm oder von einem anderen Dienst benutzt werden kann. Nutzer von Diensten werden als Clients bezeichnet. Ein Dienst stellt eine bestimmte Funktionalität zur Verfügung. Dies kann ein Druck- oder Scandienst sein, eine mathematische Berechnung, ein Kommunikationskanal oder auch das aktuelle Wetter.
- *Lookup Service*
Der Lookup Service ist die zentrale Stelle zum Finden von Diensten in einem JINI System. Über ein Discovery Protokoll finden Dienste heraus, wo sich der Lookup Service befindet. Über das Join Protokoll melden sie sich dann an. Der Lookup Service mappt Schnittstellen mit Dienste Objekten, welche Informationen über den wirklichen Dienst beinhalten. Diese Proxy Objekte können aber auch Verweise auf weitere Lookup Dienste beinhalten, wodurch sich eine hierarchische Architektur umsetzen lässt.

JINI ist eine Java Technologie, die auf bestehenden Standards aufbaut. Für die Kommunikation wird hauptsächlich Remote Method Invocation (RMI) verwendet, eine Technik zum Aufruf entfernter Methoden. Da lediglich die Schnittstellen für die Kommunikation definiert werden können aber auch andere Middleware Technologien wie CORBA oder SOAP eingesetzt werden.¹⁷

Im Prinzip erweitert JINI die Java Anwendungsumgebung auf ein Netzwerk. Teilnehmer müssen dabei nicht unbedingt eine Java Virtual Machine (JVM) besitzen, diese kann durch einen Proxy ersetzt werden. Viele Ziele von JINI erinnern an Microsofts UPnP, das ebenfalls ein diensteorientiertes, dynamisches Netzwerk aufbaut und unabhängig von der Übertragung ist. Dass man heute kaum noch

¹⁷ Middleware bezeichnet den Teil komplexer verteilter Softwaresysteme, der die Kommunikation zwischen den unabhängigen Teilsystemen ermöglicht. Die Komplexität der zugrunde liegenden Infrastruktur wird dabei verborgen.

etwas von JINI hört, könnte an der fehlenden Unterstützung seitens Microsoft liegen. [3.26]

3.5 Neue Entwicklungen

3.5.1 CHAIN

Mit der CHAIN Plattform definiert der europäische Verband der Hausgerätehersteller (CECED), in dem die wichtigsten europäischen Gerätehersteller vertreten sind, ein Datenaustauschprotokoll für vernetzte Haushaltsgeräte. CHAIN steht für Ceced Home Appliances Interoperating Network und soll es Haushaltsgeräten wie Kühl- und Gefrierschränken, Waschmaschinen und Backöfen ermöglichen über eine offene, standardisierte Plattform zu kommunizieren. Ziel ist es den Datenaustausch von Geräten unterschiedlicher Hersteller zu vereinfachen. So soll die Fernsteuerung der Geräte, Lasten- und Energie-Management, Ferndiagnose und automatische Instandhaltung der Hausgeräte, sowie Downloads und Updates von Daten, Programmen und Internet Services möglich sein. Die Verbindung der Geräte erfolgt über Powerline, also über das Stromnetz und über Funk.

Leider gibt es bisher außer oberflächlichen Pressemitteilungen mit Funktionsbeschreibungen kaum öffentlich zugängliche Informationen über die CHAIN Architektur. Insbesondere wäre es interessant, wie Downloads und Updates von Programmen realisiert werden. [3.27]

3.5.2 Serve@Home

Das Home Automation System serve@home von Siemens ist eine Komplettlösung zur Integration von Geräten im Heimnetz. Haushaltsgeräte wie Gefriertruhe, Waschmaschine oder Kühlschrank sind über das Stromnetz mit einem Gateway verbunden. Gesteuert werden Sie über ein Tablet PC oder über ein Handy.

Das System ist offen, und es lassen sich auch Geräte anderer Hersteller steuern. Zur Kommunikation wird das Stromnetz nach dem EHS Standard genutzt. Alternativ ist aber auch Funk zum Anschluß der Geräte möglich. Als Kommunikationsprotokoll wird CHAIN (Ceced Home Appliances Interoperating Network) benutzt. Über einen

Ethernet Anschluß läßt sich das System zudem in bestehende EIB Systeme integrieren. [3.28]

3.6 Entwicklung von der Sicherheit zum Komfort

Wie bei der Beschreibung der Technologien zu sehen ist, gibt es große Bestrebungen zu gemeinsamen Standards zu kommen. Den Bedarf nach einer einheitlichen Kennzeichnung für die Automation gab es allerdings schon vor deren Einzug in die Heimnetze. Für den hochsensiblen Sicherheitsbereich von Kraftwerken wurde schon früh ein System geschaffen, mit dem Geräte eindeutig und durchgängig gekennzeichnet werden. Das Kraftwerks-Kennzeichnungssystem (KKS) ersetzt unterschiedliche, proprietäre Systeme durch ein einheitliches und durchgängiges System. Inzwischen ist das KKS ein auch international eingesetzter Quasi-Standard. Das System kann unabhängig vom Leitsystem, sozusagen herstellerunabhängig, eingesetzt werden. Die beteiligten Entitäten werden hierarchisch und funktionsbezogen strukturiert. [3.29] Über das System soll die räumliche und funktionsbezogene Position der Entitäten wie Anlagenteile und Betriebsmittel identifiziert werden. [3.30]

Aus der hochsensiblen Sicherheitsproblematik entstanden, hat sich das Konzept dann ebenfalls als sehr nützlich für die Gebäudetechnik erwiesen. Auch hier spielen Sicherheitsüberlegungen eine Rolle, wenn auch keine so entscheidende wie in Kernkraftwerken. Geräte, die in einem Gebäudenetz angeschlossen sind, müssen eindeutig identifizierbar sein. Es ist aus Sicherheitsgründen beispielsweise nicht uninteressant, an welchen Stellen im Gebäude Kameras angeschlossen werden.

Dem deutschen Institut für Normung (DIN) dient das KKS als Basis für ein neues und allgemeines Kennzeichnungssystem, das in der DIN Norm 6779 beschrieben ist. [3.31] Dem System liegt ebenfalls ein hierarchisches Strukturmodell zugrunde. Die Systematik ist eindeutig und durchgängig und dient als Basis für die Koordination verschiedener Bereiche in der Gebäudetechnik. Es handelt sich also um eine Art Sprache für die Abbildung von Daten und Informationen und zum Informationsaustausch zwischen verschiedenen Parteien.

Allmählich finden die Technologien der Gebäudetechnik ihren Weg in den Konsumer- und Komfortbereich. Hier ist die technologische

Vielfalt aber ungemein größer. Deshalb ist eine Sprache, die „jeder“ spricht, auch schwerer zu finden.

Die Entwicklung der Automation vom Sicherheitsbereich von Kraftwerken über die Gebäudeautomation bis hinein in den Komfortbereich von Heimnetzen erinnert an die Geschichte des Internets, dessen Entwicklung ebenfalls aus Sicherheitsüberlegungen begonnen wurde. Ursprünglich sollte das Internet, bzw. dessen Vorgänger, das Arpanet, die Kommunikation zwischen Militärbasen im Falle des Verlustes einzelner Anlagen ermöglichen. Aber auch in der privaten Heimvernetzung ist der Sicherheitsaspekt immer noch vorhanden, wenn auch nicht mehr so stark wie in der Kraftwerk- und Gebäudetechnik. Fällt beispielsweise die Gefriertruhe aus, sollte dies so schnell wie möglich erkannt, und entsprechend reagiert werden.

4 Reparatur Szenario

Wie wir gesehen haben führt, die wachsende Vernetzung im Heimbereich zu einer immer größeren Anzahl unterschiedlicher Protokolle und Technologien. Dem konnte bisher auch das Bestreben einiger Hersteller zu gemeinsamen Protokoll Standards nicht wirklich entgegen wirken.¹⁸ Wir brauchen also eine Lösung, die über gute Integrationsfähigkeiten über bestehende und mögliche neue Technologien verfügt. Diese Lösung verwendet folgende Schlüsselkomponenten:

- Selbstorganisation im Netz und Zero Konfiguration
- Unterstützung unterschiedlicher Netze und Transportprotokolle
- Vereinbarung eines Kennzeichnungssystems für Geräte und Rollen

Der erste Punkt wird durch den Einsatz von UPnP Komponenten erreicht. Als Lösung für die Unterstützung unterschiedlicher Netze und Protokolle bietet sich OSGi an.¹⁹ Der Weg zu einem einheitlichen Kennzeichnungssystem wird durch eine Ontologie erreicht.²⁰

Des Weiteren wird natürlich dem Bedarf entfernter Software Updates Rechnung getragen. Es soll also möglich sein, automatisch Software Updates auf Geräte zu laden. Der Nutzer steht dabei wenig in der Verantwortung. Deshalb werden Softwarestände und andere gerätespezifische Informationen nicht von ihm im Heimnetz, sondern an einer zentralen Stelle von einem Dienstleister verwaltet. Dieser sog. Repair Service Provider stellt auch die eigentliche Software für die Updates in einem Service Repository zur Verfügung. Sicherheitsaspekte wie der Schutz des Heimnetzes oder die Autorisierung eines Reparatur Mitarbeiters werden hier nicht genauer behandelt. Überlegungen dazu finden sich unter [4.01].

Das folgende Szenario soll den Einsatz und die Funktionalität der zu entwickelnden Service Anwendung verdeutlichen. Dabei wird ein

¹⁸ Vgl. Kapitel 3: KNX, CHAIN, etc.

¹⁹ Vgl. Kapitel 6 OSGi

²⁰ Vgl. Kapitel 5: Semantik

Haushaltsgerät über ein Gateway analysiert und mit neuer Software aktualisiert.

4.1 Beschreibung

Ein modernes Haushaltsgerät ist an das Heimnetz eines Nutzers angeschlossen. Der Besitzer des Haushaltsgerätes bemerkt einen Fehler und möchte das Gerät reparieren lassen. Bei dem zu wartenden Gerät handelt es sich in diesem Szenario um eine Waschmaschine, die an das private Heimnetz des Besitzers angeschlossen ist. Über einen einfachen Webbrowser kann dieser auf die Maschine zugreifen und bestimmte Basisdienste nutzen. Beispielsweise kann er die Startzeit einstellen oder prüfen, ob der Waschvorgang schon beendet ist. Sicher kann man den Sinn und Zweck der Fernsteuerung einer Waschmaschine in Frage stellen. Es ist aber durchaus denkbar, dass sich der Besitzer, der sich in seinem Arbeitszimmer im dritten Stock befindet, vergewissern will, ob die Wäsche im Keller schon fertig ist, ohne dass er dazu in den Keller will.

Nachdem das Gerät nicht mehr korrekt arbeitet, beauftragt der Besitzer eine Wartungsfirma, den sog. Repair Service Provider (RSP) mit der Reparatur des Gerätes. Diese entsendet einen Service Mitarbeiter, der im Szenario Repair Service Assistant (RSA) genannt wird zum Besitzer. Der RSA soll die eigentliche Reparatur durchführen. Unter anderem ist es seine Aufgabe, über das Gateway und das Heimnetz auf das zu wartende Gerät zuzugreifen und festzustellen, wo der Fehler liegt. Er liest bestimmte Geräteinformationen aus und stellt fest, dass ein Software Update notwendig ist. Der normale Besuch eines Service Mitarbeiters soll also auf die Anforderungen der Fernwartung inklusive eines Software Updates erweitert werden.

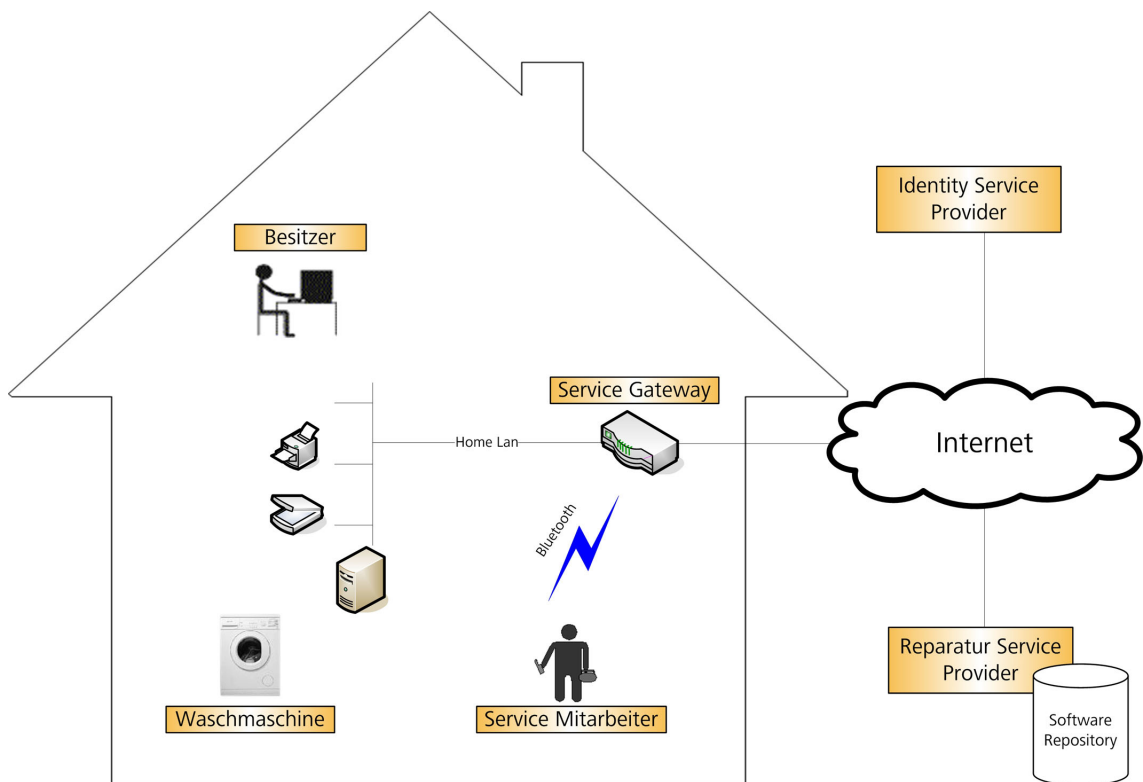


Abbildung 5 Beteiligte Entitäten des Reparatur Szenario

Als Zugangsg r t dient ihm sein Personal Data Assistant (PDA),  ber den er sich via Bluetooth mit dem Gateway verbindet. Das Gateway ist verantwortlich f r das Freischalten des zu wartenden Ger tes und den Schutz des  brigen Netzwerkes. Nachdem der RSA die entsprechenden Informationen, wie das Modell und den aktuellen Softwarestand, vom Ger t erhalten hat, verbindet er sich mit dem RSP, von dem er eine Liste mit passenden Updates erh lt. Dann initiiert er das Update, das vom zentralen Software Repository des RSP auf die Waschmaschine geladen wird. Nach erfolgreichem Update kann der Nutzer wieder auf das Ger t zugreifen. Abbildung 5 zeigt schematisch die beteiligten Personen und Entit ten.

4.2 Beteiligte Entit ten

4.2.1 Residential Service Gateway

Das Gateway spielt die zentrale Rolle der Kontrollstelle des Heimnetzes. Es ist verantwortlich f r die Kommunikation zwischen den unterschiedlichen Ger ten im Haus. Es vermittelt zwischen unterschiedlichen Teilnetzen und stellt den Zugriff auf die Dienste der

Geräte her. Auch verbindet es das Heimnetz mit dem Internet. Über das Gateway ist es möglich auf Geräte im Heimnetz von außerhalb zuzugreifen. Es schützt das Heimnetz vor dem Zugriff Unbefugter. In dem Szenario kommuniziert das Gateway mit der Waschmaschine über UPnP und bereitet die Dienste für den Web Client des Nutzers auf. Das Gateway benötigt dafür einen HTTP Server. Der Reparatur Mitarbeiter greift auf das Gateway mit einem PDA via Bluetooth (BT) zu. Zusätzlich verfügt das Gateway noch über einen permanenten Internetzugang. Abb. 6 zeigt die Kommunikationskomponenten des Gateways.

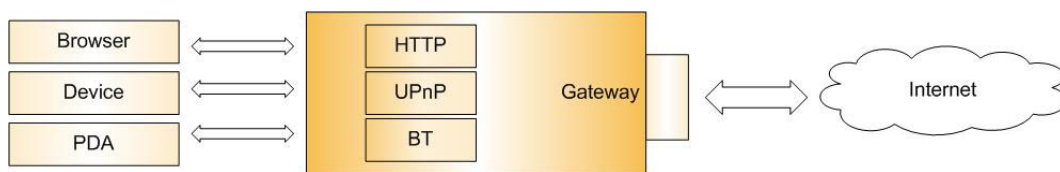


Abbildung 6 Schematischer Aufbau der Gateway Kommunikation

Der Schutz des Heimnetzes ist ein wesentlicher Bestandteil des Sicherheitskonzeptes. Das Gateway sollte nur jene Geräte für den RSA frei schalten, die dieser auch wirklich benötigt. Im konkreten Fall also nur die Waschmaschine. Eine Möglichkeit wäre es, dass der Besitzer die Waschmaschine manuell über seinen Client frei schaltet. Dies ist aber nicht wünschenswert, da der Besitzer nicht unnötig belastet werden sollte. Eine Alternative könnte der automatische Abgleich des Mitarbeiters mit den Geräten anhand einer Ontologie durch das Gateway sein.²¹ Ist in der Ontologie zum Beispiel festgelegt, dass ein Waschmaschinen Reparatur Mitarbeiter eine Waschmaschine reparieren kann, könnte das Gateway dies als ausreichend erachten, diesem Zugang zum Gerät zu gewähren. Der Mitarbeiter müsste sich gegenüber dem Gateway allerdings als Reparatur Mitarbeiter autorisieren.

