

Ein wissensbasiertes Framework zur flexiblen Konfiguration von AAL-Umgebungen

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

Dr.-Ing.

bei der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

von

Dipl.-Inform. Tom Zentek

Hauptreferent: Prof. Dr. Rudi Studer
Korreferent: Prof. Dr. Wilhelm Stork

Tag der mündlichen Prüfung: 5.2.2015

Danksagung

An dieser Stelle möchte ich all denen meinem Dank aussprechen, die mich bei dieser Arbeit unterstützt haben und ohne die sie nicht möglich gewesen wäre.

Ganz herzlich danke ich meinem Doktorvater Prof. Dr. Rudi Studer für eine ständige Unterstützung sowie eine kreative und professionelle Arbeitsumgebung, die es auch zuließ interdisziplinären Forschungsfragen nachzugehen. Weiterhin danke ich Prof. Dr. Wilhelm Stork für die Möglichkeit der Anbindung meiner Arbeit an die Anwendungsdomäne Ambient Assisted Living. Prof. Dr. Stefan Nickel und Prof. Dr. Jan Kowalski danke ich für die Begleitung meiner mündlichen Prüfung.

Weiterhin danke ich Prof. Dr. Catherina Burghart und Dr. Jürgen Bock für die Unterstützung meiner Arbeit, angefangen bei konstruktiven Gesprächen bis hin zu Diskussionen über technische Details. Dr. Asarnusch Rashid danke ich für die effiziente und motivierende Leitung der Forschungsgruppe, die kreative Unterstützung bei der Ideenfindung sowie die hilfreichen wissenschaftlichen Diskussionen.

Daneben möchte ich allen ehemaligen und aktuellen Kollegen am Lehrstuhl und am FZI für die stets kollegiale Atmosphäre danken. Ganz herzlich danke ich Peter Wolf, Markus Ewald, Christian Reichelt, Dr. Simone Braun und Dr. Valentin Zacharias für ihre Ratschläge und Diskussionen und den daraus erwachsenen Freundschaften.

Meinen Eltern danke herzlich ich für ihre fortwährende Unterstützung und ihr Interesse an meiner Arbeit.

Meiner Frau Regine Hornung und unserer Tochter Emilie danke ich von ganzem Herzen für ihre unermüdliche Unterstützung, ihre Liebe und Motivation.

Allen meinen lieben Freunden danke ich für die Ausdauer, Ruhe und Geduld, womit sie mir stets zur Seite standen und mich immer wieder aufgemuntert haben.

Kurzfassung

Die Kommunikation von Alltagsgegenständen miteinander sowie im „Internet der Dinge“ nimmt beständig zu und macht aus unserem Umfeld zunehmend eine intelligente Umgebung. Ambient Assisted Living (AAL) beschreibt eine Anwendungsdomäne dieser intelligenten Umgebungen, deren Ziel es ist ältere Menschen in Alltagssituationen zu unterstützen. Die Herausforderung ist es, die vielen heterogenen Komponenten in der AAL-Umgebung geeignet zu arrangieren, um auf die individuellen, schnell wechselnden Bedürfnisse der Nutzer einzugehen. Viele Hersteller bieten hierfür proprietäre Produkte oder Insellösungen an. Diese mangelnde Interoperabilität soll durch einen wissensbasierten Konfigurationsprozess überwunden werden und eine hohe Flexibilität beim Aufbau von AAL-Umgebungen sicherstellen.

Entlang der Methodik des Design Science Research und unterstützt durch die Fokusgruppen- und die Living Lab-Methode wurde ein wissensbasiertes Konfigurationsframework entwickelt. Die Analyse identifizierte elf Akteure, die insgesamt 63 Anforderungen an die Konfiguration der AAL-Umgebungen stellen. Weiterhin wurde die Konfigurierbarkeit von 43 AAL-Umgebungen und 55 AAL-Anwendungsfällen analysiert und daraus weitere Anforderungen abgeleitet. Eine abschließende Validierung und Gewichtung aller Anforderungen erfolgte durch 27 Domänenexperten.

Entsprechend der Analyseergebnisse wurde das Konfigurationsframework entwickelt und implementiert. Es unterstützt die Integration neuer Komponenten in die AAL-Umgebung entlang eines Konfigurationslifecycle. Der Kern des Konfigurationsframeworks ist eine ontologiebasierte Repräsentation aller relevanten Informationen. Diese Repräsentation ermöglicht eine flexible Unterstützung durch ein gemeinsames Verständnis und die Wiederverwendbarkeit der einzelnen Komponenten im Konfigurationslifecycle. Eine Steigerung der Flexibilität in AAL-Umgebungen durch das Framework wurde durch zehn Domänenexperten in der Evaluation nachgewiesen.

Diese Arbeit entwickelt geeignete Prozesse, Werkzeuge und ein ontologiebasiertes Austauschformat für Informationen zwischen den einzelnen Integrationsschritten sowie Unterstützung bei der Wiederverwendung und Automatisierung. Hierzu wurden die verschiedenen Vor- und Nachteile des Frameworks zur flexiblen Konfiguration von AAL-Umgebungen evaluiert. Es konnte gezeigt werden, dass ein wissensbasiertes Konfigurationsframework die Heterogenität in AAL-Umgebungen mindern kann.

Inhaltsverzeichnis

Danksagung	iii
Kurzfassung	v
1 Einleitung	1
1.1 Forschungsfragen	3
1.2 Veröffentlichungen	5
1.3 Überblick der Arbeit	7
2 Methodik der Arbeit	9
2.1 Phase 1: Analyse	13
2.1.1 Analyse der Akteure und ihrer Anforderungen	13
2.1.2 Analyse des Konfigurationsbedarfs bei AAL-Plattformen	15
2.1.3 Analyse des Konfigurationsbedarfs bei AAL-Anwendungsfällen	16
2.2 Phase 2: Design	19
2.2.1 Konzeption des Konfigurationsframeworks	20
2.2.2 Implementierung des Konfigurationsframeworks	22
2.3 Phase 3: Evaluation	22
2.3.1 Qualitätskriterien	23
2.3.2 Evaluation des Konfigurationsframeworks	32
2.4 Zusammenfassung	38
3 Anforderungen an flexible AAL-Umgebungen	41
3.1 Akteure in einer AAL-Umgebung	41
3.1.1 Endanwender	42
3.1.2 Entwickler	44
3.1.3 Deployer	45
3.1.4 AAL-Anbieter	46
3.1.5 Assistenz-Anbieter	46
3.1.6 Zusammenfassung	47
3.2 Anforderungen an flexible AAL-Umgebungen	48
3.3 Flexibilität von AAL-Plattformkandidaten	56
3.3.1 Sichtung der AAL-Plattformkandidaten	57
3.3.2 Auswertung der Plattformkandidaten	64
3.3.3 Zusammenfassung	74
3.4 Konfigurationsbedarf von AAL-Referenzanwendungsfällen	76
3.4.1 Validierung der Anwendungsfallrepräsentanten	76
3.4.2 Bedarfsextraktion	79
3.4.3 Zusammenfassung	95