4.2.2 Heimnetz

Das Heimnetz umfasst alle Netzkomponenten des Besitzers. Neben dem Gateway gehören dazu Geräte wie PC, Laptop, Fileserver, Drucker, Scanner, Audio und Video Komponenten, sowie sog. White Goods wie Gefriertruhe, Kühlschrank und natürlich die Waschmaschine.

²¹ Vgl. Kapitel 5: Semantik

4.2.3 Waschmaschine

Die Waschmaschine ist mittels Ethernet an das Heimnetz angeschlossen. Sie kommuniziert über das Universal Plug and Play (UPnP) Protokoll. Damit ist keine Installation notwendig. Nach der Verbindung mit dem Netzwerk werden die Maschine und all ihre Softwaredienste automatisch im Netzwerk bekannt gegeben. Die Waschmaschine stellt neben Diensten zur normalen Steuerung auch Informationen über sich zur Verfügung. Dazu gehört neben dem Modell und der Seriennummer auch ihr aktueller Softwarestand. Damit die Maschine Remote Software Update fähig ist, muss sie einen solchen Update Dienst anbieten.

4.2.4 Besitzer

Besitzer ist derjenige, dem das zu wartende Gerät gehört. Ihm gehört auch das Heimnetz, er hat also vollen Zugang auf alle Geräte. Dennoch sollte er so wenig wie möglich mit der Konfiguration der Komponenten belastet werden. Der Besitzer erteilt einen Reparaturauftrag an den RSP. Er will sicher sein, dass nur ein autorisierter Reparatur Mitarbeiter im Haus arbeitet und dass sein Heimnetz geschützt ist.

4.2.5 Reparatur Service Provider (RSP)

Der Reparatur Service Provider (RSP) erhält vom Besitzer den Auftrag für die Waschmaschinenreparatur. Er kennt Geräte unterschiedlicher Hersteller und kann anhand der Seriennummer und des Softwarestandes mögliche Updates ermitteln, die er über ein Software Repository anbietet. Weiter kennt der RSP Mitarbeiter, die den Reparaturauftrag ausführen können und delegiert den Auftrag an diese weiter.

4.2.6 Reparatur Service Mitarbeiter (RSA)

Der Service Mitarbeiter kommt zum Haus und weist sich dem Besitzer gegenüber aus. Er ist vom RSP autorisiert und muss dies dem Besitzer

gegenüber kenntlich machen. Er greift über das Gateway auf die Waschmaschine zu und analysiert sie. Danach verbindet er sich zum RSP und initiiert ein Software Update.

4.2.7 Identity Provider

Der Vollständigkeit halber soll noch der Identity Provider genannt werden, der Teil eines Sicherheitskonzeptes für das Szenario ist. Er ist ein externer Dienstleister, der die Aufgabe der Authentisierung des RSA übernimmt.

4.3 Ablauf

Der grobe Ablauf des Szenarios ist in Abbildung 7 dargestellt.

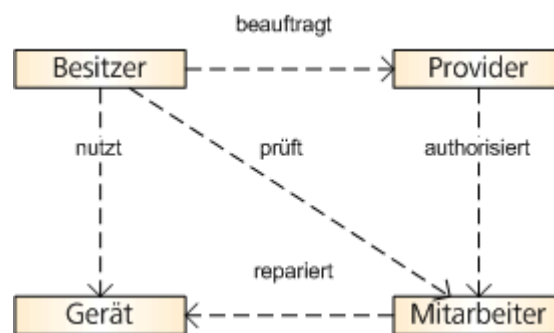


Abbildung 7 Ablauf des Szenarios

- Der Besitzer nutzt die Waschmaschine und stellt fest, dass sie nicht mehr funktioniert. Er erhält zum Beispiel eine Fehlermeldung.
- Der Besitzer teilt dies unter Angabe des Geräteherstellers und des Modells dem Service Provider mit.
- Der Service Provider ermittelt einen Service Mitarbeiter und schickt diesen zum Besitzer.
- Der Mitarbeiter weist sich beim Besitzer aus und repariert das Gerät.

Die Reparatur des Gerätes gliedert sich in folgende Schritte:

- Der Mitarbeiter verbindet sich mit dem Gateway via Bluetooth.
- Das Gateway prüft den Mitarbeiter anhand eines Credentials oder anhand eines Abgleichs der Kennzeichnung des Gerätes mit der

Kennzeichnung des Mitarbeiter und schaltet die Waschmaschine frei. Alternativ kann der Zugang auch manuell erfolgen.

- Der Mitarbeiter liest Informationen vom Gerät und fordert vom RSP eine Liste mit passender Software für das Gerät.
- Der Mitarbeiter initiiert das Update.
- Der Besitzer greift auf das Gerät zu.

Dieses Szenario dient der Darstellung des möglichen Einsatzes für ein entferntes Software Update. Natürlich könnte sich der Mitarbeiter auch direkt mit der Waschmaschine verbinden und die Software direkt aufspielen, zum Beispiel über ein serielles Kabel. Er benötigt aber eine Verbindung zum RSP für die Ermittlung der Updates. Ist an der Hardware nichts kaputt, muss der Mitarbeiter für ein einfaches Update gar nicht mehr ins Haus kommen, das Update könnte vollständig automatisch geschehen. Dies setzt allerdings ein starkes Sicherheitskonzept voraus.

Das Szenario lässt sich mit freien Softwaretools umsetzen. Eine Untersuchung von OSGi ergibt, dass es sich wegen seiner guten Integrationsfähigkeit gut als Basistechnologie für das Gateway eignet²². Wie in Kapitel 3.4.1 gezeigt wurde, bietet sich UPnP für die Kommunikation zwischen den Geräten an. Die Tatsache, dass sich Geräte automatisch finden, sobald sie an das Netz angeschlossen werden, macht es besonders benutzerfreundlich. Der Besitzer muss nichts konfigurieren.²³

Im letzten Kapitel wurden unterschiedliche Technologien beschrieben, sowie einige Bestrebungen für gemeinsame Standards. Das Bestreben der Hersteller, gemeinsame Standards im Heimbereich zu schaffen, kann durch eine gemeinsame Semantik unterstützt werden. In Zukunft werden semantische Modelle wohl stark an Bedeutung gewinnen. Deshalb soll vor der technischen Umsetzung ein semantisches Modell über das Service Szenario erstellt werden. Das Modell wird aber nicht zwingend in die spätere technische Umsetzung des Szenarios einfließen.

²² Vgl. Kapitel 6: OSGi

²³ Vgl. Kapitel 7.1: UPnP

5 Semantik

Weshalb brauchen wir für die Reparatur Service Anwendung ein semantisches Modell bzw. eine Ontologie? Um dies zu verstehen soll zuerst erläutert werden, was man unter semantischen Modellen versteht, und wie und wo sie eingesetzt werden. Danach wird eine konkrete Ontologie für die Service Anwendung entwickelt. Dies geschieht mit dem freien Entwicklungstool Protégé. Zum Schluss erfolgt ein Ausblick auf den möglichen Einsatz der Service Ontologie und wie sie in bestehende Modelle integriert werden kann.

5.1 Ontologie

Eine Ontologie in der Informatik ist eine Definition von Begriffen eines bestimmten Wissensgebietes, und wie diese miteinander in Beziehung stehen. Zusätzlich zu den Begriffen und ihren Relationen enthält die Ontologie Regeln über die Verwendung der definierten Begriffe. Eine Ontologie dient damit zur Repräsentation und zum Austausch von Wissen eines bestimmten Fachgebietes. Damit der Austausch automatisch erfolgen kann, wird die Ontologie normalerweise in einer maschinenlesbaren Sprache verfasst, z. B. auf Basis von XML.

Einsatz einer Ontologie

Ursprünglich werden Ontologien in der Informatik im Bereich der Künstlichen Intelligenz (KI) eingesetzt. Wissen wird formalisiert und in eine maschinenlesbare Form gebracht. Mit Hilfe von definierten Regeln werden Maschinen befähigt, Schlussfolgerungen zu ziehen und somit gewissermaßen intelligent zu handeln. Seit der Forschung im Bereich des *Semantic Web* sind Ontologien wieder verstärkt im Gespräch. Das Semantic Web ist eine WWW-Weiterentwicklung des W3C, bei der Webseiten mit Meta-Informationen über die Bedeutung der Inhalte erweitert werden.²⁴ Dies ermöglicht völlig neue Suchmöglichkeiten. Eine Website über Fußball kann z. B. von einem Suchprogramm anhand einer Sport Ontologie dem Thema Sport

²⁴ W3C: Das World Wide Web Consortium ist ein Gremium für die Standardisierung von Web Technologien. Offiziell nicht berechtigt, echte Standards festzulegen, gelten viele der sog. Requests for Comments (RFC) als de-facto-Standards, beispielsweise HTML, XML und CSS.

zugeordnet werden, obwohl sie in den Metadaten nicht explizit als Sportseite gekennzeichnet ist. [5.01]

Abbildung 8 zeigt eine mögliche Lösung für eine Ontologie-basierte Suchmaschine. Eine Suchanfrage nach dem Wort „Sport“ wird über einen Webservice an eine *Semantic Engine* (SE) geschickt. Dabei handelt es sich um eine Software, welche eine Ontologie verarbeiten kann. Die SE gleicht die Metadaten der indizierten Webseiten mit einer Sport Ontologie ab und liefert alle Ergebnisse zurück, die etwas mit Sport zu tun haben.

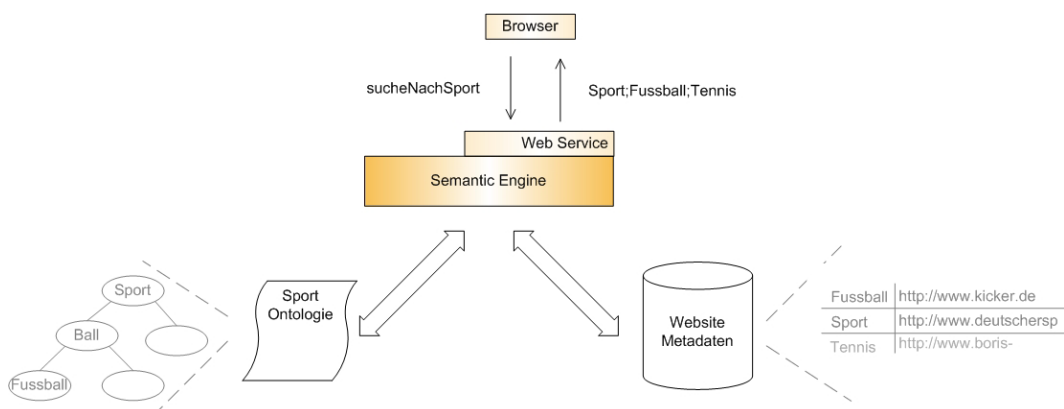


Abbildung 8 Ontologie basierte Suchmaschine

Neben dem Semantic Web werden heute in allen Wissensbereichen standardisierte Ontologien entwickelt, die von Domain Experten genutzt werden können, um Wissen auszutauschen. Den vielfältigen Einsatz von Ontologien zeigen folgende Beispiele:

- Kategorisierung von Webseiten (Yahoo!)
- Kategorisierung von Produkten (Amazon)
- Weltweiter Austausch von medizinischem Fachwissen (SNOMED)
- Allgemeine Klassifizierung von Produkten und Diensten (www.unspsc.org)

Zusammenfassend lässt sich sagen, dass mit einer Ontologie Wissen über ein bestimmtes Fachgebiet modelliert werden soll. In unserem Falle soll das Wissen über einen Reparatur Service von Haushaltsgeräten modelliert werden. Wie bereits erwähnt ist es wichtig, dass die Ontologie in einer maschinenlesbaren Form vorliegt. Es gibt viele Sprachen, die sich dafür eignen. Ich verwende für die

Reparatur Ontologie die auf XML basierende Web Ontology Language (OWL), die ich im Folgenden kurz erkläre.

5.2 OWL

OWL ist eine formale Sprache zur Beschreibung von Ontologien für das semantische Web, die es Such-Agenten ermöglicht, diese zu verarbeiten. Die Semantic Engine aus dem obigen Sportbeispiel könnte z. B. OWL Dateien verarbeiten. Es handelt sich um eine W3C Spezifikation und basiert historisch auf *DAML+OIL* und technisch auf *RDF* und *RDF-S*. Dabei handelt es sich um XML Vokabulare zur Beschreibung von Objekten. [5.02]

OWL unterteilt sich in drei Unterarten, die sich in Umfang und Einsatz unterscheiden.

- **OWL Lite** dient zur Klassifizierung und kann somit für die Erstellung von Taxonomien und Thesauri eingesetzt werden.
- **OWL DL** basiert auf der Description Logic, einem entscheidbaren Teil der Prädikatenlogik. Durch Einschränkungen der Sprache wird gewährleistet, dass Ontologien vollständig berechenbar sind. Dies ist eine Voraussetzung dafür, dass Software Agenten mit der Ontologie arbeiten können.
- **OWL FULL** bietet mehr Sprachkonstrukte ohne Einschränkungen an, lässt sich aber nicht mehr vollständig berechnen.

Derzeit wird fast ausschließlich OWL-DL eingesetzt, da Software Agenten auf eine vollständige Bearbeitbarkeit angewiesen sind.

In OWL werden die zu beschreibenden Fachbegriffe einer Ontologie durch ein mengenorientiertes Klassenmodell repräsentiert, welches Klassen, Instanzen und Eigenschaften als Sprachkonstrukte kennt. Klassen und Eigenschaften werden unabhängig voneinander modelliert, für beide gibt es Vererbung. Während Klassen die Fachbegriffe der Ontologie repräsentieren, lassen sich über Eigenschaften Beziehung zwischen diesen modellieren. Instanzen sind einzelne Individuen von Klassen.

Eine wichtige Verbesserung von OWL gegenüber ihren Vorgängern ist die Einführung von Äquivalenzklassen, also Klassen, die anders heißen, aber das gleiche bedeuten. Äquivalente Klassen werden später noch genauer beschrieben. Weiter können Klassen als disjunkt definiert werden. Ein anschauliches Beispiel für den Einsatz von disjunkten Klassen sind zwei Pizza Klassen, eine für vegetarische und eine für nicht vegetarische Pizzas. Eine Pizza Instanz kann selbstverständlich nicht gleichzeitig vegetarisch und nicht vegetarisch sein. [5.03]

In OWL gibt es zwei Arten von Eigenschaften. *DataProperties* werden dazu benutzt, Klassen mit einfachen Datentypen zu verbinden. *ObjectProperties* ermöglichen hingegen Relationen zu anderen Klassen. Man kann sich dies ähnlich wie Java Attribute vorstellen. Den *DatatypeProperties* entsprechen Attribute, die einfache Datentypen wie *int*, *long* oder *double* aufnehmen können. *ObjectProperties* sind vergleichbar mit Java Attributen, welche Referenzen auf Instanzen anderer Klassen aufnehmen können. Eigenschaften lassen sich in OWL sehr genau typisieren. Sie können

- invers
- symmetrisch
- funktional (Kardinalität von 1)
- transitiv

sein.

Für eine als invers deklarierte Eigenschaft muss ein korrespondierendes Gegenstück existieren, etwa zur Eigenschaft *hatMutter* eine Eigenschaft *hatKind*. Eine Eigenschaft *verheiratetMit* würde man dagegen als symmetrisch und funktional definieren. Wenn ein Mann mit einer Frau verheiratet ist, dann ist diese automatisch auch mit dem Mann verheiratet. Und zumindest in westlichen Ländern ist diese Verbindung nur einmal erlaubt, also funktional. Ist eine Eigenschaft transitiv und verbindet sowohl die Klassen A und B, als auch B und C miteinander, so steht automatisch auch A mit C in Relation. Hat z. B. Max einen Vorfahr Hans und Hans einen Vorfahr Peter, so ist Peter auch ein Vorfahr von Max, was in Abb. 9 dargestellt ist.