3.5	Zusammenfassung	95
4	Konzeption des Konfigurationsframeworks	97
4.1	Der Konfigurationslifecycle	99
4.1.1	Entwicklung	100
4.1.2	Planung	101
4.1.3	Installation	103
4.1.4	Personalisierung	104
4.1.5	Wartung	105
4.2	Akteure des Konfigurationslifecycle	106
4.3	Konfigurierbare AAL-Services	108
4.3.1	Konfigurationskomplexität	109
4.3.2	Konfigurationsoptionen	111
4.3.3	Konfigurationsregeln	112
4.3.4	Konfigurationsinstanzen	113
4.3.5	Konfigurationsdefinitionen	115
4.3.6	Manifest	121
4.4	Ontologien	122
4.5	Zusammenfassung	125
5	Implementierung des Konfigurationsframeworks	127
5.1	Werkzeuge für die Entwicklung	127
5.2	Werkzeuge für die Planung	128
5.3	Werkzeuge für die Installation, die Personalisierung und die Wartung	133
5.4	Dokumentation und Schulungsunterlagen	138
5.5	Zusammenfassung	139
6	Evaluation des Konfigurationsframeworks	141
6.1	Evaluation der Qualitätsanforderungen an das initiale Konfigurationsframework	141
6.1.1	Ergebnisse des Evaluationstests	142
6.1.2	Ergebnisse der Anforderungsevaluation	143
6.1.3	Diskussion der Ergebnisse	148
6.2	Evaluation des finalen Konfigurationsframeworks	149
6.2.1	Ergebnisse der finalen Evaluation	149
6.2.1.1	Effektivität	150
6.2.1.2	Effizienz	155
6.2.1.3	Zufriedenheit	158
6.2.1.4	Vollständigkeit	160
6.2.2	Diskussion der Ergebnisse	161
6.3	Zusammenfassung	161
7	Zusammenfassung und Ausblick	165
7.1	Zusammenfassung	165
7.2	Ausblick	168
	Literaturverzeichnis	171
A	Material zu den Evaluationen	181
A.1	Integrationsanleitung für den „Nutritional Advisor“	181

A.2	Fragebogen zum Pretest	182
A.3	Fragebogen zu der Anforderungsevaluation	183
A.4	AHP-Berechnung	186
A.5	Fragebogen zur Frameworkevaluation	188
A.6	Konfigurationsaufgaben der Frameworkevaluation	189

Abbildungsverzeichnis

1.1	AAL-Umgebung mit ihren Komponenten und Akteuren	2
2.1	Methodik der Entwicklung des Konfigurationsframeworks	10
2.2	Angepasstes Product Quality Model mit Fragen für AAL	30
2.3	Vollständigkeits- und Konsistenzprüfung	30
3.1	Übersicht zu den einzelnen Akteuren	42
3.2	Bewertung der Plattformkandidaten	75
4.1	Überblick zum Konfigurationsframework	98
4.2	Überblick zum Konfigurationslifecycle	99
4.3	Überblick zur Entwicklung	100
4.4	Überblick zur Planung	101
4.5	Überblick zur Installation	103
4.6	Überblick zur Personalisierung	104
4.7	Überblick zur Wartung	106
4.8	Übersicht zu den einzelnen Akteuren	107
4.9	Übersicht zu den Komplexitätsklassen von AAL-Anwendungsfällen . .	110
4.10	Überblick über die Konfigurationsinstanzen	114
4.11	Überblick über die Konfigurationsdefinition	116
4.12	Überblick über das configItem	117
4.13	Ausschnitt der Profile-Ontologie	123
5.1	Darstellung des Configuration Editors in der einfachen Editoransicht .	129
5.2	Überblick zum DA-Client	130
5.3	Klassenbibliothek zwischen DA-Client und DA-Server.	131
5.4	Komponentendiagramm des DA-Servers	132

5.5	Darstellung des Control Centres und geöffnetem Ontologie Editor . . .	133
5.6	Darstellung des Konfigurationsfensters eines AAL-Service	134
5.7	Komponentendiagramm der Configuration-Komponente	136
5.8	Sequenzdiagramm der Konfigurationsabläufe	137
6.1	Auswertung der Evaluation	142
6.2	Überblick zu den Teilnehmern (I)	144
6.3	Überblick zu den Teilnehmern (II)	144
6.4	Prioritätsverteilung der Einzelwertungen der Teilnehmer	145
6.5	Überblick zu den Fragen der Gruppe 1	146
6.6	Überblick zu den Fragen der Gruppe 2	146
6.7	Überblick zu den Teilnehmern (I)	150
6.8	Überblick zu den Teilnehmern (II) - AAL-Erfahrung in Jahren	150
6.9	Auswertung der Ausgaben der Hands-On-Phase	152
6.10	Leistungsfähigkeit Q5	153
6.11	Erfasste Bedienfehler auf Basis der Videoanalyse (n=22)	153
6.12	Auswertung der Aufgabenbearbeitung	154
6.13	Durchschnittlicher Aufwand zum Entwickeln und Integrieren (n=9) .	156
6.14	Ausführungszeiten für Aufgabe 1 und 2 ohne Testen	157
6.15	Gemessener Speed-Up	158
6.16	Umsetzung Q3	158
6.17	Komplexität Q7	159
6.18	Auswertung der Zufriedenheit	159
6.19	Funktionalität Q4	160
A.1	Fragebogen Seite 1	183
A.2	Fragebogen Seite 2	184
A.3	Fragebogen Seite 3	185
A.4	AHP-Berechnung	187
A.5	Fragebogen Seite 1	188
A.6	Konfigurationsaufgaben Seite 1	189
A.7	Konfigurationsaufgaben Seite 2	190
A.8	Konfigurationsaufgaben Seite 3	190
A.9	Konfigurationsaufgaben Seite 4	191