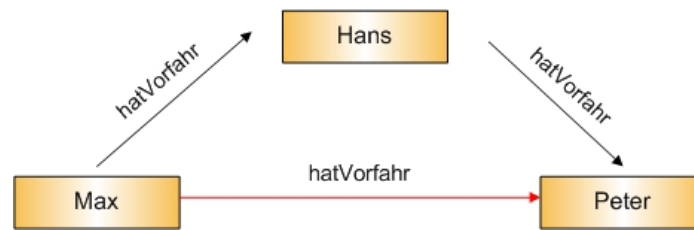


Abbildung 9 Transitive Eigenschaft

Regeln

Eine Stärke von OWL-DL ist die Möglichkeit der Definition von Regeln zur Verwendung der Sprachkonstrukte. Es kann genau bestimmt werden, welche Klassen über welche Eigenschaften mit anderen Klassen in Relation stehen dürfen, können oder müssen. Eine Regel kann zum Beispiel festlegen, dass eine Klasse „Kind“ über die Eigenschaft „hatMutter“ mit genau einer Klasse „Mutter“ in Relation stehen muss, was in Abbildung 10 als UML dargestellt ist.

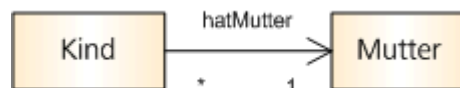


Abbildung 10 Regel über Mutter/Kind Relation

So sehen Auszüge aus dem zugrunde liegenden OWL File aus. Der XML Knoten Class besitzt für jede *Restriction* einen Subknoten. Der Bezug zu einer Eigenschaft und die Art der Einschränkung werden ebenfalls durch Subknoten realisiert.

```
<owl:Class rdf:ID="Kind">
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hatMutter"/>
    </owl:onProperty>
    <owl:someValuesFrom rdf:resource="#Mutter"/>
  </owl:Restriction>
  <owl:Restriction>
    <owl:cardinality >1</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#hatMutter"/>
    </owl:onProperty>
  </owl:Restriction>
</owl:Class>
```

Ob alle Regeln eingehalten werden kann von einem *Reasoner* geprüft werden. Dabei handelt es sich um eine Interferenzmaschine, die das Modell auf Konsistenz prüft. Da es sich bei OWL um XML handelt, schleichen sich bei der Erstellung eines Dokumentes schnell Fehler

ein. Obwohl das Dokument syntaktisch korrekt ist, kann die Semantik fehlerhaft sein. Beispielsweise sind unsinnige Klassen denkbar, etwa eine Pizza Klasse, die gleichzeitig als vegetarisch und nicht vegetarisch definiert wurde. Der Reasoner weist nach Prüfung der Ontologie auf diese Inkonsistenz hin. Eine weitere Aufgabe des Reasoner ist das Finden von äquivalenten Klassen.

Äquivalenz Klassen

Wie bereits erwähnt, sind äquivalente Klassen solche mit gleicher Bedeutung. Für einen Menschen ist es offensichtlich, dass eine weibliche Person mit einem Kind auch eine Mutter ist. Für einen Computer ist dies nicht ohne weiteres ersichtlich. Mutter wurde im obigen Beispiel als Klasse mit eigenem Namen definiert. Äquivalente Klassen können alternativ auch über bestimmte Merkmale beschrieben werden. Wir können z. B. die folgenden Bedingungen festlegen: Instanzen einer Klasse Person, die gleichzeitig die Eigenschaft „istWeiblich“ und „hatKind“ besitzen, sind Mütter. Die Klasse aller Instanzen, die diese Bedingungen erfüllen, ist äquivalent zur Klasse „Mutter“ und wird vom Reasoner als solche erkannt. Wir haben somit eine anonyme Superklasse beschrieben, nämlich die Klasse aller Personen mit der Eigenschaft „istWeiblich“ und „hatKind“. Diese ist in Abbildung 11 gezeigt.

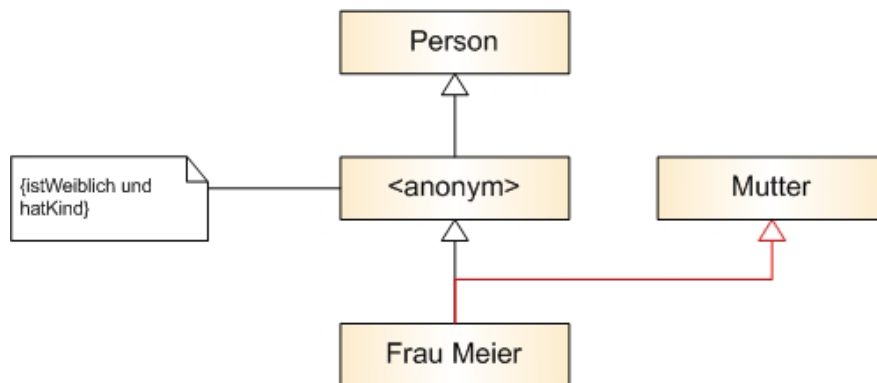


Abbildung 11 Anonyme Subklassen

Obwohl OWL noch sehr jung ist, gibt es bereits gute Editoren. Für die Erstellung der Repair Service Ontologie wurde das grafische Entwicklungstool Protégé der Stanford University eingesetzt. Protégé ist ein Open Source Projekt und bietet unter anderem einen offenen, erweiterbaren Editor zur Erstellung von Ontologien. Protégé wurde nicht speziell für OWL Ontologien entworfen, lässt sich aber über

Plugins erweitern. Es folgt eine kurze Einführung zum Tool und den benötigten Plugins. [5.04]

5.3 Protégé

Protégé ist ein integriertes Software Entwicklungswerkzeug. Neben der Erstellung von Ontologien unterstützt es die Erzeugung von Formularen für die leichte, domainspezifische Eingabe von Wissen. Um Ontologien in OWL sinnvoll zu entwickeln benötigt man ein Plugin für die OWL Sprachunterstützung und ein Plugin für die Visualisierung von Graphen, mit dem sich das Klassenmodell visualisieren lässt. [5.06][5.07] Zur Prüfung der Konsistenz der Klassen benötigt man ferner einen Reasoner. Als Reasoner empfiehlt sich der ebenfalls frei erhältliche Razer, dessen Integration mit Protégé sehr gut funktioniert (siehe Kasten).

Zusammen mit den Plugins und dem Reasoner eignet sich das Tool nun gut zur Erstellung von Ontologien in OWL. Es gibt mehrere sehr gute Tutorials, um das Tool kennenzulernen. [5.08]

Protégé kann die Vererbungshierarchie des Klassenmodells darstellen. Man kann sich alle namentlich definierten Klassen in einem sogenannten Asserted Klassenmodell anzeigen lassen. Zusätzlich dazu findet der Reasoner durch Auswertung der definierten Regeln zu jeder Klasse weitere, nicht namentlich als solche definierte Eltern-Klassen. Diese werden von Protégé im sog. Inferred Klassenmodell dargestellt. Das Inferred Modell nutzt also sehr stark Mehrfachvererbung, da zu vielen Klassen weitere Superklassen gefunden werden. Damit die Ontologie während der Entwicklung nicht zu unübersichtlich wird, empfiehlt es sich bei der Definition von Klassen auf Mehrfachvererbung zu verzichten. Werden Klassen semantisch von zusätzlichen Klassen abgeleitet, sollte man dies durch Regeln kenntlich machen und diese Abhängigkeiten vom Reasoner finden lassen. Im folgenden Beispiel soll der Unterschied zwischen dem in Abb. 12 und 13 dargestellten Asserted und Inferred Modell verdeutlicht werden.

Razer ist eine freie Implementierung eines DIG konformen Reasoners. Nach dem Starten lauscht er auf einem Port auf XML-basierte Anfragen. Als Trägerprotokoll dient HTTP. Der Reasoner wird unabhängig von Protégé gestartet, man muss lediglich IP Adresse und Port einstellen. Protégé kann mit Hilfe des Reasoners Klassen auf Konsistenz prüfen, sowie Äquivalenz-, Super- und Instanzklassen finden. Vor der Anfrage sendet Protégé die gesamte Ontologie an den Reasoner. [5.09] Eine Anfrage zur Prüfung der Konsistenz mit passender Antwort ist in folgendem Traceauszug gezeigt:

```
Protégé: <satisfiable id="Fridge">  
Razor: <true id="Fridge"/>
```

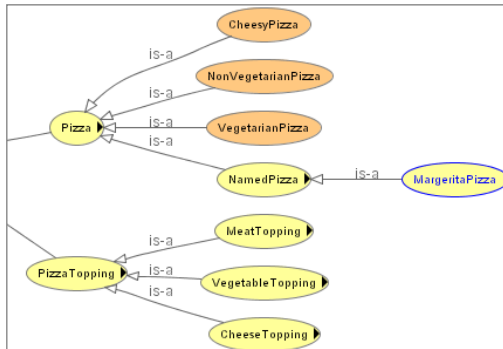


Abbildung 12 Asserted Klassenmodell

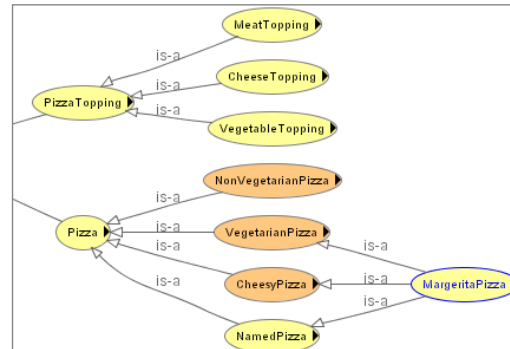


Abbildung 13 Inferred Klassenmodell

Eine Pizza Ontologie beschreibt Klassen für Pizzen und Klassen für den Pizzabelag. Neben konkreten Klassen wie MargheritaPizza gibt es allgemeine Klassen wie z. B. VegetarianPizza, für die keine Subklassen definiert sind. Zu diesen Klassen werden aber Regeln definiert, die festlegen, wann irgendeine Pizza Klasse, zu diesen gehört. Eine VegetarianPizza ist beispielsweise eine Pizza, die keine Relation zu einem MeatTopping hat, während eine CheesyPizza mind. eine Relation zu einem CheeseTopping benötigt. Das in Abb. 12 gezeigte Asserted Klassenmodell zeigt die MargheritaPizza lediglich als Subklasse von NamedPizza. Nicht zu sehen ist, dass sie eine Relation zu CheeseTopping und keine Relation zu einem MeatTopping besitzt. Mit Hilfe des Reasoners erkennt Protégé, dass die MargheritaPizza die Voraussetzungen für CheesyPizza und für VegetarianPizza erfüllt und fügt sie in das Inferred Klassenmodell ein. [5.08] Die Definition der Regeln in Protégé für den Käsebelag ist in Abbildung 14 gezeigt. Man wählt eine Restriction (*someValuesOf*) für ein bestimmtes Property (*hasTopping*) und wählt die Zielklasse aus.

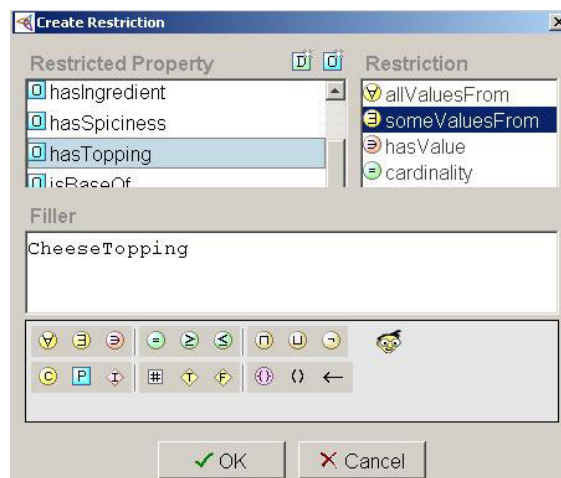


Abbildung 14 Regeldefinition in Protégé

Nachdem diese Regeln definiert wurden, erkennt der Reasoner, dass eine Margherita Pizza auch eine CheesyPizza ist und fügt sie dieser Superklasse zu.

Class	Changed superclasses
AmericanHotPizza	Added SpicyPizza, InterestingPizza, CheesyPizza, NonVegetarianPizza
AmericanPizza	Added CheesyPizza, InterestingPizza, NonVegetarianPizza
MargeritaPizza	Added VegetarianPizza, CheesyPizza
SohoPizza	Added CheesyPizza, InterestingPizza, VegetarianPizza

Abbildung 15 Klassifizierung durch den Reasoner

Bei der Visualisierung in Protégé muss man allerdings einige Abstriche machen. Beim Betrachten der oben gezeigten Klassenmodelle wird die Schwäche der Visualisierung deutlich. Das Plugin zur Visualisierung von Protégé ist nicht in der Lage Beziehungen zwischen Klassen darzustellen. Dies stellt bei der Entwicklung einer Ontologie eine große Einschränkung dar. Wünschenswert wäre eine grafische Darstellung der Eigenschaften, über welche die Klassen in Relation stehen. Diese werden in ihrer Gesamtheit lediglich als Liste gezeigt. Eine mögliche Umgehung dieses Problems wird später in Kapitel 5.5 gezeigt.

5.4 Service Ontologie

Die Service Ontologie dient der herstellerunabhängigen Kennzeichnung von Geräten und Diensten, die das Remote Management von Haushaltsgeräten betreffen. Es war schwierig echte Informationen zu finden, wie die Hersteller bisher vorgehen, um Haushaltsgeräte zu aktualisieren. Zwar gehört zum Funktionsumfang von CHAIN die Möglichkeit von Remote Updates, aber eine detaillierte Beschreibung wie solche Updates durchgeführt werden gibt es nicht.²⁵ Informationen darüber wie Geräte und passende Software vor allem herstellerübergreifend gekennzeichnet werden, finden sich bisher nicht. Es scheint so zu sein, dass Software Updates bei Haushaltsgeräten bisher mit seriellem Kabel und proprietären Protokollen durchgeführt werden.

Deshalb werden, ausgehend vom Reparatur Szenario, die beteiligten Personen, Geräte, Dienste und Rollen definiert und in OWL

²⁵ Vgl. Kapitel 3.5.1: CHAIN

abgebildet. Im Prinzip werden dazu sämtliche Entitäten als OWL Klassen beschrieben. Zur Erinnerung folgt hier noch einmal der Inhalt des Szenarios: Der Besitzer einer Waschmaschine beauftragt einen externen Service Provider mit der Reparatur des Gerätes. Der Service Provider, der diesen Dienst für Geräte unterschiedlicher Hersteller anbietet, schickt einen Service Mitarbeiter zum Besitzer, der das Gerät analysiert und über ein Software Repository des Service Providers die passende Software aufspielt. Die Analyse beinhaltet einen Abgleich der Geräteinformationen mit dem Reparatur Service Provider.

Zuerst werden sämtliche Entitäten als Klassen und Subklassen definiert. Danach werden, getrennt von den Klassen, die Eigenschaften festgelegt, welche die Klassen genauer beschreiben. In OWL werden Eigenschaften separat von den Klassen definiert und können dann verschiedenen Klassen zugeordnet werden. Im Zentrum der Ontologie stehen Geräte und Dienste, die jeweils in einer eigenen Klasse modelliert werden. Es gibt Klassen für die beteiligten Personen und Organisationen. Da für ein Software Update von Geräten bestimmte Informationen benötigt werden, müssen diese natürlich auch abgebildet werden.

Rollen und Personen

Der Besitzer des zu wartenden Gerätes und der Service Mitarbeiter werden als Subklassen der Klasse Person modelliert. Service Provider und Gerätehersteller hingegen werden als Organisationen dargestellt. Bei den Personen interessiert vor allem, in welchem Radius sie agieren können. Der Mitarbeiter sollte sich nicht zu weit vom Wohnort des Besitzers befinden. Der Service Provider zeichnet sich aus durch die Hersteller, deren Geräte er unterstützt.

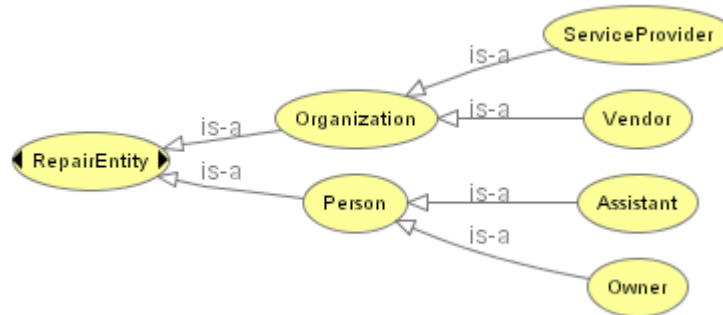


Abbildung 16 Personen und Organisationen

Device

In der Device Klasse werden Geräte modelliert. Denkbar sind Subklassen für unterschiedliche Geräte. Hier werden allerdings nur Haushaltsgeräte wie Kühlschränke, Gefriertruhen, Trockner und natürlich Waschmaschinen modelliert. Ein Gerät zeichnet sich durch seinen Namen, bestimmte Informationen und seine Funktionalität aus. Die Funktionalität wird über Dienste angeboten. Wie in Abbildung 17 zu sehen ist, werden in der Ontologie neben einigen konkreten, namentlich definierten Geräten zwei weitere allgemeine Geräteklassen definiert, von denen es keine direkten Instanzen gibt. Dabei handelt es sich um die Klassen *UpdateableDevice* und *NonUpdateableDevice*. Diese sind auch farblich hervorgehoben. Daran erkennt man, dass für diese Klassen ausreichende Regeln definiert sind, damit der Reasoner zu diesen Klassen zugehörige Instanzen finden kann.

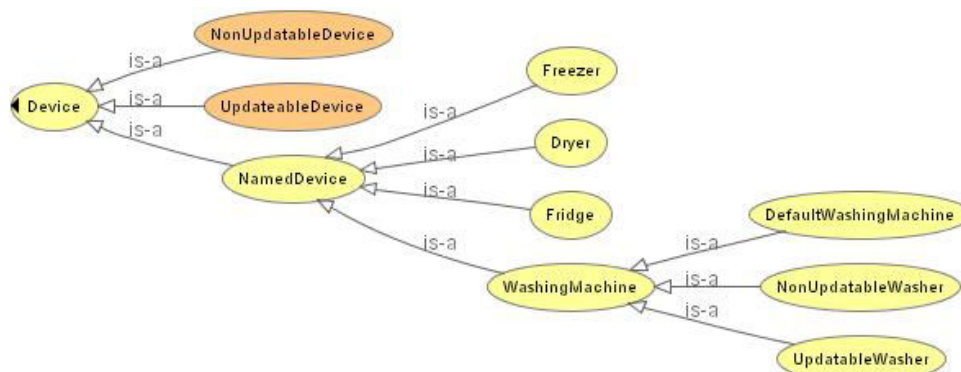


Abbildung 17 Ontology Device classes

Im Falle eines Software Updates interessiert es zu allererst, ob ein Gerät überhaupt updatefähig ist. Deshalb werden für die Klasse UpdateableDevice Regeln beschrieben, nach denen eine Zuordnung von beliebigen Instanzen zu dieser Klasse erfolgen kann. Diese Regeln sind relativ einfach. Die fragliche Instanz muss lediglich zu der Klasse Device (oder einer ihrer Subklassen) gehören und einen Update Dienst anbieten. Abb. 18 zeigt die Darstellung dieser Regeln in Protégé. Die Einordnung unter *Necessary & Sufficient* bedeutet, dass die Regeln nicht nur erfüllt sein müssen, sondern deren Erfüllung durch beliebige Instanzen anderer Klassen auch ausreichend für deren Zugehörigkeit zu UpdateableDevice ist.

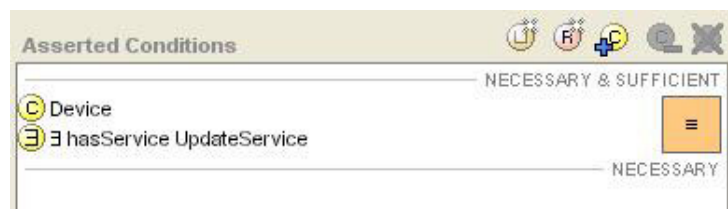


Abbildung 18 Regeln für updatefähige Geräte

Dienst

Ein Dienst repräsentiert eine bestimmte Funktionalität eines Gerätes. Das kann eine Berechnung sein, ein Druck- oder Scandienst oder das aktuelle Wetter. Über eine Eigenschaft *hatDienst* stehen Geräte mit Diensten in Verbindung. Für ein Remote Update ist vor allem der Update Dienst wichtig. Dieser besagt, dass das Gerät updatefähig ist, und beschreibt die Informationen die dafür benötigt werden, beispielsweise eine Update Adresse.

Information

In der Ontologie werden Informationen in einer allgemeinen Informationsklasse definiert, von der es unterschiedliche Subklassen gibt. Geräte besitzen unterschiedliche Informationen, die über entsprechende Dienste abgerufen werden können. Für die Nutzung einiger Dienste sind manchmal bestimmte Informationen notwendig. Für ein Software Update wird neben den Informationen zur Bestimmung des Gerätes wie beispielsweise Geräteklasse, Modell oder Seriennummer vor allem der aktuelle Softwarestand und eine Adresse für das Update benötigt. In Abbildung. 19 wird die Ontologie

im Asserted Klassenmodell gezeigt, also mit allen namentlich definierten Klassen.



Abbildung 19 Asserted Klassenmodell

Mithilfe des Reasoners können dann alle Geräte ermittelt werden, die updatefähig sind. In diesem Falle ist dies lediglich ein Gerät. Alle anderen werden als nicht updatefähig erkannt. Abbildung 20 zeigt das Inferred Klassenmodell.



Abbildung 20 Inferred Klassenmodell

Die Ontologie ist sehr rudimentär. Aufgrund des Aufbaus von OWL lässt sie sich aber leicht erweitern. Das resultierende OWL File kann jetzt weiterverwendet werden. Eine Semantic Engine kann beispielsweise erkennen, dass ein Mitarbeiter, der in der Ontologie eine Beziehung zu einem bestimmten Gerätetyp besitzt, dieses reparieren darf. Die Ontologie kann aber auch als Basis für eine Implementierung einer Anwendung verwendet werden. Dafür muss sie allerdings in ein anderes Format gewandelt werden.

5.5 Mapping

Die Reparatur Ontologie dient der Kennzeichnung der Entitäten (Geräte und Rollen) des Konfigurationskonzepts. Die Ontologie stellt eine abstrakte Sicht auf das System dar und eignet sich als Sprache sowohl für Menschen als auch für Maschinen. Umgesetzt ist sie mit OWL. Neben OWL gibt es einige andere Ontologie Sprachen. Mittels Transformations-Tools lassen sich Ontologien in andere

Ontologiesprachen oder in andere Modelle überführen. Es ist z. B. möglich, ein semantisches Modell in ein Datenbankschema oder in ein objektorientiertes Modell umzuwandeln.

5.5.1 OWL/UML Umwandlung

Wie bereits erwähnt sind die Möglichkeiten der Visualisierung des Protégé Plugins begrenzt. Zwar wird die Vererbungshierarchie der Klassen gezeigt, aber nicht die Relationen zwischen ihnen. Die Darstellung von Relationen ist aber eine Stärke von UML. Deshalb ergibt sich die Idee, das OWL Modell in UML umzuwandeln. Über das Internet lassen sich einige Dokumente finden, die die Umwandlung von Ontologien im Allgemeinen und die Transformation von OWL nach UML im Speziellen behandeln. Das Interesse scheint groß, was sich im regen Austausch in Mailforen zeigt. [5.10][5.11][5.12]

DAML UML Enhanced Tool (DUET)

Zuerst habe ich das DUET Tool untersucht. [5.13] Dieses Tool kann OWL Ontologien in UML XMI 1.3 und 1.4 umwandeln.²⁶ Es gibt drei Versionen von DUET. Neben der Standalone Version gibt es jeweils ein Plugin für die CASE Tools Rational Rose und ArgoUML.²⁷ Beide Plugins sind nur unter großem Aufwand lauffähig. Das Plugin für ArgoUML läuft beispielsweise nur mit einer nicht mehr erhältlichen Argo Version. Die Standalone Version von DUET hingegen läuft auf Anhieb. Leider ist die erzeugte XMI Datei aber nicht zu verwenden. Sie enthält unzählige neue, fehlerhafte Klassenbeschreibungen. Nach dem Import in Together und Poseidon erhält man ein nicht verwendbares Klassendiagramm, das unzählige falsche Klassen erhält. [5.14]

Protégé UML Plugin

Inzwischen gibt es für Protégé ein Plugin, das die direkte Speicherung der Ontologie in UML XMI erlaubt. [5.15] Der Test des Plugins ist erfolgreich und die XMI Datei lässt sich in Together importieren. Die

²⁶ XMI: XML Meta Interchange ist ein XML Standard zum Austausch von Modellen zwischen Softwareentwicklungswerkzeugen, beispielsweise von UML. Den Import von XMI unterstützen inzwischen einige CASE Tools.

²⁷ CASE: Computer Aided Software Engineering Tools unterstützen den Software-Ingenieur bei der Softwareentwicklung.

meisten CASE Tools bieten die Möglichkeit an, aus dem importierten Modell automatisch ein UML Klassendiagramm zu erzeugen. Leider ist die Zeit, die man investieren muss, um ein solches automatisch generiertes Klassendiagramm in eine entsprechende Form zu bringen, immens. Bei größeren Ontologien lohnt der Aufwand nicht. Dazu kommt, dass man bei einer Änderung der Ontologie neu transformieren muss und alle Änderungen des Diagramms verloren sind. Die Vorgehensweise, eine OWL Datei in UML XMI zu transformieren ist also nur bedingt geeignet, die beschränkte Visualisierung des Protégé Plugins zu erweitern. Hat man allerdings eine fertige Ontologie, die als Basis für weitere Arbeiten dienen soll, macht es durchaus Sinn, ein UML Modell zu generieren. Java Klassen, die aus dem Modell erzeugt werden, sind beispielsweise mit dem semantischen Modell konsistent. In Abb. 21 ist ein Auszug aus dem resultierenden UML Klassenmodell gezeigt, in dem die Beziehung zwischen Gerät und Dienst gezeigt sind.

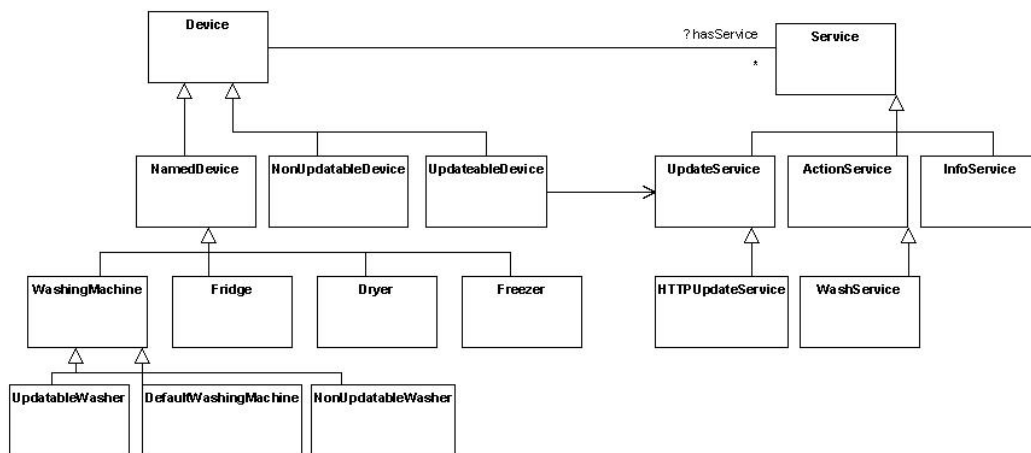


Abbildung 21 Aus der Ontologie erzeugtes UML Klassenmodell

Nach diesem Kapitel über eine gemeinsame Kennzeichnung von Geräten und Diensten befasst sich das nächste Kapitel mit der technischen Umgebung für Dienste.

6 OSGi

Die Open Service Gateway Initiative (OSGi) wurde im März 1999 gegründet. Ziel war es eine Plattform für Dienste im wachsenden Bereich der Heim- und Automobilautomation zu schaffen. Zurzeit sind in der OSGi Allianz etwa 40 Mitglieder aus allen Bereichen der Wirtschaft vertreten. [6.01]

Ausgangspunkt für die Gründung der OSGi war, wie so oft, das Internet mit seinen unerschöpflichen Möglichkeiten Geld zu verdienen. Das Internet ermöglicht die Entwicklung völlig neuer Dienste, Wertschöpfungsketten und Geschäftsmodelle. Nach Ansicht der OSGi Allianz bedurfte es eines allgemeinen Standards, der von möglichst vielen Richtungen der Industrie getragen wird, um diese neuen Möglichkeiten zu fördern. Ursprünglich gingen die Gründer davon aus, dass das vernetzte Heim die nächste Stufe darstellt, um die neuen Dienste an den Mann zu bringen. Inzwischen gelangten aber selbst die ursprünglichen Gründungsmitglieder zu der Auffassung, dass eine auf Industriestandards basierende Dienste Plattform in allen Bereichen nützlich ist, in denen Dienste übers Netz auf Geräte geliefert und verwaltet werden sollen. Ein Einsatz ist zum Beispiel ein modernes Automobil Infotainmentsystem, wie es schon von der Firma BMW eingesetzt wird. [6.02]

Ergebnis der OSGi Bemühungen ist die Definition eines Java Frameworks für die Verwaltung und die Auslieferung von Diensten in Netzen. Die Architektur der sogenannten OSGi Service Plattform wird von der OSGi Allianz dabei lediglich als Spezifikation angeboten. [6.03] Konkrete Implementierungen kommen von unabhängigen Herstellern und der unermüdlichen Open Source Gemeinde, beispielsweise Knopflerfish.²⁸

Da OSGi ursprünglich vor allem für die Heim Automation gedacht war, wurden den besonderen Anforderungen an CPU und Speicher der dort häufig eingesetzten Embedded Devices Rechnung getragen.

²⁸ Vgl. Kapitel 7.2.1 Knopflerfish

6.1 Service Plattform – Sparsamer Container für Dienste

Die OSGi Service Plattform stellt eine standardisierte, Komponentenorientierte Laufzeitumgebung für Netzdienste in Form eines Java Frameworks dar.²⁹ Die Plattform fungiert als Software Container, der den Komponenten – in der OSGi Terminologie als Bundles bezeichnet – fundamentale Basisdienste zur Verfügung stellt, vergleichbar etwa mit einem EJB Container. Dazu gehören vor allem die Steuerung des Lebenszyklus, die Kooperation zwischen den Komponenten und Sicherheitsdienste. Insbesondere auf die ersten beiden soll später noch genauer eingegangen werden.

Die Entscheidung für Java als Basistechnologie erfolgte wegen der benötigten Plattformunabhängigkeit. Trotz des Einsatzes von Java ist die Kernplattform klein und effizient. Als Laufzeitumgebung genügen schon sehr kleine Virtuelle Maschinen (VM). In der OSGi Spezifikation werden die Klassen beschrieben, die von einer Java Laufzeitumgebung unterstützt werden müssen. Bei der Zusammenstellung wurde besonders auf den Speicherverbrauch geachtet. So können schon kleinere VMs als Laufzeitumgebung eingesetzt werden. Dazu gehören z. B. auch manche VMs der Java Micro Edition (J2ME). [6.04]

Zudem kommt in der OSGi Plattform ein starkes Modell für die Koexistenz der Softwarekomponenten zum tragen. Alle Komponenten laufen dabei in einer einzigen VM. Dies führt dazu, dass der Speicherverbrauch erheblich gesenkt und die Kosten der Interprozesskommunikation fast gänzlich eliminiert werden. Ferner werden von mehreren Komponenten gemeinsam genutzte Klassen nur ein einziges Mal geladen, und nicht, wie sonst üblich, von jeder Klasse neu, was den Speicherverbrauch weiter senkt. [6.05] Viele J2ME-Geschäftsanwendungen benötigen beispielsweise die Bibliothek des Java Messaging Service (JMS) für die Kommunikation, die etwa 40Kbyte Code umfasst. J2ME selbst hat keine JMS Unterstützung. Laufen jetzt 6 J2ME Anwendungen, muss jede die Bibliothek selbst importieren. Bei 6 Anwendungen werden also 200KByte für den immer gleichen Code belegt. Durch den OSGi Lademechanismus wird die JMS Bibliothek lediglich einmal geladen, was in Abb. 22 dargestellt ist.

²⁹ Der Begriff Framework bezeichnet in der Objekt-orientierten Programmierung eine Standard Softwarearchitektur.

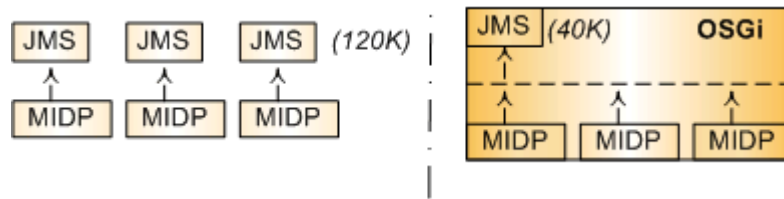


Abbildung 22 Laden von Klassen in OSGi

Im Folgenden soll die Architektur etwas genauer beleuchtet werden, ohne aber zu sehr in die technischen Details einzutauchen. Ein besonderer Schwerpunkt liegt auf den Mechanismen zur Kooperation und Interaktion der Komponenten, da dies die technische Basis für eine einfache Integration von OSGi mit anderen Technologien darstellt.