A.10 Konfigurationsaufgaben Seite 5	191
A.11 Konfigurationsaufgaben Seite 6	192
A.12 Konfigurationsaufgaben Seite 7	192
A.13 Konfigurationsaufgaben Seite 8	193
A.14 Konfigurationsaufgaben Seite 9	193
A.15 Konfigurationsaufgaben Seite 10	194
A.16 Konfigurationsaufgaben Seite 11	194
A.17 Konfigurationsaufgaben Seite 12	195
A.18 Konfigurationsaufgaben Seite 13	195
A.19 Konfigurationsaufgaben Seite 14	196
A.20 Konfigurationsaufgaben Seite 15	196
A.21 Konfigurationsaufgaben Seite 16	197
A.22 Konfigurationsaufgaben Seite 17	197
A.23 Konfigurationsaufgaben Seite 18	198

Tabellenverzeichnis

2.1	Fokusgruppen in Phase 1	13
2.2	AAL-Projekte mit den dazugehörigen Anwendungsfällen	17
2.3	Fokusgruppen in Phase 2	19
2.4	Fokusgruppen in Phase 3	23
2.5	Kernfragen des Evaluationsworkshops	33
2.6	KLM-Operatoren mit dazugehörigen Zeiten nach [Kier01]	36
3.1	Verstöße der Plattformkandidaten der Industrie	58
3.2	Verstöße der Plattformkandidaten der Open Source	59
3.3	Verstöße der Plattformkandidaten der Forschung	59
3.4	Bewertung der Kategorie Anwendungsfälle	65
3.5	Bewertung der Kategorie Anwendungsfälle	67
3.6	Bewertung der Kategorie Hardware	69
3.7	Bewertung der Kategorie Support	71
3.8	Bewertung der Kategorie Werkzeuge	73
3.9	Referenzanwendungsfälle und ihre Anwendungsfallkategorien	78
3.10	Konfigurationsbedarf: AALS I: Health Management	80
3.11	Konfigurationsbedarf: AALS II: Nutritional Advisor	81
3.12	Konfigurationsbedarf: AALS III: Agenda and Reminder	82
3.13	Konfigurationsbedarf: AALS IV: Safety and Security at Home	83
3.14	Konfigurationsbedarf: AALS V: Help when Outdoors	84
3.15	Konfigurationsbedarf: AALS VI: Long Term behaviour Analyser	85
3.16	Konfigurationsbedarf: AALS VII: AALfficiency	86
3.17	Konfigurationsbedarf: AALS VIII: Food and Shopping	87
3.18	AALS IX: Personal Safety	88
3.19	Konfigurationsbedarf: AALS X: Medication Manager	89

3.20	Wissensobjekte: AALS I: Health Management	90
3.21	Wissensobjekte: AALS II: Nutritional Advisor	91
3.22	Wissensobjekte: AALS III: Agenda and Reminder	91
3.23	Wissensobjekte: AALS IV: Safety and Security at Home	92
3.24	Wissensobjekte: AALS V: Help when Outdoors	92
3.25	Wissensobjekte: AALS VI: Long Term behaviour Analyser	93
3.26	Wissensobjekte: AALS VII: AALfficiency	93
3.27	Wissensobjekte: AALS VIII: Food and Shopping	93
3.28	Wissensobjekte: AALS IX: Personal Safety	94
3.29	Wissensobjekte: AALS X: Medication Manager	94
4.1	Zuordnung der Referenzanwendungsfälle zu den Komplexitätsklassen	110
6.1	Darstellung des Mehraufwands zwischen Ausgabe 1 und 2	156

1. Einleitung

Die Etablierung des „Internet der Dinge“ (engl. Internet of Things) durchsetzt unsere Lebensumgebungen immer mehr [RivdM14]. Überall um uns herum werden Gegenstände des täglichen Lebens in die Lage versetzt zu kommunizieren. Diese Kommunikation von Alltagsgegenständen miteinander und mit dem Internet macht aus unserem Umfeld zunehmend eine „intelligente Umgebung“ (eng. Smart Environment). Das Ziel dieser Umgebungen ist es, durch intelligente Geräte die täglichen Bedürfnisse der Menschen in dieser Umgebung zu unterstützen [CoDa04]. Erreicht wird dieses Ziel durch die Kombination verschiedener Technologien. Hardware in Form von Sensoren und Aktuatoren werden über Middleware-Technologien miteinander verbunden und durch menschliche Dienstleistungen erweitert.

Ambient Assisted Living (AAL) beschreibt eine Anwendungsdomäne von intelligenten Umgebungen, die darauf spezialisiert ist, hilfsbedürftigen, oft älteren Menschen sowie ihren formellen und informellen Betreuern im Alltag Unterstützung zu bieten. AAL ist ein „hybrides Produkt: (1) eine technische Basisinfrastruktur im häuslichen Umfeld [...] und (2) Dienstleistungen durch Dritte“ [VDE 12a, S.14]. AAL ist ein sehr junges Forschungsfeld, das in den letzten Jahren durch den demografischen Wandel, die steigenden Kosten im Gesundheitssystem und den Ärzte-/Pflegemangel zunehmend an Bedeutung gewinnt.

Der medizinische Fortschritt sowie sinkende Geburtenraten sind die Hauptgründe für die Veränderung der Altersstruktur in Deutschland. Die steigende Lebenserwartung führt dazu, dass im Jahr 2030 auf zwei Erwerbstätige ein Rentner kommt. Im Jahr 1970 war das Verhältnis noch 4:1. Dies bedeutet einen Anstieg der über 65-jährigen von 33% [Stat11]. Mit dem medizinischen Fortschritt gehen allerdings immense Kosten einher. So stiegen beispielsweise die durchschnittlichen Gesundheitsausgaben in Deutschland zwischen 2003 und 2011 um fast 1000 Euro auf 3660 Euro pro Kopf und Jahr [Stat14]. Diese Aspekte und das zunehmend höhere Durchschnittsalter der Ärzteschaft führen trotz Zunahme der Anzahl der Medizinstudierenden zu einem Mangel an medizinischem Personal [Rich10].