6.2 Architektur

Wie bereits erwähnt handelt es sich bei OSGi um ein Java Framework, welches Softwarekomponenten (Bundles) eine Laufzeitumgebung zur Verfügung stellt. Die Bundles laufen auf der OSGi Plattform, welche ihrerseits eine Java Laufzeitumgebung benötigt. Dies wird in Abbildung 24 schematisch dargestellt. Einige besonders wichtige Bundles wie ein HTTP Server müssen laut Spezifikation schon vom Plattformhersteller geliefert werden. Für andere, wie zum Beispiel ein Bundle für die Unterstützung des UPnP Protokolls, wurden lediglich die Schnittstellen definiert. Implementierungen kommen dann von Dritten.³⁰

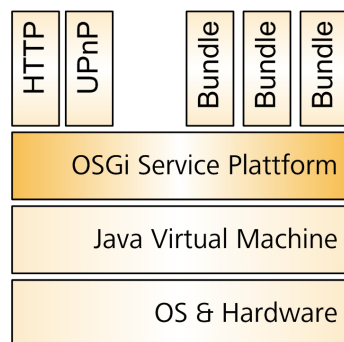


Abbildung 24 OSGi Plattform

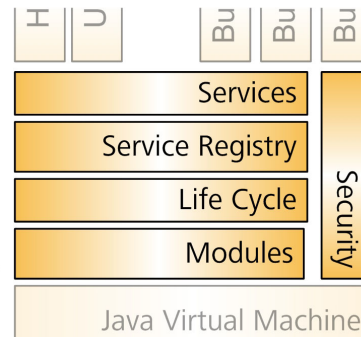


Abbildung 23 Plattform Architektur

³⁰ Vgl. Kapitel 7.2.3: UPnP Base Driver

Die unterschiedlichen Bereiche der eigentliche Service Plattform sind in Abbildung 23 dargestellt. Es gibt Bereiche für das Laden der Klassen (Modules), den Lebenszyklus (*Life Cycle*) und die Sicherheit (*Security*). *Services* und die *Service Registry* sind für die Kooperation zwischen den Bundles zuständig.

6.2.1 Lebenszyklus eines Bundles

Über die Plattform lässt sich der gesamte Lebenszyklus einer OSGi Softwarekomponente, eines Bundles, steuern. Ein Bundle wird über die Plattform installiert, gestartet, aktualisiert, gestoppt und am Ende deinstalliert. Dabei kann es die in Abb. 25 dargestellten Zustände einnehmen. Nachdem ein Bundle installiert ist erzeugt das Framework ein Objekt, welches das Bundle repräsentiert und über welches der restliche Lebenszyklus gesteuert wird. Bevor das Bundle gestartet werden kann, versucht das Framework alle Abhängigkeiten zu anderen Bundles, Klassen und Packages aufzulösen. [6.06]

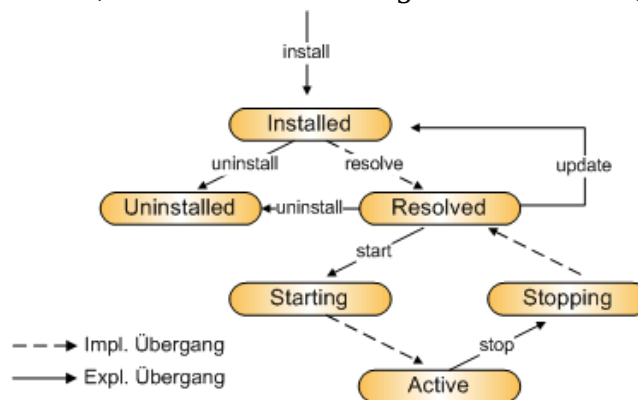


Abbildung 25 Lebenszyklus eines OSGi Bundles

Physikalisch handelt es sich bei einem Bundle um ein ZIP-kompatibles JAR Archiv, in dem alle Java Klassen mit ihren zugehörigen Ressourcen samt Archivbeschreibung, dem sog. Manifest File, zusammengefasst sind. Wichtig ist, dass eine Klasse des Archivs über die OSGi Schnittstelle *BundleActivator* verfügt und die implementierende Klasse im Manifest angegeben ist. Über diese Schnittstelle greift die Plattform auf das Bundle zu und steuert den Lebenszyklus. Die Plattform selbst ist dabei unabhängig vom Lebenszyklus der Bundles. Um ein Bundle zu aktualisieren ist es nicht notwendig die Plattform neu zu starten. Damit sind auch andere in der Plattform laufenden Bundles nicht betroffen.

Schließlich soll noch erwähnt werden, dass laut OSGi Spezifikation der Lebenszyklus der Bundles, sowie der Plattform selbst vollständig fernsteuerbar sein muss. Dies ist für viele Industriebereiche eine Grundvoraussetzung für den Einsatz von OSGi. Dabei definiert die Spezifikation aber lediglich eine Minimalschnittstelle, die aber zu den gängigen Remote Management Standards kompatibel ist. [6.07]

6.2.2 Kooperation der Bundles durch Services

Die eigentliche Aufgabe der OSGi Service Plattform besteht darin, Software Dienste zu verwalten. Ein Bundle kann als Laufzeithülle einer Softwarekomponente betrachtet werden, während ein OSGi Dienst (Service) die eigentliche Funktionalität zur Verfügung stellt. Durch das Anbieten und Nutzen von Funktionalität durch Dienste können Bundles kooperieren. Ein Bundle muss seine Aufgabe also nicht ganz allein lösen, sondern kann auf die Dienste anderer Bundles zurückgreifen. Dieses Teilen von Funktionalität führt dazu, dass das System klein bleibt. Durch die Nutzung von Diensten entstehen allerdings Abhängigkeiten der Bundles untereinander, welche von der Plattform kontrolliert werden.

Service

Dienste sind das eigentliche Herz der OSGi Service Plattform. Der Mechanismus für Dienste soll deshalb im Detail erklärt werden. Will ein Bundle eine bestimmte Funktionalität zur Verfügung stellen, kapselt es diese in ein OSGi Service Objekt. Jede Java Klasse kann als Service Objekt fungieren. Sie sollte über mindestens eine Schnittstelle verfügen und muss von ihrem Bundle unter dieser in einer zentralen Stelle der Plattform registriert werden, der OSGi Service Registry. Ein Bundle kann mehrere Dienste registrieren. Ein Dienst ist dabei eng an den Lebenszyklus seines Bundles gebunden. Wird ein Bundle gestoppt, dann werden auch alle von diesem Bundle registrierten Dienste aus der Registry entfernt. [6.08]

Service Registry

Die Service Registry ist die zentrale Stelle der Plattform, in der alle Dienste registriert sind. Über die Registry können Bundles Dienste finden. Sie können die Registry unter Angabe einer Schnittstelle nach den zugehörigen Dienste-Objekten durchsuchen. Durch das Nutzen

von Diensten anderer Bundles entstehen Abhängigkeiten zwischen den Bundles. Wird zum Beispiel ein Bundle gestoppt, dessen Dienste von anderen Bundles genutzt werden, so können diese evtl. nicht mehr korrekt arbeiten. Die Plattform stellt jedoch ein komfortables Benachrichtigungssystem zur Verfügung, über welches die Bundles rechtzeitig über solche Vorkommnisse informiert werden und entsprechend reagieren können.

Event System

Benötigt ein Bundle einen Dienst, der noch nicht registriert ist, ist es nicht notwendig, ständig die Registry zu fragen, ob der Dienst inzwischen vorhanden ist. Es besteht vielmehr die Möglichkeit, sich von der Service Plattform darüber unterrichten zu lassen, wenn ein neuer Dienst registriert wird. Für den Benachrichtigungsmechanismus setzt das OSGi Framework das Observer bzw. Publisher/Subscriber Pattern um, welches auch in anderen Java Technologien eingesetzt wird. In Abb. 26 sieht man, dass das Framework die Rolle des Subjects übernimmt, während ein Bundle zum Observer wird. Ein an einer Zustandsänderung interessiertes Bundle abonniert diese Information beim Framework. Dafür teilt es dem Framework eine Schnittstelle mit, unter der es Nachrichten empfangen kann. Solche Nachrichten gibt es für Zustandsänderungen von Diensten, Bundles oder auch der Plattform selbst. Allerdings enthalten solche Nachrichten nur die Information, dass sich der Zustand eines Services verändert hat, und nicht, um welchen Dienst es sich konkret handelt. Dies muss ein Bundle nachdem es benachrichtigt wurde bei der Registry nachfragen.

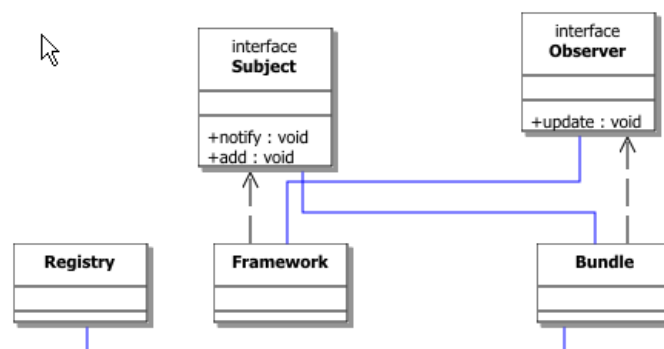


Abbildung 26 OSGi Event System

6.3 Gegenüberstellung von OSGi und UPnP

Oft wird die Frage gestellt, wie UPnP und OSGi zusammenhängen, bzw. wann welche Technologie eingesetzt wird. UPnP ist ein Daten Kommunikations-Protokoll, das keine Aussage darüber trifft, wo und wie Programme ausgeführt werden. Dies wird in der jeweiligen Implementation entschieden. Dagegen beschreibt die OSGi Service Plattform wo und wie Programme ausgeführt werden, aber nicht, wie sie miteinander kommunizieren. Während UPnP also die Kommunikation von Softwarediensten regelt, stellt OSGi die Laufzeitumgebung zur Verfügung, ohne festzulegen welche Protokolle unterstützt werden. Beide Standards konkurrieren also nicht, sondern sind vielmehr komplementär und lassen sich sehr gut kombinieren. [6.02] In der OSGi Spezifikation gibt es deshalb auch ein eigenes Kapitel, indem UPnP Schnittstellen definiert werden.

6.4 Zusammenfassung

Die OSGi Service Plattform bringt viele Vorteile mit sich und adressiert einige Probleme in der Entwicklung im Bereich der Heimautomation. Aus Sicht der Unternehmensplaner kann der auf breiter Basis stehende Industriestandard OSGi als sichere Investition gelten, die auch mittelfristig unterstützt wird. Die Wirtschaftlichkeit der Programmierung für Embedded Devices wird erhöht. Zum einen berücksichtigt OSGi durch sein ausgeklügeltes Koexistenzmodell zusammen mit der Unterstützung von J2ME die begrenzten Ressourcen von Embedded Devices. Gleichzeitig kann der Entwickler durch den Einsatz eines Komponentenmodells die Vorteile der OO-Programmierung nutzen. Durch die Verbindung des starken Kooperationsmodells mit Diensten und Registry, sowie der Möglichkeit der Integration von Protokollen lassen sich verschiedene Netztechnologien miteinander verbinden. Der in heutigen Heimnetzen anzutreffende Protokoll Wildwuchs wird etwas beherrschbarer.

OSGi wird heute in vielen Bereichen eingesetzt. Die Automobilindustrie setzt auf die Vorteile der Fernwartung. Und der Lifecycle Mechanismus für das dynamische Laden von Bundles wird z. B. von der Entwicklungsumgebung Eclipse zum Laden seiner Plugins verwendet. Ob allerdings der lang ersehnte Durchbruch auf dem Markt der Haustechnik und Heimautomation durch den Einsatz von OSGi erreicht wird ist weiterhin fraglich. Das Thema Privacy, also die

Frage wie die Privatsphäre des Kunden geschützt wird, wenn z. B. sein OSGi Gateway ferngewartet wird, ist noch gänzlich ungelöst. Der Begriff der Privacy erscheint in der aktuellen OSGi Spezifikation nur als Randbemerkung. Aufgrund der technischen Möglichkeiten ist OSGi dennoch eine gute Wahl für die Umsetzung eines Remote Update Konzepts.

7 Technische Umsetzung

Die beiden Schlüsselkomponenten Benutzerfreundlichkeit durch Zero Configuration und Integration unterschiedlicher Technologien finden in der technischen Umsetzung Anwendung. Eine Service Anwendung soll dabei einen konkreten Wartungsvorgang verdeutlichen. Mit Hilfe der Anwendung wird ein Haushaltsgerät über ein Netzwerk analysiert und mit neuer Software zu versorgt. Die Anwendung integriert über unterschiedliche Technologien, der Nutzer ist nur wenig involviert. Die Service Anwendung basiert auf offenen, standardisierten Technologien. Zentrale Komponente des Systems ist ein Gateway. Aufgrund seiner guten Integrationsfähigkeiten wird als technische Basis des Gateways OSGi gewählt. Die Kommunikation mit dem Gerät erfolgt über UPnP. Das Gateway kommuniziert mit der Waschmaschine, dem Web Client, dem Software Repository und einem Bluetooth Client. Zuerst folgt die Umsetzung der Waschmaschine, danach wird eine Serveranwendung auf dem Gateway entworfen.

7.1 Waschmaschine: Design und Implementierung

Aufgrund fehlender echter UPnP Haushaltsgeräte wird ein solches Gerät simuliert. Ein Java Programm implementiert die Dienste einer Waschmaschine und bietet sie als UPnP Services im Netz an. Für UPnP Control Points spielt es dabei keine Rolle, ob hinter der Software ein wirkliches Gerät steht oder nicht. Zu Demonstrationszwecken wurden von mir folgende Dienste implementiert, die sich im Wesentlichen aus den in Abbildung 27 gezeigten Anwendungsfällen *Waschen*, *Information* und *Wartung* ergeben. Zum Waschen gehören das sofortige Starten, das Festlegen eines späteren Startzeitpunktes mittels eines Timers, sowie das vorzeitige Beenden des Waschvorgangs. Als Informationen stehen dem Benutzer Waschstatus (Bereit, Waschend, Beendet), Startzeit, Waschkdauer und Restwaschzeit zur Verfügung. Im Wartungsfall kann man zusätzliche Informationen wie z. B. den Softwarestand oder einen Fehlerstatus abrufen. Der Update Dienst erlaubt das Laden einer Datei auf die Waschmaschine.

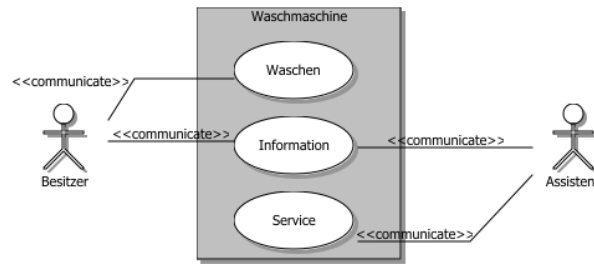


Abbildung 27 Anwendungsfälle der Waschmaschine

7.1.1 Design

Die Simulation ist sehr einfach und besteht aus zwei Hauptklassen und wenigen Hilfsklassen. Die Klasse *UPnPWasherDevice* ist für die Funktionalität der Maschine verantwortlich. Sie wird unterstützt durch einen Timer, ein Soundmodul und eine Klasse für das Laden neuer Software. Die grafische Darstellung der Waschmaschine und eine einfache grafische Benutzerschnittstelle (GUI) wird durch die Klasse *UPnPWasherGUI* realisiert. Sie zeigt immer den aktuellen Status der Waschmaschine an. Für den Abgleich des Zustandes der Waschmaschine mit der Anzeige wird das Observer Pattern verwendet. Sobald der Zustand der Waschmaschine sich ändert, benachrichtigt die Klasse *UPnPWasherDevice* die GUI, die sich als Observer registriert hat. Nach der Benachrichtigung fragt diese den neuen Status der Maschine ab und aktualisiert sich entsprechend. Das Design ist in Abbildung 28 dargestellt.

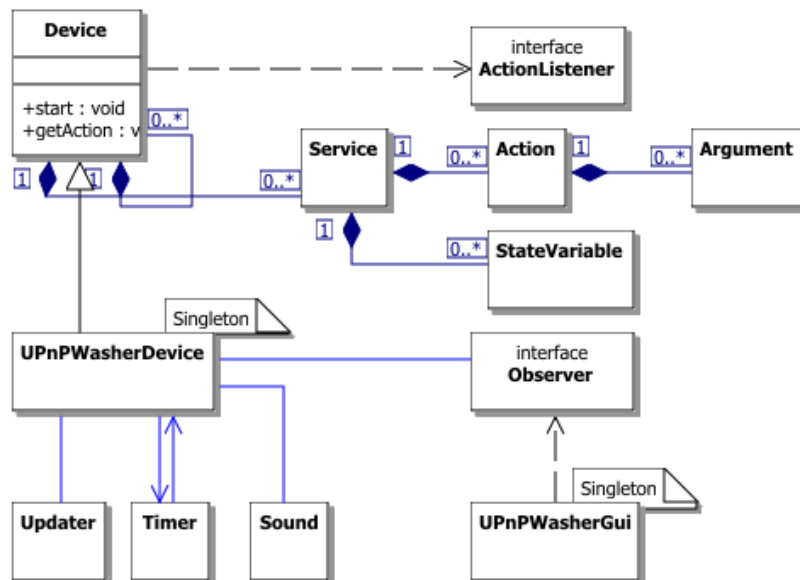


Abbildung 28 Klassendiagramm der Waschmaschine

Die Waschmaschine lässt sich über die GUI starten und stoppen. Die normale Kontrolle der Maschine erfolgt jedoch über UPnP Control Nachrichten, die von einem UPnP Control Point versendet werden.