Als *AAL-Umgebung* bezeichnet man die Umsetzung von AAL als Anwendungsdomäne der intelligenten Umgebungen. Die Abbildung 1.1 veranschaulicht die Kernkomponenten und Akteure einer AAL-Umgebung. Die drei Kernkomponen-

ten der AAL-Umgebung sind die *AAL-Plattform* mit an ihr angeschlossener *AAL-Hardware* und *AAL-Software*. Die AAL-Plattform ist eine Middleware zur Abstraktion im Sinne der Service Orientierten Architektur (SOA). Im Besonderen unterstützt sie die AAL-spezifischen Hardwareanforderungen an die *Sensoren* und *Aktuatoren* (Kurzform: *Aktor*) in Form von Geräten (z.B. Medizintechnik, Gebäudesensorik) und Protokollen. Die von den Sensoren gelieferten Informationen werden in der AAL-Plattform aufbereitet und an die AAL-Software weitergegeben. Diese kann nun entsprechend der Bedürfnisse des *Endanwenders* die Daten verarbeiten und gegebenenfalls über die Plattform Aktoren auslösen. Das Zusammenspiel der AAL-Hardware, AAL-Software und externer Dienstleistungen zur Bedürfnisbefriedigung der Endbenutzer wird als *AAL-Anwendungsfall* bezeichnet. Die Implementierung eines AAL-Anwendungsfalls obliegt dem *Entwickler*, der selbst nicht Teil der AAL-Umgebung ist. Der *Deployer* stellt eine einwandfreie Funktion der AAL-Umgebung sicher, indem er alle drei Komponenten der Umgebung managt.

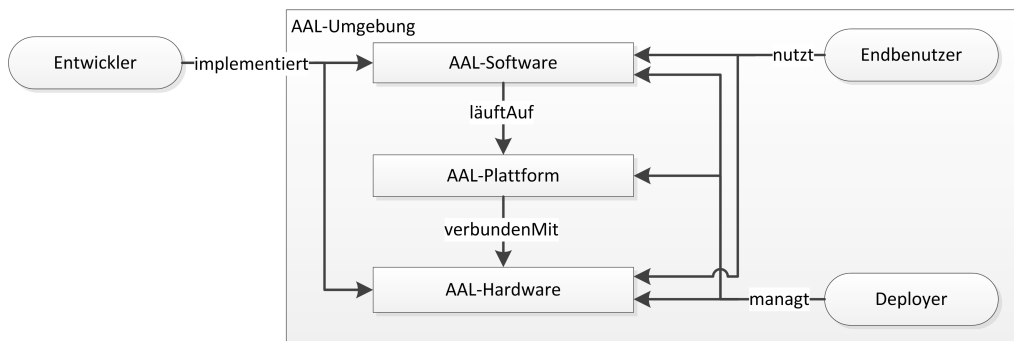


Abbildung 1.1: AAL-Umgebung mit ihren Komponenten (eckige Kästen) und Akteuren (runde Kästen)

Ein typisches Beispiel für einen AAL-Anwendungsfall ist die Ernährungsunterstützung älterer Menschen (Endanwender). Eine ausgewogene Ernährung und ausreichende Flüssigkeitsaufnahme stellen, gerade in Kombination mit verschiedenen Erkrankungen im hohen Alter eine Herausforderung dar. Ohne Unterstützung kann ein selbständiges Leben so stark eingeschränkt sein [BWRAA⁺07].

Eine AAL-Umgebung kann hierbei beispielsweise einen Menschen mit Diabetes auf verschiedene Arten unterstützen. Durch die Sensorinformationen aus der AAL-Umgebung kann das System an die Nahrungszubereitung und -einnahme erinnern. Im Vorfeld kann durch Überwachung der Lebensmittel und im Abgleich zum medizinischen Profil des Endbenutzers sichergestellt werden, dass passende Lebensmittel ausreichend vorhanden sind. Sollte dies nicht der Fall sein, kann rechtzeitig durch eine Information an den Endbenutzer oder an einen Dienstleister die Beschaffung angestoßen werden. Auch während des Zubereitens der Mahlzeit kann das System durch seine Sensoren dem Ablauf folgen und, wo notwendig, unterstützend eingreifen. Weiterhin können sogar Gefahrensituationen (z.B. vergessener eingeschalteter Herd) erkannt und entsprechend deeskaliert werden. Alle AAL-Anwendungsfälle sind bedarfsgesteuert, so dass der Bewohner (gehört zur Gruppe der Endbenutzer) an „guten Tagen“ vom Unterstützungssystem nichts mitbekommt, aber sich sicher sein kann, bei Bedarf unterstützt zu werden.

Im Rahmen des EU-Projektes universAAL¹ wurde eine Referenzarchitektur für AAL-Umgebungen geschaffen. In Ihr können AAL-Anwendungsfälle wie die Ernährungsberatung (eng: Nutritional Adviser) aus dem Beispiel unterstützt werden. Diese Arbeit begleitete das Projekt von Anfang bis Ende der vierjährigen Laufzeit (2010 - 2014) und nutzt die in universAAL entstandene Referenz-AAL-Plattform. Das in dieser Arbeit entstandene Konfigurationsframework² wurde in das universAAL Referenzmodell übernommen.