7.1.2 Cyberlink

Die Simulation implementiert die UPnP Protokolle nicht selbst, sondern nutzt eine UPnP Implementierung in Java von Cybergarage. Unter dem Namen Cyberlink stellt der Japaner Satoshi Konno freie Bibliotheken für UPnP Entwickler in C++ und Java zur Verfügung. Die Bibliotheken implementieren die UPnP Protokolle und stellen sie dem Entwickler über einfache Schnittstellen zur Verfügung. Es ist damit relativ einfach, UPnP Geräte (in Java) zu entwickeln. [7.01]

Für die UPnP Kommunikation erweitert *UPnPWasherDevice* die *Device* Klasse. Dadurch wird sie zu einem UPnP Root Device. Nach dem Aufruf der *start()* Methode sendet die Waschmaschine eine *Notify* Nachricht an die Standard Multicast Adresse (239.255.255.250:1900). (siehe Kasten) Die wichtigste Angabe darin ist die Adresse, unter welcher die Gerätebeschreibung (*description.xml*) über einen Geräte-eigenen HTTP Server abgerufen werden kann. In dieser Datei findet ein Control Point alle angebotenen Dienste inkl. der Adressen, unter denen weitere Beschreibungen mit genauen Informationen über die Nutzung dieser zu finden sind. Angesprochen werden die Dienste dann über UPnP Control Nachrichten.

```
NOTIFY * HTTP/1.1
SERVER: Windows 2000/5.0 UPnP/1.0 CyberLink/1.3.2
CACHE-CONTROL: max-age=1800
LOCATION: http://192.168.123.225:4004/description.xml
NTS: ssdp:alive
NT: upnp:rootdevice
USN: uuid:alcatelUpnPWasherDevice::upnp:rootdevice
HOST: 239.255.255.250:1900
```

7.1.3 Control Nachrichten

Bei Control Nachrichten kann man zwischen Query Nachrichten für die reine Statusabfrage ohne Übergabeparameter und Action Nachrichten mit Argumenten unterscheiden. Für beide Typen implementiert die Device Klasse Listener. Der Einfachheit halber nutzt die Waschmaschine lediglich Action Nachrichten. Aktionen

werden durch eine eigene abstrakte Klasse repräsentiert, die sich nicht direkt instanziierten lässt. Instanzen werden vielmehr unter der Angabe des Namens über die Device Klasse angefordert. Wichtig ist, dass dieser Name mit der in der Service Beschreibung angegebenen Aktionsnamen übereinstimmt, da die Instanz mit weiteren Angaben aus dieser Datei initialisiert wird. Die Waschmaschine besitzt einen Timer, mit dem ein zukünftiger Starttermin gesetzt werden kann und der über UPnP Dienste zugreifbar ist. Im Folgenden werden beispielhaft die betreffenden Einträge aus der Timer Service Beschreibung für die Aktion zur Abfrage der Restlaufzeit, sowie ihr korrespondierender Java Code, gezeigt. Zuerst der Eintrag der Beschreibungsdatei:

```
<action>
  <name>GetTimeLeft</name>
  <argumentList>
    <argument>
      <name>TimeLeft</name>
      <relatedStateVariable>Time</relatedStateVariable>
      <direction>out</direction>
    </argument>
  </argumentList>
</action>
```

Im Java Code der Waschmaschine erzeugt man unter Angabe des gleichen Namens dann eine Action Instanz, die dieser Aktion entspricht:

```
UPnPWasherDevice device = new UPnPWasherDevice();
Action getTimeLeftAction = device.getAction("GetTimeLeft");
```

Damit die Waschmaschine auf eine eingehende Control Nachricht reagieren kann, wird bei der Action Instanz ein ActionListener registriert. Wie im Klassendiagramm zu sehen ist, implementiert die *Device* Klasse selbst schon eine ActionListener Schnittstelle. Deshalb wird eine Instanz von *UPnPWasherDevice* übergeben, welche die Device Klasse ja erweitert:

```
getTimeLeftAction.setActionListener(device);
```

Die ActionListener Schnittstelle besitzt genau eine Methode, die von der Klasse *UpnPWasherDevice* überschrieben wird. Sie wird jedes Mal gerufen, wenn eine entsprechende Control Nachricht über den HTTP Server empfangen wird. In dieser Methode wird die eigentliche Business Logik aufgerufen, in diesem Fall die *getTimeLeft* Methode des Timers.

```
public boolean actionPerformed(Action action)
{
    String actionName = action.getName();
    boolean ret = false;
    if(actionName.equals("GetTimeLeft"))
    {
        Argument timeArg = action.getArgument("TimeLeft");
        ret = timeArg.setValue(timer.getTimeLeft()+" sec.");
    }
    return ret;
}
```

7.1.4 Update Service

Fast der wichtigste Dienst der Waschmaschine ist die Möglichkeit eines Remote Updates. Dieser wird über eine UPnP Nachricht kontrolliert. Um neue Software aufzuspielen benötigt ein Client zuerst die Update URL des Gerätes. Diese erhält man über den UPnP Dienst *Update* vom Gerät selbst. Dann wird die Datei als HTTP Post Nachricht an diese URL gesendet. Nachdem die Datei erfolgreich geladen wurde erhält man eine Bestätigung. Der Ablauf ist in folgendem Sequenzdiagramm zu sehen. (*Anm.: ServiceClient ist kein Objekt, sondern ein Browser. Es handelt sich also streng genommen nicht um ein korrektes UML Diagramm*)

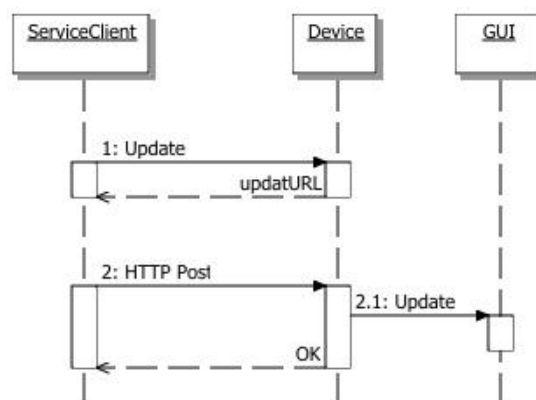


Abbildung 29 Sequenzdiagramm des File Uploads.

Sobald eine Datei vollständig geladen ist, setzt die Waschmaschine ihren Status auf *Updated* und benachrichtigt wiederum den Observer über die Zustandsänderung. Der Ladevorgang wird über eine veränderte Bildanzeige simuliert. Die GUI zeigt in Abhängigkeit des Zustandes unterschiedliche Bilder einer Waschmaschine an. Zur Demonstration des Datei Uploads wird ein neuer Satz Bilder auf die Maschine geladen. Ist der Update erfolgreich wird dies dann sofort an der grafischen Darstellung sichtbar. Abbildung 30 zeigt die Maschine beim Initialisieren, beim Waschen und nach einem Datei Upload.



Abbildung 30 Zustand der Maschine beim Anmelden der UPnP Dienste, beim Waschen und nach erfolgreichem Update.

Wie bereits erwähnt besitzt die Waschmaschine einen eigenen HTTP Server, über den die UPnP Kommunikation läuft. Über diesen wird auch der Datei Upload realisiert. Dafür wird einfach die `HttpRequestReceived` Methode der Device Klasse überschrieben. Darin wird dann geprüft, ob es sich bei der Ziel URL der empfangenen HTTP Nachricht um eine Datei Upload Adresse handelt. Wenn ja, wird diese Nachricht an die Klasse `UpnPWasherUpdater` weitergeleitet. Dort werden alle enthaltenen Files extrahiert und in das Zielverzeichnis geschrieben. Die Demo erwartet als File Upload ein ZIP File, das die neuen Bilder enthält. Handelt es sich bei der URL nicht um die Update Adresse wird der Aufruf einfach an die Superklasse (Device) weitergegeben. [7.02]

Als großes Problem stellte sich die Implementierung des HTTP Servers in den Cyberlink Bibliotheken heraus. Dieser ist ausschließlich für die Verarbeitung von textbasierten Nachrichten ausgelegt, da ja eigentlich nur UPnP Nachrichten versendet und empfangen werden. Der Server empfängt Nachrichten wie in Java üblich als Bytes über einen sogenannten *InputStream*. Dieser wird

dann für die weitere Verarbeitung als Text codiert. Handelt es sich aber, wie bei einem Datei Upload der Fall, beim Inhalt der HTTP Nachricht um Binärdaten, werden diese verändert und damit unbrauchbar. Gelöst wurde das Problem von Thorsten Dobelmann durch eine kleine Änderung der Cyberlink Bibliotheken. Diese wurden dahingehend verändert, dass vor der Textcodierung geprüft wird, ob es sich bei den Nutzdaten der HTTP Nachricht um Binärdaten handelt oder nicht. [7.03]

Getestet wurde der Datei Upload mit einem kleinen Tool aus dem Apache Projekt, mit welchem sich Dateien als HTTP Multipart Nachrichten auf einen Webserver laden lassen. Für die Demo müssen die Dateien in einer ZIP Datei verpackt sein. Abbildung 31 zeigt das Lade Tool.

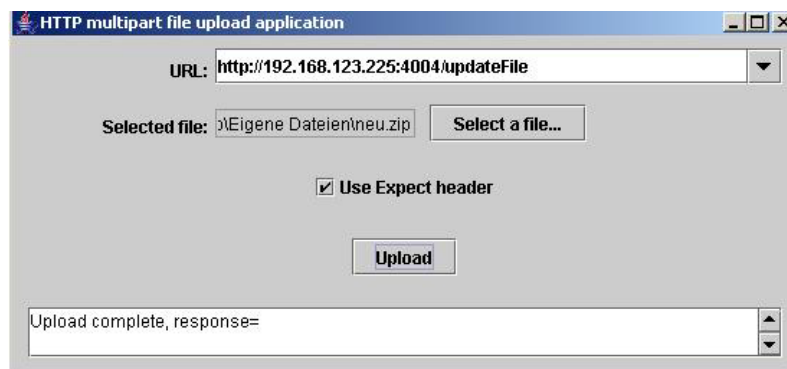


Abbildung 31 Apache HTTP Uploader

7.2 Gateway

Hauptkomponente des Systems ist das Service Gateway, welches zwischen den Clients des Benutzers und des Service Mitarbeiters, der Waschmaschine sowie dem Service Provider vermittelt. Die Funktionen der Waschmaschine können über einen einfachen Beispiel Control Point gesteuert (und getestet) werden. In der Diplomarbeit wird deshalb auf die Implementierung einer Web Client Anwendung für die normale Steuerung (Start, Stop, etc.) verzichtet. Stattdessen wird der Service Client für die Analyse und das Update entworfen. Technische Basis ist die im letzten Kapitel eingeführte OSGi Plattform. Als konkrete OSGi Implementierung wird das Open Source Projekt Knopflerfish verwendet. [7.04]

7.2.1 Knopflerfish

Das Open Source Projekt Knopflerfish basiert auf dem Gatospace GDSP OSGi Framework, dessen Mitglieder zu den Gründungsmitgliedern von OSGi gehören. Knopflerfish ist eine vollständige Implementierung der OSGi Spezifikation nach der Version R3. Es enthält die wichtigsten Komponenten der Plattform. Neben den Kernkomponenten zum Steuern des Lebenszyklus der Bundles gehören dazu der HTTP Server und der Device Manager. Letzterer verwaltet Software Komponenten, die mit Gerätetreibern in Verbindung stehen. Zusätzlich bietet Knopflerfish eine ganze Reihe weiterer, nützlicher Komponenten. Es gibt beispielsweise einen Logging Dienst, eine komfortable Konsole zur Steuerung der Plattform, sowie den Desktop. Bei letzterem handelt es sich um eine umfangreiche GUI. Der Desktop zeigt das gesamte Framework sehr übersichtlich an. Die wichtigsten Aktionen wie das Starten und Stoppen der Bundles kann man darüber erledigen. Zusätzlich werden alle wichtigen Informationen der Bundles oder einzelner Dienste angezeigt. Abbildung 32 zeigt den Desktop. Der Webserver ist als aktives Bundle farblich hervorgehoben. Im rechten Fenster sind die von diesem Bundle in der Registry exportierten Dienste zu sehen.

Der Desktop selbst ist ebenfalls ein Bundle, das die Swing Bibliotheken der Java Standard Edition verwendet. Hauptzweck ist die lokale Steuerung des Frameworks. Über ein optionales SOAP Bundle ist es aber auch denkbar, entfernte Frameworks zu steuern. [7.05]

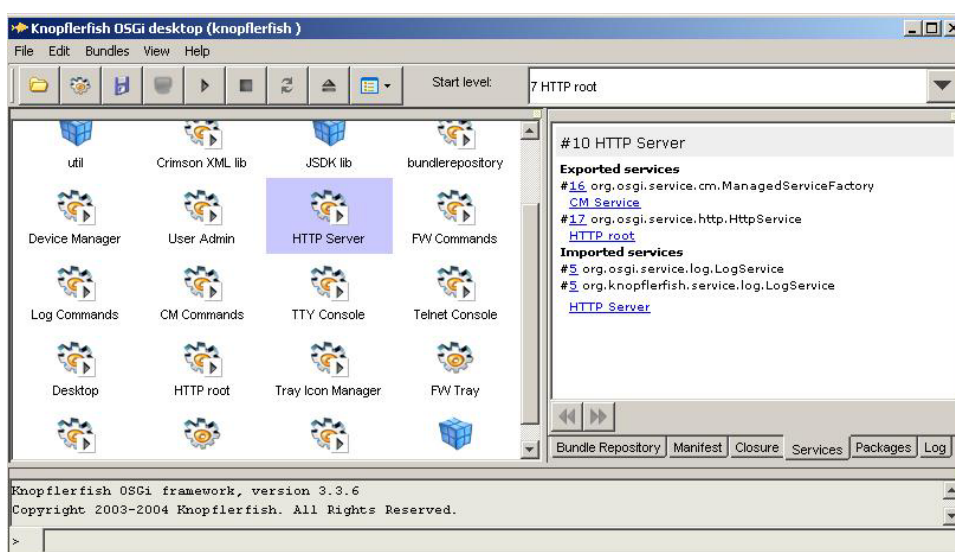


Abbildung 32 Der Knopflerfish Desktop

Als sehr nützlich bei der Entwicklung der Anwendung hat sich die Knopflerfish Konsole erwiesen. Neben Befehlen über die Standard Ein- und Ausgabe (*stdin und stdout*) bietet sie die Möglichkeit, Befehle über TCP an das Framework zu senden. Die Bundles werden in Eclipse entwickelt, als JAR verpackt und über Telnet an das Framework gesendet. Ein Aufruf des Update Befehls veranlasst das Framework, das Bundle zu stoppen, das Bundle JAR neu zu laden, und das Bundle neu zu starten. Dank Telnet Unterstützung durch das Build Tool Ant lässt sich der gesamte Deploy Prozess automatisieren.³¹

7.2.2 Design

Über eine Anwendung auf dem Gateway will der Reparatur Assistent ein Software Update realisieren. Dafür wird das Gerät analysiert und eine Liste mit passender Software erstellt. Aus dieser wählt er ein Update aus und installiert es auf dem Gerät. Für die Erstellung der Update Liste müssen Informationen des Gerätes mit Informationen des Service Providers verglichen werden.

Durch den Einsatz der UPnP Technologie auf Geräteseite spielt es keine Rolle, wo die Waschmaschine steht. Sie wird über einen HTTP Client, also einen Web Browser, gesteuert. Durch den Einsatz der Web Technologie wird auch der Client unabhängig vom Standort. Jeder PC mit einem Browser, der eine Verbindung zum Heimnetz hat, kann als Steuergerät genutzt werden. Das Gateway spielt jetzt die Rolle des Vermittlers zwischen der UPnP Welt der Waschmaschine und der HTTP basierten Kommunikation auf Browser Seite. Es wird also eine Komponente benötigt, welche einerseits die UPnP Dienste der Waschmaschine kontrollieren kann, und andererseits dazu in der Lage ist, diese Kontrolle als HTML aufzubereiten. Diese Funktionalität wird durch ein HTTP Servlet und einen UPnP Control Point realisiert.³² Die OSGi Plattform besitzt eine eigene Servlet Engine, bei der Servlets registriert werden können. Der Control Point wird durch

³¹ Ant ist ein Java-basiertes Tool zum Bauen von umfangreichen Software Projekten. Es ist das Gegenstück des aus der C Welt bekannten *Make*. Da Ant Build Files im Gegensatz zu Make Files in gut lesbarem XML geschrieben werden, ist Ant aber ungemein komfortabler zu bedienen.

³²Ein HTTP Servlet ist eine serverseitige Softwarekomponente einer Web Anwendung, die in einem Applikationsserver läuft. Im Gegensatz zu statischen HTML Seiten generieren HTTP Servlets dynamische Antworten auf HTTP Requests. Da es sich bei einem Servlet um eine JAVA Klasse handelt, können Servlets auf den vollen Sprachumfang von Java zugreifen.

den sog. UPnP Base Driver unterstützt, der für die echte UPnP Kommunikation zuständig ist und im folgenden Kapitel beschrieben wird. Abbildung 33 zeigt die Hauptkomponenten der Service Anwendung.

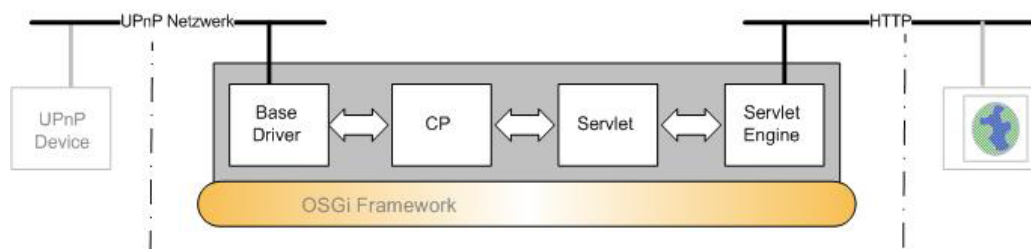


Abbildung 33 Kommunikation in der Service Anwendung

7.2.3 Base Driver

Der UPnP Base Driver ist ein Bundle, das die UPnP Welt mit der OSGi Welt verbindet. Es handelt sich um eine freie Implementierung von Domoware, welche die in der OSGi Spezifikation definierten UPnP Schnittstellen umsetzt. Sie baut ebenfalls auf den UPnP Bibliotheken von Cyberlink auf, die auch für die Implementierung der Waschmaschine verwendet wurden. [7.06]

Der Base Driver verbindet UPnP und OSGi in einer für Anwendungsentwickler sehr komfortablen Weise. Er besitzt eine Import und eine Export Funktion. UPnP Geräte im Netz werden vom Base Driver erkannt. Er übernimmt die Rolle eines Control Points, der über das UPnP Discover Protokoll Geräte selbstständig findet. Nach dem Discovery eines Gerätes erzeugt der Base Driver ein Java Objekt, welches das Gerät repräsentiert, und registriert es in der Service Registry. Durch die Registrierung in der OSGi Service Registry können alle Bundles, die auf der Plattform laufen auf diese Dienste zugreifen, ohne dass sie sich mit der eigentlichen UPnP Kommunikation auseinander setzen müssen. Das registrierte Service Objekt fungiert als Proxy für den eigentlichen UpnP Dienst. Bundles können die UpnP Dienste genauso nutzen, als wären es ganz normale OSGi Dienste. Umgekehrt können Bundle Dienste mithilfe des Base Drivers leicht im UPnP Netz veröffentlichen. Dafür müssen sie den Dienst lediglich unter der Schnittstelle *UPnPDevice* in der Registry anmelden. Der Base Driver erzeugt dann passende Gerätebeschreibungen und sendet diese an das UpnP Netz. Für UPnP Control Points erscheinen die so registrierten OSGi Dienste dann als

gewöhnliche UPnP Geräte. Die Service Anwendung nutzt ausschließlich die Import Funktion. Die beiden Base Driver Funktionen sind in Abbildung 34 gezeigt.

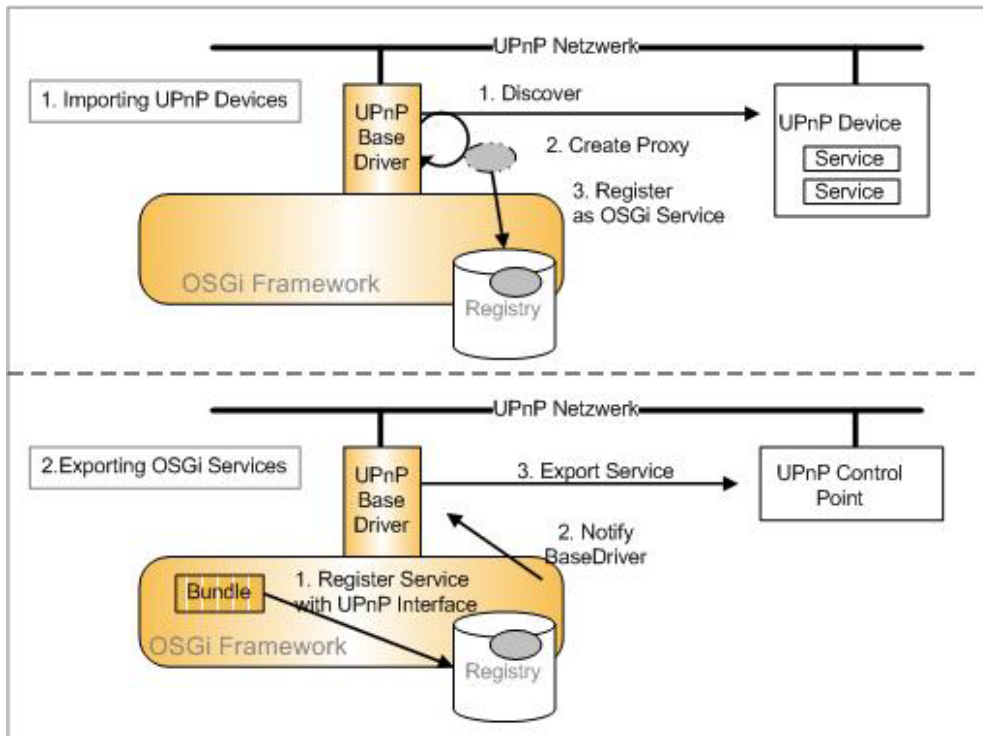


Abbildung 34 Base Driver Funktionalität

7.2.4 Ablauf

Der Ablauf des Update Vorgangs ist folgender. Der Service Mitarbeiter ruft eine Service URL des Gateways. Die Anfrage wird von einem HTTP Servlet verarbeitet. Das Servlet erfragt vom Control Point alle Geräte, die updatefähig sind und erzeugt aus diesen eine Liste, die als Antwort an den Mitarbeiter gesendet wird. Dieser wählt das betreffende Gerät aus und sendet eine zweite Anfrage an das Servlet. Nun erfragt das Servlet Informationen über den Control Point. Mit diesen Informationen, etwa Modell und Softwarestand, stellt es eine Anfrage nach möglichen Updates an den Service Provider. Aus der Antwort des Service Providers generiert das Servlet erneut eine Liste sendet es wiederum an den Service Mitarbeiter. Dieser wählt ein Update aus der Liste aus und sendet eine letzte Anfrage an das Servlet. Das Servlet lädt sich jetzt das tatsächliche Update vom Repository des Service Providers und aktualisiert damit

über den Control Point die Waschmaschine. Der Ablauf ist in Abbildung 35 dargestellt.

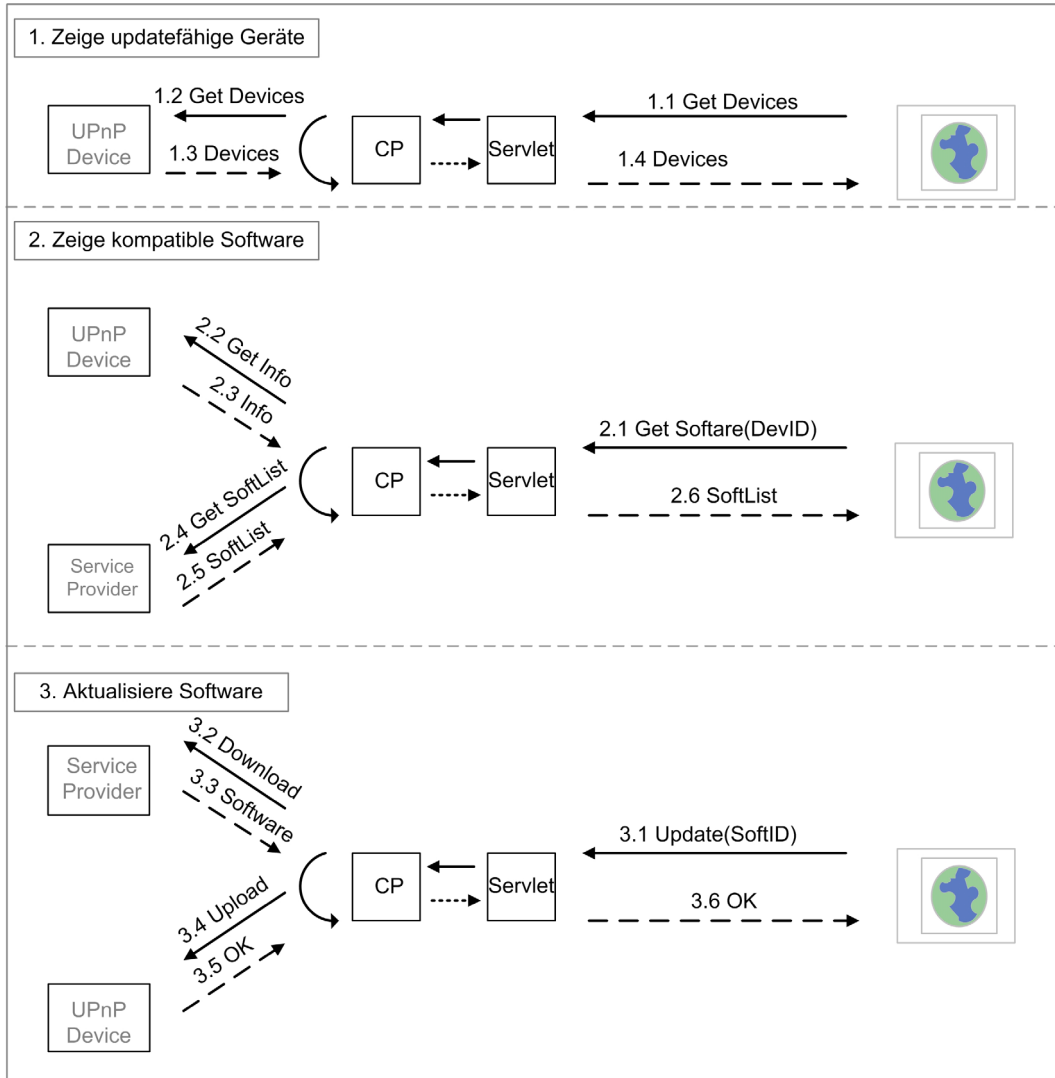


Abbildung 35 Ablauf des Updates in dre Schritten

Da sich der Service Provider nicht im lokalen Netz befindet, wird eine Möglichkeit benötigt, über das Wide Area Network (WAN) zu kommunizieren. Eine mögliche Lösung wäre die Kommunikation über Web Services. Eine andere Lösung ist ein WAN Zugangs Bundle für den Zugriff über JXTA. Die Entwicklung eines JXTA Bundles ist Teil einer weiteren Diplomarbeit, die sich während der Entstehung dieser Arbeit aber noch im Anfangsstadium befand. Die Verbindung zum Service Provider wird deshalb vorerst nicht realisiert.

7.2.5 Implementierung

Die Klasse ControlPoint ist für den Zugriff auf die Funktionen der Geräte zuständig. Dafür greift sie auf die Registry zu um UPnP Geräte zu ermitteln. Für jedes Gerät wird dabei geprüft, ob es einen UpdateDienst besitzt. Der ControlPoint meldet sich beim Framework als ServiceListener an, und wird damit über die Registrierung neuer Devices benachrichtigt. Er verfügt also immer über eine aktuelle Liste updatefähiger Geräte. Für die Registrierung und De-Registrierung von UPnP Geräten ist der Base Driver verantwortlich. Im Falle der UPnP Waschmaschine muss der Control Point einen HTTP Upload initiieren. Dieser wird in einer eigenen Klasse FileUpload realisiert, die einige Hilfsklassen der Apache Commons Bibliotheken zur Erzeugung einer HTTP Multipart Message nutzt. [7.07] Dadurch ist der Control Point eng mit der UPnP Waschmaschine verknüpft. Um das Servlet davon unabhängig zu machen, greift es nicht direkt auf den Control Point zu, sondern nutzt die Update Schnittstelle, die der Control Point implementiert. In dieser Schnittstelle sind die Methoden *getDevices*, *getSoftware(devId)* und *updateDevice(softId)* definiert, die unabhängig von der technischen Beschaffenheit des Updates sind. Wenn Geräte eine andere Art des Updates benötigen, muss die Control Point Implementierung ersetzt werden. Das Servlet ist davon nicht betroffen. Abbildung 36 zeigt das Klassenmodell der Service Anwendung.

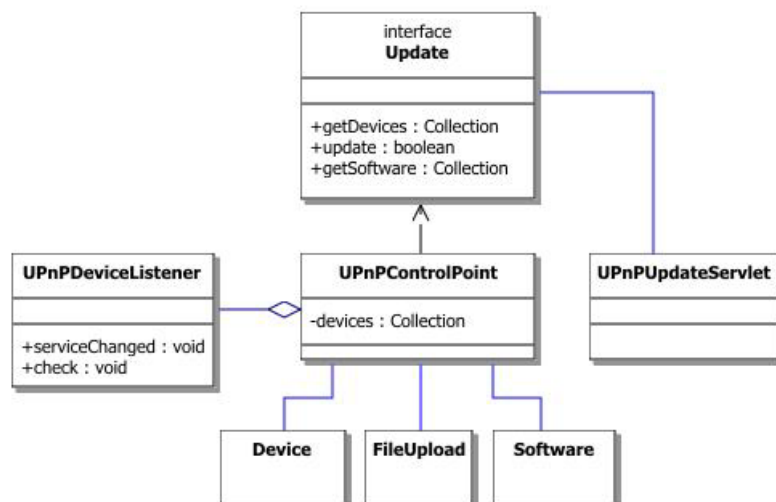
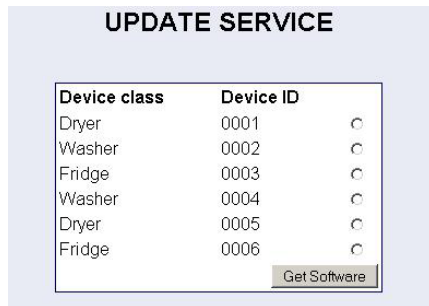


Abbildung 36 Klassendiagramm der Service Anwendung

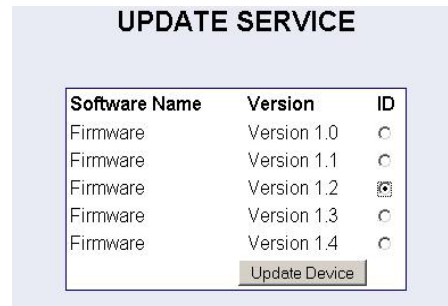
Für den Prototyp wird der Zugang zu einem RSP für den Abgleich nicht benötigt. Die Datei kommt von einem lokalen Software Repository. Abb. 37 und 38 zeigt die generierten HTML Seiten für die Anzeige der updatefähigen Geräte und der möglichen Software. Nach Drücken des Update Buttons (Abb.38) ruft das Servlet die Methode *update()* des Control Points auf, der daraufhin einen File Upload auf die Waschmaschine startet.



The screenshot shows a web page titled "UPDATE SERVICE". It contains a table with two columns: "Device class" and "Device ID". The table lists six devices: Dryer (0001), Washer (0002), Fridge (0003), Washer (0004), Dryer (0005), and Fridge (0006). To the right of each device ID is a radio button. Below the table is a button labeled "Get Software".

Device class	Device ID	
Dryer	0001	<input type="radio"/>
Washer	0002	<input type="radio"/>
Fridge	0003	<input type="radio"/>
Washer	0004	<input type="radio"/>
Dryer	0005	<input type="radio"/>
Fridge	0006	<input type="radio"/>

Abbildung 38 GetDevices



The screenshot shows a web page titled "UPDATE SERVICE". It contains a table with three columns: "Software Name", "Version", and "ID". The table lists six software versions: Firmware (Version 1.0), Firmware (Version 1.1), Firmware (Version 1.2), Firmware (Version 1.3), and Firmware (Version 1.4). To the right of each version is a radio button. Below the table is a button labeled "Update Device".

Software Name	Version	ID
Firmware	Version 1.0	<input type="radio"/>
Firmware	Version 1.1	<input type="radio"/>
Firmware	Version 1.2	<input checked="" type="radio"/>
Firmware	Version 1.3	<input type="radio"/>
Firmware	Version 1.4	<input type="radio"/>

Abbildung 37 GetSoftware

7.3 Weitere Entwicklung

WAN Zugang

Gegen Ende dieser Diplomarbeit wurde ein weiteres Projekt begonnen, die Entwicklung eines JXTA OSGi Base Drivers. Dies ist eine gute Möglichkeit für die Verbindung des Gateways mit einem Service Provider über das WAN. Als reines Kommunikationsprotokoll wäre der Einsatz von JXTA übertrieben, da es seinerseits SOAP für das Versenden der Nachrichten nutzt und sich dadurch nicht von Webservices unterscheidet. Der Vorteil von JXTA ist, dass der Ort des Service Providers nicht bekannt sein muss. Er wird über eine Art Gelbe Seiten dynamisch erfragt.

Bluetooth

Bisher ist der Service Mitarbeiter mit dem Netz über Ethernet verbunden. Sicherheitsaspekte werden in der bisherigen Anwendung nicht berücksichtigt. Es ist angedacht, dass der Service Mitarbeiter sich mit einem PDA über Bluetooth mit dem Gateway verbindet. Dadurch wäre gewährleistet, dass die Kommunikation über das Gateway läuft. Das Gateway könnte dann Sicherheitsaufgaben wie den Schutz des restlichen Netzes übernehmen. Dafür benötigt die

Zielplattform ein Bluetooth Modul. Zusätzlich wird ein OSGi Bluetooth Treiber ähnlich wie dem UPnP Base Driver benötigt. Soll der Client des Service Mitarbeiters weiterhin ein Browser sein, muss Bluetooth HTTP unterstützen.