1.1 Forschungsfragen

Den AAL-Anwendungsfällen kommt eine zentrale Aufgabe bei der Bedienung der Endanwenderbedürfnisse zu. Ein Anwendungsfall unterstützt eine konkrete Assistenzsituation durch die Kombination von Hardware, Software und menschliche Dienstleistungen. Der Begriff eines Anwendungsfalls ist im Bereich von AAL nicht leicht zu definieren. Dennoch zeigen verschiedene Arbeiten ([Empi07, Wald07, Baum08, IsGVG08a, Robl08]) im Kern gleiche Konzepte, die eine allgemein gültige Definition erlauben:

“A concrete assistance value brought to elderly people (or people in situation of dependency and lack of autonomy like disabled) and / or their care network to help them live a full and independent life. To bring such a value, any combination of hardware, software, and human effort may be incorporated.” [Sala11]

Dies bedeutet ein AAL-Anwendungsfall bringt älteren Menschen (oder Menschen mit Einschränkungen) und/ oder ihrem Pflegenetzwerk eine konkrete Unterstützung für ein unabhängiges Leben. Erreicht wird diese Unterstützung durch verschiedene Kombinationen von Hardware, Software und menschlichem Aufwand in einem Anwendungsfall. Die technische Implementierung eines AAL-Anwendungsfalls bezeichnet man als AAL-Service. Sie stellt die zentrale Komponente in AAL-Umgebungen zur flexiblen Erweiterung derselben mit neuen Funktionen dar.

Das Management einer AAL-Umgebung ist somit in erster Linie das Management von AAL-Anwendungsfällen. Bezüglich dieser Aufgaben stehen zwei Herausforderungen besonders hervor:

Heterogenität von AAL-Services: Im Alltag sollen bei einem Endbenutzer AAL-Komponenten verschiedener Entwickler eingesetzt werden, damit sie sich die Ressourcen der AAL-Umgebung teilen können. Auch die gemeinsame Nutzung von Daten zwischen verschiedener AAL-Hardware und AAL-Software sollte möglich sein. Bisher entwickeln Hardwarehersteller und Technologieanbieter stand-alone-Lösungen mit ihren eigenen Datenrepräsentationen und proprietären Standards.

¹<http://www.universAAL.org> (Zugriff am 21.11.2014)

²Da es sich bei universAAL um ein EU-Projekt handelt, sind Teile dieser Arbeit in Englisch gehalten und an den entsprechenden Stellen übersetzt.

Individuelle und schnell wechselnde Bedürfnisse: Der Endanwender hat individuelle Anforderungen bezüglich seiner häuslichen Umgebung oder seiner familiären Umstände. Weiterhin finden sich vielfältige Bedürfnisse entsprechend seines medizinischen Zustandes, seiner individuellen Vorlieben oder physischen und psychischen Leistungsfähigkeit. Gerade in Bezug auf den Gesundheitszustand und die generelle Verfassung ergeben sich schnell verändernde Bedürfnisse bei den älteren Menschen.

Diesen beiden Herausforderungen kann mit einer flexiblen AAL-Umgebung begegnet werden. Das Maß für Flexibilität ist, wie einfach sich die Systemmöglichkeiten an die externen Anforderungen anpassen lassen („how easily a system’s capabilities can be modified in response to external changes“ [RyJC13]). Hieraus ist zu schließen, dass Flexibilität das entscheidende Erfolgskriterium zum erfolgreichen Management von AAL-Umgebungen ist. Eine Lösung bietet die ontologiegebaute Konfiguration als Teilgebiet der wissensbasierten Konfiguration [Stum97]. Durch ihre Kerneigenschaften bieten Ontologien [Grub95] den entscheidenden Vorteil:

- Sie stellen ein gemeinsames Verständnis des Wissens über die AAL-Umgebung und ihrer Komponenten sowie Akteure her.
- Wiederverwendbare Konzepte sichern die Interoperabilität von verschiedenen AAL-Anwendungsfallkomponenten.

Mit dem Ziel, die Flexibilität des Managements von AAL-Umgebungen zu steigern, wird in dieser Arbeit die Verwendung der ontologiebasierten Konfiguration untersucht. Hierzu bedarf es einer Aufstellung der beteiligten Akteure und ihrer Anforderungen. Weiterhin sind AAL-Plattformen und AAL-Anwendungsfälle auf ihren Konfigurationsbedarf hin untersucht werden.

Bisherige Arbeiten [opti13, Nara13, Raal13] zu AAL-Plattformen greifen zu kurz, um ein flexibles Management von AAL-Umgebungen zu ermöglichen, oder setzen andere Akzente in ihren Untersuchungen. Somit konnte nicht auf ihnen aufgebaut werden. Die relevanten bisherigen Vorarbeiten sind:

Das Projekt optimAAL [opti13] hatte zum Ziel, eine tragfähige, erweiterbare Kompetenzplattform zu entwickeln. Hierbei fokussierte man sich auf vier Kategorien (Domänenwissen, Referenzlösungen, Entwicklungsmethoden und Evaluationen), die bei der Einführung und Entwicklung von AAL-Umgebungen helfen sollen. Zur Bewertung der einzelnen Technologien wird ein Qualitätsmodell entworfen. Im Zuge der Entwicklung des Bewertungsmodells wurden verschiedene AAL-Plattformen untersucht. Es wurden 14 AAL-Plattformen hauptsächlich Produkte aus Forschungsprojekten betrachtet. Diese AAL-Plattformen weisen eine fixe Konfiguration der Anwendungsfälle auf.

Das EU-Projekt universAAL [Nara13] entwickelte ein Referenzmodell für AAL-Plattformen. Hierzu wurden die beteiligten Rollen in AAL-Umgebungen und ihre Anforderungen identifiziert. Die Konsolidierung von geeigneten Systemen bezog sich aber nur auf acht Plattformen aus der Forschung. Die Analyse dieser Kandidaten erfolgte sehr intensiv, jedoch hauptsächlich auf technischer Ebene. Diese Arbeit

entstand im Rahmen dieses Projektes und erweitert die universAAL Referenzarchitektur um Funktionen zur flexiblen Konfiguration.

RAALI [Raal13] ist ein EU Projekt mit dem Ziel, Interoperabilität zwischen AAL-Plattformen und anderen Systemkomponenten in der AAL-Umgebung zu schaffen. Dafür wurde eine Sichtung der verfügbaren AAL-Plattformen vorgenommen. Die bisherigen Ergebnisse stellen keine tiefe Metaanalyse von AAL-Plattformen dar [LMDM⁺12]. Eine flexible Konfiguration wurde nicht untersucht.

Da somit keine systematische Arbeiten zum flexiblen Management von AAL-Umgebungen existieren, widmet sich der erste Teil der Arbeit der Frage nach den grundsätzlichen Anforderungen:

Forschungsfrage 1: *Was sind die Anforderungen an ein flexibles Management von AAL-Umgebungen und die damit verbundene Konfiguration?*

Diese Forschungsfrage wird eingehend in Kapitel 3 behandelt und wird beantwortet mit einer Liste von Anforderungen, die in Kapitel 6.1 evaluiert und gewichtet werden.

Aufbauend auf der ersten Frage und den gefundenen Anforderungen soll durch die ontologiebasierte Konfiguration die Möglichkeit geschaffen werden, vorhandene heterogene Strukturen aufzulösen, um ein flexibles Management der AAL-Umgebung zu ermöglichen. Bisherige konzeptionelle Ansätze [WBKL⁺13, Kirw13] zu einer Interoperabilität der AAL-Anwendungsfälle sind als nicht ausreichend zu bewerten. Sie beschäftigen sich lediglich mit einzelnen Aspekten oder Teilen des AAL-Anwendungsfalls. Dennoch müssen Ansätze aus Smart Home [NoMo06, KaNF08, CTPRC09] und der wissensbasierten Konfiguration [YMWZ09, ABBC⁺11, DoYS11] berücksichtigt werden. Ein effizienter und effektiver Umgang mit AAL-Anwendungsfällen über den gesamten Lebenszyklus in der AAL-Umgebung wurde bisher nicht betrachtet. Es ist zu untersuchen, wie ein Konfigurationsframework hierbei die verschiedenen Akteure unterstützen kann. Weiterhin muss die Zufriedenheit aller beteiligten Akteure sichergestellt werden und die Konzepte müssen vollständig sein. Die zweite Forschungsfrage lautet somit:

Forschungsfrage 2: *Welche Vor- und Nachteile bietet eine ontologiebasierte Konfiguration von AAL-Anwendungsfällen für ein flexibles AAL-Umgebungsmanagement?*

Im Kapitel 4 wird Forschungsfrage 2 durch die Konzeption eines Konfigurationsframeworks beantwortet. Im Kapitel 5 wird die Implementierung des Konfigurationsframeworks dargestellt. Diese Implementierung wird in Kapitel 6.2.1 für die Evaluation genutzt.

1.2 Veröffentlichungen

Die Ergebnisse dieser Arbeit wurden in mehreren Publikationen veröffentlicht.

Analysemodell: Die Entwicklung eines Analysemodells für AAL und die daraus gewonnenen Anforderungen in Kapitel 3 wurden in dem Buch *Ambient Assisted Living* veröffentlicht [ZYRR14]:

Zentek, Yumusak, Reichelt, Rashid (2014). Which AAL Middleware matches my requirements? An Analysis of Current Middleware Systems and

a Framework for Decision-Support. In: Ambient Assisted Living, Publisher: Springer Berlin Heidelberg, Editors: Wichert, Reiner

Konfigurationsframework: Das konzipierte und entwickelte Framework aus Kapitel 4 wurde erstmals beschrieben in [ZRWK09]. Die zentrale Idee einer hybriden Konfiguration aus semantischen und herkömmlichen Konzepten ist veröffentlicht in [ZeWo11]:

Zentek, & Wolf (2011). Ontologiebasierte Konfigurationsprozesse in AAL-Umgebungen. In INFORMATIK 2011 Lecture Notes in Informatics, Band P192 (pp. 172). Berlin.

Das gesamte Framework mit seinem Lifecycle und Ausschnitte der dazugehörigen Evaluation aus Kapitel 6 findet sich in dem Buch „Ambient Assisted Living“ [ZeMR13]:

Zentek, Marinc, Rashid (2013). Methods and Tools for Ontology-Based Configuration Processes of AAL Environments. In: Ambient Assisted Living (pp. 213-224). , Publisher: Springer Berlin Heidelberg, Editors: Wichert, Reiner and Klausning, Helmut

Die Nutzung des Konfigurationsframeworks sowie seine Einbindung in das AAL-Referenzmodell von universAAL wurde auf dem *International Symposium on Ambient Intelligence (ISAmI)* publiziert [RFGIS⁺13]:

Ram, Furfari, Girolami, Ibanez-Sanchez, Lazaro-Ramos, Mayer, Zentek (2013). universAAL: Provisioning Platform for AAL Services. In A. van Berlo, K. Hallenborg, J. M. C. Rodriguez, D. I. Tapia, & P. Novais (Eds.), ISAmI (Vol. 219, pp. 105–112). Springer Berlin Heidelberg.

Weiterhin wurden im Rahmen dieser Arbeit Veröffentlichungen zur Living Lab-Methode im Bereich von AAL publiziert [RaZS11, RRRZ12].

Weitere Veröffentlichungen zeigen, dass sich die entwickelten Konfigurationskonzepte teilweise auf benachbarte Domänen der Telemedizin [ZeRa11] und Pflege [ZeRa12] zur Steigerung der Interoperabilität übertragen lassen.

Die wissenschaftlichen Herausforderungen dieser Arbeit wurden im Rahmen eines „doctoral consortium“ diskutiert, welches im Rahmen der Konferenz AAL Forum 2010 in Odense, Dänemark abgehalten wurde.

Das im Rahmen dieser Arbeit entwickelte Framework ist Teil der Referenzarchitektur für AAL-Umgebungen geworden und in seiner Open Source Implementierung als SDK³ verfügbar.

³Software Development Kit

1.3 Überblick der Arbeit

Kapitel 1, *Einleitung*, motiviert die Notwendigkeit von flexiblem Management in AAL-Umgebungen. Der Stand der Forschung ist nicht in einem separaten Kapitel zusammengefasst, sondern findet sich zum besseren Verständnis eingereiht im dazugehörigen Kapitel wieder.

Kapitel 2, *Methodik der Arbeit*, beschreibt das wissenschaftliche Rahmenwerk. Es werden die in der Analyse verwendeten Methoden sowie die Vorgehensweise bei der Konzeption dargestellt. Abschließend beinhaltet dieses Kapitel die Evaluationsmethode.