Embedded Linux

Die Service Anwendung auf Basis der Knopflerfish Plattform läuft bisher auf einem Windows oder Linux PC. OSGi ist aber eigentlich für Embedded Devices ausgelegt. Da als Gateway in einem Heimnetz in den seltensten Fällen ein PC, sondern vielmehr ein DSL Router verwendet wird, liegt es auf der Hand, dass dies die Zielplattform ist. Inzwischen gibt es immer mehr solcher Geräte, auf denen eine spezielle, abgespeckte Linux Version läuft. Deshalb wurde in einer weiteren Diplomarbeit damit begonnen, OSGi auf ein Embedded Device zu portieren. Auf dem Zielgerät wird eine virtuelle Maschine benötigt, welche die von OSGi benötigten Bibliotheken unterstützt.

8 Schlussbetrachtungen

8.1 Zusammenfassung und Fazit

Ziel dieser Diplomarbeit war die Entwicklung eines Konzeptes für die Remote Konfiguration von Haushaltsgeräten. Es sollte eine Lösung gefunden werden, um Haushaltsgeräte über ein Netzwerk zu analysieren und mit neuer Software zu aktualisieren. Dabei sollte der Wunsch der Hersteller nach einer benutzerfreundlichen Lösung genauso berücksichtigt werden wie die Tatsache, dass in heutigen Heimnetzen eine Vielzahl unterschiedlicher Technologien anzutreffen ist. Zudem sollte das Konzept nicht auf einen Hersteller begrenzt sein, sondern vielmehr offen für unterschiedliche Anwender sein.

Der Bedarf dieses Konzeptes entsteht durch den wachsenden Markt für Heimnetze mit steigender Funktionalität der vernetzten Geräte, zu denen inzwischen auch Haushaltsgeräte wie Waschmaschinen und Kühlschränke gehören. Moderne Geräte erlauben die Aktualisierung mit neuen Softwarediensten. Dies bringt die Notwendigkeit mit sich, für ein Software Update erforderliche Informationen bereitzustellen. Für das Konzept wurden folgende Schlüsselkomponenten benannt:

- Selbstorganisation der Netzkomponenten
- Integration unterschiedlicher Technologien und Protokolle
- Herstellerunabhängige Kennzeichnung von Geräten und Diensten

Verglichen mit den oben benannten Anforderungen scheinen diese Komponenten die richtige Antwort zu sein. Die gestellten Forderungen werden durch den Einsatz geeigneter Technologien erfüllt. UPnP als Technologie für Netzkomponenten, die sich selbst konfigurieren, entlastet den Benutzer. Für den Betrieb von UPnP Geräten sind keinerlei Einstellungen notwendig. Die Integration unterschiedlicher Technologien wird durch den Einsatz von OSGi als Technologie für das Gateway gut gelöst. Es ist in der Lage unterschiedliche Teilnetze transparent zu verbinden. Durch seine Unterstützung für Remote Control ist es zudem bestens für Remote Software Updates geeignet. Das semantische Modell schließlich beschreibt die Informationen in einer allgemeinen, gut lesbaren Form

und kann dazu genutzt werden, Wissen zwischen Herstellern auszutauschen. Gerade der Wunsch nach einer Lösung für alle kann durch das Modell unterstützt werden.

Bei der technischen Umsetzung wurden ausschließlich neue, freie Technologien verwendet. Knopflerfish als freie OSGi Implementierung, der UPnP Basedriver von Domoware und die Cyberlink UPnP Bibliotheken können frei verwendet werden. Der Prototyp der Service Anwendung erfüllt die Anforderungen des Konzeptes nach einfacher Bedienbarkeit durch den Nutzer und der Berücksichtigung der heterogenen IT Landschaft in Heimnetzen. Die Umsetzung des Clients als Web Anwendung erlaubt die Kontrolle der Waschmaschine mit einem Browser unabhängig vom Standort des Nutzers. Nicht berücksichtigt sind Aspekte, welche die Sicherheit betreffen. So erfolgt bisher keine automatische Prüfung des Service Mitarbeiters, wie sie in der Szenariobeschreibung genannt wurde.

Auch das semantische Modell, das mit OWL in einem maschinenlesbaren Format vorliegt, und damit vom Gateway genutzt werden kann, fand keinen Einsatz in der Service Anwendung. Es kann aber durchaus als Grundlage für die Kommunikation zwischen unterschiedlichen Herstellern dienen.

Man darf gespannt sein, wie die Entwicklung weitergeht. Ob sich vernetzte Waschmaschinen für die breite Masse wirklich durchsetzen bleibt abzuwarten. Der Fortschritt im Heimnetz ist allerdings nicht mehr aufzuhalten. Mit einer guten Lösung für die Fernwartung der Geräte können sich Anbieter von der Konkurrenz abheben und Kosten sparen. Gerade aber die Frage nach dem Schutz des Heimnetzes vor unbefugtem Zugriff wird meiner Meinung nach eine entscheidende Rolle spielen bei der Akzeptanz des Remote Eingriffs aus Sicht der Kunden.

8.2 Persönliches Fazit

Durch mein Studium der Medieninformatik konnte ich ein breites und aktuelles Informatikwissen aufbauen. Neben theoretischen und praktischen Grundlagen erlangte ich die Fähigkeit, mich in neue, mir unbekannte Technologien einzuarbeiten. Dies erwies sich während der Arbeit an dieser Diplomarbeit als sehr hilfreich. Das Thema Remote Konfiguration Management ist sehr umfangreich, so konnte

ich eine Vielzahl moderner Technologien kennen lernen. Dabei benötigten die jeweiligen Teilbereiche viel Zeit. So war die Erstellung einer Übersicht über die unterschiedlichen Technologien in Heimnetzen sehr aufwändig, obwohl sie im Ergebnis mehr der allgemeinen Hintergrundinformation dient.

Die Recherche über Remote Software Updates stellte sich als schwierig heraus, aufgrund der Tatsache, dass insgesamt nur sehr wenig Informationen darüber verfügbar sind, wie solche Updates heute schon gelöst werden.

Neu für mich war die Theorie von semantischen Modellen. Obwohl die eigentliche Service Ontologie sehr einfach ist, erforderte deren Entwicklung viel Vorarbeit. Neben der Einarbeitung in das Protégé Tool und OWL nahm die Suche nach einer Umwandlungsmöglichkeit von OWL nach UML mehr Zeit in Anspruch als geplant war. Zugunsten der Arbeit an der Ontologie habe ich bewusst auf eine umfangreiche Ausprogrammierung der Service Anwendung verzichtet. Die Beschäftigung mit der Ontologie erschien mir für meine fachliche Entwicklung nützlicher, als die Anwendung meiner Programmierkenntnisse.

Insgesamt war die Entscheidung für dieses Projekt sehr gut. Besonders befriedigend empfinde ich die Tatsache, dass die Diplomarbeit Basis für eine Weiterentwicklung des Konzepts ist. Bereits bei Abschluss dieser Arbeit beschäftigten sich bei Alcatel zwei weitere Diplomarbeiten im Umfeld des Remote Konfiguration Management auf Basis von OSGi.

9 Anhang

9.1 Abkürzungsverzeichnis

AP	Access Point
API	Application Programming Interface
BBS	Basic Service Set
BootTP	Bootstrap Protocol
CHAIN	Ceced Home Appliances Interoperating Network
CPU	Central Processing Unit
DCP	Device Control Protocol
DHCP	Dynamic Host Configuration Protocol
DIG	Description Logic Implementers Group
EES	Extended Service Set
EIB	European/Electrical Installation Bus
EJB	Enterprise Java Beans
EHS	European Home System
GENA	General Event Notification Architecture
GUI	Graphical user Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilating, and Air Conditioning System
IDG	Internet Gateway Device
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IrDA	Infrared Data Association
ISM	Industrial, Scientific and Medical
JAR	Java Archive
J2SE	Java 2 Platform, Standard Edition
J2ME	Java 2 Platform, Micro Edition
JINI	Java Intelligent Network Infrastructure
JMS	Java Message Service
KNX	Konnex Hausbus
LAN	Local Area Network
LON	Local Operating Network
MAC	Medium Access Layer
MMS	Multimedia Message Service
OSGi	Open Service Gateway Initiative
OS	Operation System
OSI	Open Systems Interconnection
OWL	Web Ontology Language
PAL	Protocol Adaption Layer
PDA	Personal Data Assistant
PLC	Powerline Communication
PnP	Plug And Play
PXE	Preboot Execution Environment

RDF	Ressource Description Format
RFC	Request For Comment
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSA	Repair Service Assistant
RSP	Repair Service Provider
SE	Semantic Engine
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
STP	Shielded Twisted Pair
TCP	Transport Control Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
UPnP	Universal Plug and Play
URL	Uniform Resource Location
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
UUID	Universal Unique Identifier
VM	Virtual Machine
W3C	World Wide web Consortium
WAN	Wide Area Network
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network
WiFi	Wireless Fidelity
WWW	World Wide Web
XML	Extensible Markup Language

9.2 Quellenverzeichnis

Chapter 2: Einführung

- [2.01] Das undoofe Haus, Tobit; www.tobit.de
- [2.02] Studie von Harris Interactive; www.harrisinteractive.com
- [2.03] Nokia White Paper: Device Management – enabling effortless adoption of mobile devices; 2002
- [2.04] Wikipedia: Softwareverteilung

Chapter 3: IT Landschaft heutiger Heimnetze

- [3.01] Home Services;
www.rd.francetelecom.com/en/technologies/ddm200202/print_index1.htm
- [3.02] IrDa; www.irda.org/index.cfm
- [3.03] Bluetooth SIG; www.bluetooth.com/about/
- [3.04] Kafka: „WLAN: Technik, Standards, Planung u. Sich.“; 2004; Kap. 2.4
- [3.05] Kafka: „WLAN: Technik, Standards, Planung u. Sich.“; 2004; Kap. 2.2.1
- [3.06] Merkle: „Digitale Funkkommunikation mit Bluetooth“; 2002; S.68
- [3.07] IEEE; www.ieee.org
- [3.08] Firewire; www.apple.com/firewire
- [3.09] Die Geschichte von Powerline; teltarif.de/i/powerline.html
- [3.10] Homeplug Standard; www.homeplug.com
- [3.11] Funknetz mit Sicherheitslücken
teltarif.de/arch/2002/kw43/s9081.html
- [3.12] Kafka: „WLAN: Technik, Standards, Planung u. Sich.“; 2004; Vorwort
- [3.13] Kafka: „WLAN: Technik, Standards, Planung u. Sich.“; 2004; S.131
- [3.14] Lonworks von Echelon; www.echelon.com
- [3.15] EIBA -European Installation Bus Association; www.eiba.org
- [3.16] EHSA -European Home System Association; www.ehsa.com
- [3.17] BCI Batibus Club International; www.batibus.com
- [3.18] Konnex Association ; www.konnex-knx.com
- [3.19] CIC - CEBus Industry Council; www.cebuse.org
- [3.20] LCN von Isendorf; www.lcn.de
- [3.21] Kell & Colebrook: Offene Systeme für die Gebäudeautomation:
Vergleich von LonWorks und KNX
- [3.22] Hausbusse; www.uni-weimar.de/~weiprech/hausbus/
- [3.23] Kurzfassung Studie LON / KNX
www.echelon.de/support/documentation/analysis/lon-knx_study.htm
- [3.24] UPnP Forum; www.upnp.org
- [3.25] White Paper: Understanding Universal Plug and Play; 2000;
www.upnp.org/resources/whitepapers.asp
- [3.26] JINI Home www.jini.org
- [3.27] CHAIN Plattform; www.ceced.org/ebusiness/CHAIN.html
- [3.28] Bosch Siemens System Serve@Home; www.serve-home.de
- [3.29] B. Essig: Beitrag zur rechnergestützten, funktionsbezogenen
Leittechnik-Projektierung, erläutert am Beispiel der Gebäudetechnik
S.12
- [3.30] Kraftwerk Kennzeichnungen System
www.cimap.de/downloads/kkspraes_2000.pdf
- [3.31] DIN 6779 Kennzeichnungssystematik für technische Produkte

Chapter 4: Reparatur Szenario

[4.01] Dobelmann: Konfigurationsmanagement von Endgeräten

Chapter 5: Semantik

- [5.01] Ontology Development 101: A Guide to Creating Your First Ontology
- [5.02] OWL Spezifikation; www.w3.org/2004/OWL/
- [5.03] A Practical Guide To Building OWL Ontologies
Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0
- [5.04] Protégé; protege.stanford.edu
- [5.05] Protégé User Guide; protege.stanford.edu/doc/users_guide
- [5.06] Protégé OWL Plugin; protege.stanford.edu/plugins/owl
- [5.07] GraphViz; www.co-ode.org/downloads/owlviz/
- [5.08] Tutorial; protege.stanford.edu/plugins/owl/tutorial/
- [5.09] Razer; www.ata.tu-harburg.de/~r.f.moeller/racer
- [5.10] Noy & Musen: Evaluating Ontology-Mapping Tools
- [5.11] Automatic Mapping of OWL Ontologies into Java
- [5.12] Semantic Web Foren; protege.stanford.edu/mail_archive/
- [5.13] DUET; codip.grci.com/Tools/Tools.html
- [5.14] ArgoUML; argouml.tigris.org/
- [5.15] UML Plugin; protege.stanford.edu/plugins/uml/

Chapter 6: OSGi

- [6.01] OSGi Alliance www.osgi.org
- [6.02] OSGi Alliance White Paper: About the OSGi Service Plattform, S.6
- [6.03] OSGi Spezifikation
- [6.04] OSGi Spezifikation, S.427
- [6.05] OSGi Alliance White Paper: About the OSGi Service
Plattform, S.8
- [6.06] OSGi Spezifikation, S.57
- [6.07] OSGi Spezifikation, S.203
- [6.08] OSGi Spezifikation, S.65

Chapter 7: Technische Umsetzung

- [7.01] Cyberlink UPnP Framework; www.cybergarage.org/net/upnp/java/
- [7.02] Satoshi Konno: Cyberlinc Java Programming Guide
- [7.03] Dobelmann: : Konfigurationsmanagement von Endgeräten
- [7.04] Knopflerfish; www.knopflerfish.org
- [7.05] Knopflerfish Desktop; www.knopflerfish.org/desktop.html
- [7.06] UPnP Base Driver von Domoware; domoware.isti.cnr.it
- [7.07] Apache Commons jakarta.apache.org/commons/
- [7.08] Chen & Gong: "Programming Open Service Gateways with Java
Embedded Server Technology"

9.3 Abbildungsverzeichnis

Abbildung 1	Anwendungsbereiche im Heimnetz.....	6
Abbildung 2	Technologien und Anwendungsbereiche im Heimnetz	14
Abbildung 3	WLAN Netzstruktur	20
Abbildung 4	UPnP Netzstruktur.....	25
Abbildung 5	Beteiligte Entitäten des Reparatur Szenario	34
Abbildung 6	Schematischer Aufbau der Gateway Kommunikation	35
Abbildung 7	Ablauf des Szenarios	37
Abbildung 8	Ontologie basierte Suchmaschine.....	40
Abbildung 9	Transitive Eigenschaft	43
Abbildung 10	Regel über Mutter/Kind Relation	43
Abbildung 11	Anonyme Subklassen	44
Abbildung 14	Regeldefinition in Protégé	46
Abbildung 15	Klassifizierung durch den Reasoner	47
Abbildung 16	Personen und Organisationen.....	49
Abbildung 17	Ontology Device classes.....	49
Abbildung 18	Regeln für updatefähige Geräte.....	50
Abbildung 19	Asserted Klassenmodell	51
Abbildung 20	Inferred Klassenmodell	52
Abbildung 21	Aus der Ontologie erzeugtes UML Klassenmodell	54
Abbildung 22	Laden von Klassen in OSGi	57
Abbildung 25	Lebenszyklus eines OSGi Bundles	58
Abbildung 26	OSGi Event System	60
Abbildung 27	Anwendungsfälle der Waschmaschine.....	64
Abbildung 28	Klassendiagramm der Waschmaschine.....	64
Abbildung 29	Sequenzdiagramm des File Uploads.....	67
Abbildung 31	Apache HTTP Uploader.....	69
Abbildung 32	Der Knopflerfish Desktop.....	70
Abbildung 33	Kommunikation in der Service Anwendung.....	72
Abbildung 34	Base Driver Funktionalität.....	73
Abbildung 35	Ablauf des Updates in dre Schritten	74
Abbildung 36	Klassendiagramm der Service Anwendung.....	75

9.4 CD

Inhalt der beigelegten CD:

- Diplomarbeit im PDF-Format als Druck- und Bildschirmversion
- PDF Quellen
- Entwicklungstools und Service Ontologie
- Service Anwendung:
 - Quellcode der Waschmaschine und der Service Anwendung.
 - Knopflerfish OSGi Framework
 - Domoware UPnP Basedriver
 - Cyberlink UPnP Framework