Kapitel 3, *Anforderungen an flexible AAL-Umgebungen*, analysiert die Akteure, Plattformen und Services sowie die jeweiligen Anforderungen.

Kapitel 4, *Konzeption des Konfigurationsframeworks*, ist der Entwurf des Frameworks mit seinen Komponenten in Form vom Konfigurationslifecycle, den beteiligten Akteuren, den angepassten AAL-Services und den Ontologien.

Kapitel 5 und 6, *Implementierung des Konfigurationsframeworks* und *Evaluation des Konfigurationsframeworks*, beschreiben die Umsetzung des Konfigurationsframeworks sowie die Evaluation.

Kapitel 7, *Zusammenfassung und Ausblick*, schließen diese Arbeit.

Im Anhang sind die in der Arbeit verwendeten Materialien der Evaluation hinterlegt.

2. Methodik der Arbeit

Der Ansatz eines flexiblen Managements von AAL-Umgebungen mit Hilfe von ontologiebasierter Konfiguration umzusetzen, bedarf einer umsichtigen systematischen Vorgehensweise. Ziel ist es Prozesse, Komponenten und Akteure zu identifizieren und in einem Konfigurationsframework zusammenzubringen.

Das Vorgehen dieser Arbeit ist in Abbildung 2.1 dargestellt und folgt der Methodik des Design Science Research (DSR) [HMPR04]. Diese Forschungsmethode bietet das Rahmenwerk für ein strukturiertes Vorgehen, bei dem Fragestellungen methodisch über Forschungsfragen verbunden werden, die sowohl Anwendungsdomänen als auch andere konzeptionelle Wissensbasen beinhalten. Zunächst sind die Anforderungen an ein flexibles Management von AAL-Umgebungen, bezugnehmend zur Forschungsfrage 1 dieser Arbeit, zu erfassen. Um anschließend die Vor- und Nachteile einer ontologiebasierten Konfiguration, bezugnehmend zu Frage 2 dieser Arbeit, herauszuarbeiten, bedarf es der konzeptionellen Gestaltung einer passenden Lösung die anschließend implementiert und evaluiert werden kann. Für das flexible Management von AAL-Umgebungen bedarf es einer Analyse von Rollen, Organisationen und Technologien, die bereits aus anderen Domänen wie Smart Home vorhanden sind. Aber auch neue und theoretische Erkenntnisse und Konzepte, die sich im Verlauf der Forschung ergeben, können zur Lösung der Fragestellungen integriert werden [HMPR04, S. 80].

Hierzu gliedert Hevner et al. den Forschungsprozess in drei Phasen (vgl. [HMPR04, S. 78]):

Phase 1: Problemanalyse

Zuerst erfolgt die Problemanalyse, in der (vgl. Abbildung 2.1) die Akteure und ihre Anforderungen (siehe Kapitel 2.1) an ein flexibles Management erhoben werden. Weiterhin wird der Konfigurationsbedarf von AAL-Anwendungsfällen sowie AAL-Plattformen analysiert. Die Beziehungen zwischen den einzelnen Analysen sind durch Pfeile dargestellt. So beeinflussen die Akteure mit ihren Anforderungen direkt die Analyse des Konfigurationsbedarf der AAL-Plattformen und AAL-Anwendungsfällen. Alles drei zusammen bilden den Startpunkt in die Phase 2.

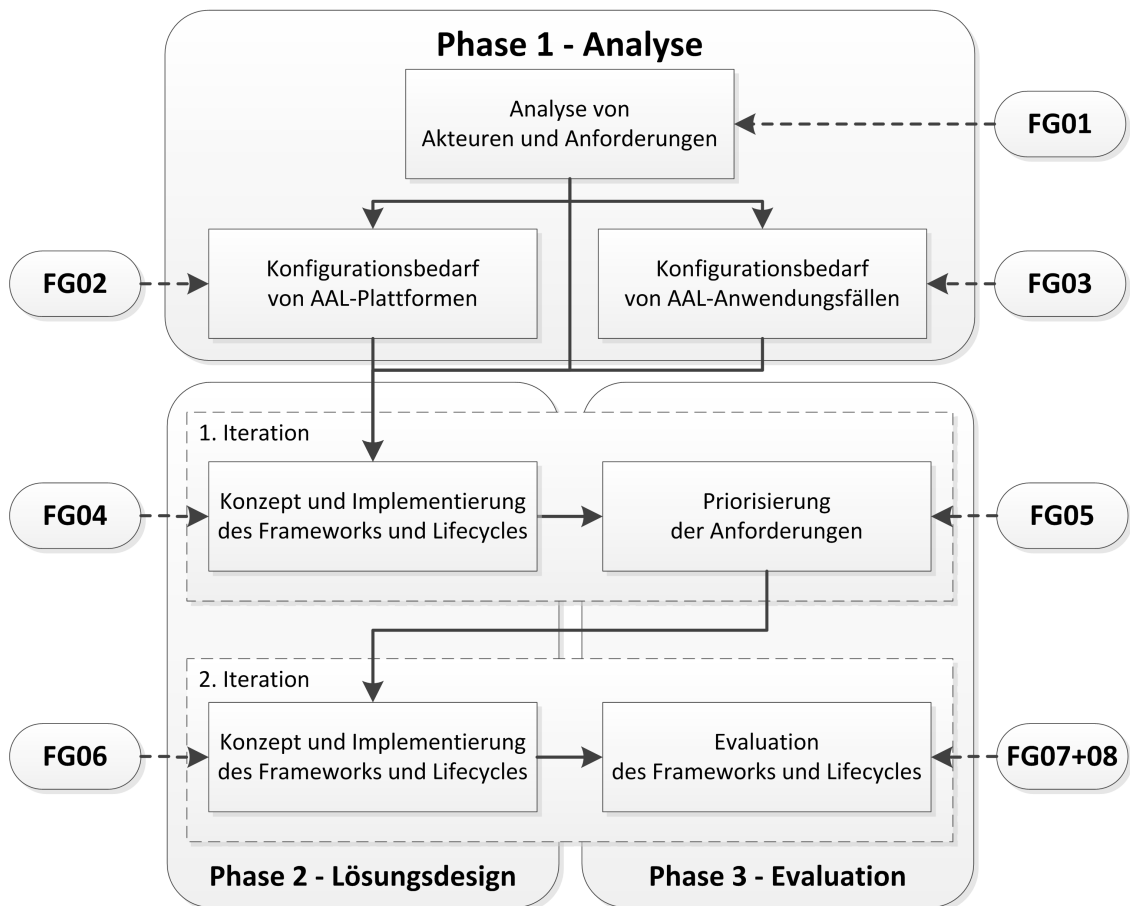


Abbildung 2.1: Übersicht zum methodischen Vorgehen bei der Entwicklung des Konfigurationsframeworks nach der Methodik Design Science Research (DSR) [HMPR04]. Dargestellt sind die drei DSR-Phasen mit ihren Komponenten und ihrer schematischen Abfolge. Jeder Komponente ist hierbei die dazugehörige Fokusgruppe (kurz FG) zugeordnet.

Phase 2: Lösungsdesign

Nach der Analyse wird ein Lösungsdesign erarbeitet. Dieses umfasst Konzepte und Prozesse (vgl. Kapitel 2.2.1), die auch implementiert werden (siehe Kapitel 2.2.2).

Phase 3: Evaluation

Mit Hilfe der in Phase 2 durchgeführten Implementierung kann jetzt eine Evaluation der gesamten Lösung erstellt werden. Die erste Evaluation (siehe Kapitel 2.3.1) dient der Priorisierung der identifizierten Anforderungen. Die zweite Evaluation (siehe Kapitel 2.3.2) bewertet die Konfigurationskonzepte und das Framework mit allen Komponenten.

Das hier gewählte Vorgehen in zwei Iterationen zur Erstellung des Konfigurationsframeworks sichert nach Markus et al. [MaMG02] die Qualität des Ergebnisses im Sinne des DSR.

Entsprechend der DSR wurde in dieser Arbeit die Methode der Fokusgruppe gewählt, um bestimmte Sachverhalte zu erarbeiten, zu validieren oder zu evaluieren. Die einzelnen Fokusgruppen (kurz: FG) sind in den runden Kästchen dargestellt und über gestrichelte Pfeile den dazugehörigen Komponenten der DSR zugeordnet. Die Fokusgruppen-Methode wurde ergänzend mit der Methodik Living Laboratories (kurz: Living Labs) in der Evaluation in Phase 3 eingesetzt.

Bei Fokusgruppen (kurz FG) handelt es sich um eine Form der Gruppendiskussion, die in der qualitativen Forschung Einsatz findet. Sie ist dadurch gekennzeichnet, „dass eine Gruppe von Personen in strukturierter oder moderierter Weise über ein bestimmtes Thema diskutiert“ [BoDo06, S. 319].

Die Methode der Fokusgruppen kann nach Lamnek [Lamn05] aus dem jeweiligen Bedarf und dem damit verbundenen Erkenntnisinteresse sehr unterschiedlich gestaltet werden. In dieser Arbeit wurde abhängig vom Diskussionsbedarf eine künstliche Gruppe aus bewusst gewählten Domänenexperten erstellt. Als Experten werden hierbei nach Meuser und Nagel [MeNa08] Menschen betrachtet, die eine Population repräsentieren und berufsbedingt über ein bestimmtes Wissen verfügen. Die Gruppe konnte hierbei wahlweise homogen oder inhomogen sein und variierte in der Teilnehmerzahl. Es gab immer thematische Vorgaben für die Fokusgruppen, an welche die Diskussionsdauer angepasst wurde. Der Ablauf war teilstrukturiert und wurde moderiert. Die Aufzeichnung erfolgte in Form von Ergebnisprotokollen. Die Ergebnisse der Fokusgruppen können zur Theoriebildung, Hypothesenprüfung [BoDo06, S. 320] und Spezifikationen von Problemlagen [Grun10, S. 195] eingesetzt werden.

Insgesamt werden in dieser Arbeit acht Fokusgruppen eingesetzt (vgl. Abbildung 2.1) mit insgesamt 40 Teilnehmern.

Ergänzend zur Methode der Fokusgruppen wird in dieser Arbeit die Methodik Living Labs eingesetzt. Die Methodik der Living Labs stellen eine Plattform zur interdisziplinäre Forschung, Entwicklung und Evaluierung neuer Technologien, Produkte und Dienstleistungen dar. Der Fokus liegt hierbei auf der Nutzereinbindung während des gesamten Innovations- und Entwicklungsprozess bis hin zur Markteinführung.

In den letzten Jahren hat sich die Methodik der Living Labs für die Forschung an AAL als geeignet erwiesen. Dies beruht auf der Vereinigung und Weiterentwicklung

verschiedener Techniken, unter anderem Open Innovation [Ches06, ChAp07], Lead User [vHip95, vHip05] und Crowdsourcing [Howe06].

Da diese Umgebungen teilweise erst in der Entstehung sind, wurden neue Methoden entwickelt, um nutzerzentriert an intelligenten Umgebungen zu arbeiten. Eines der ersten Living Labs wurde 1998 am Georgia Tech [oTec14] gegründet, um interdisziplinär verschiedene Technologien und menschliche Dienstleistungen zu erforschen.

Ausgehend von diesem ersten Ansatz gibt es inzwischen viele verschiedene Living Labs über die ganze Welt verteilt. Das European Network of Living Labs (ENoLL) gibt einen guten Überblick zu den fast 300 Living Labs¹ in Europa. Andere Zusammenschlüsse findet man beispielsweise unter Living Labs Global² oder Living-Labs@Work³. Neben Informationen zu den verschiedenen Labs im Netzwerk findet man in ihnen weitere Informationen zu den verschiedenen Methoden und Möglichkeiten einer Zusammenarbeit.

Mit der Vielzahl an Living Labs, die in den letzten Jahren entstanden sind, hat sich auch eine allgemeine Definitionsrichtung durchgesetzt. Diese wird vom ENoLL [ENOL14] passend zusammengefasst:

Living labs are real-life test and experimentation environments, where users and producers co-create innovations, in a trusted, open ecosystem that enables business and societal innovation. Living labs enable the co-creation of user-driven and human-centric research, development and innovation of technologies, products and services focused on the well-being of people.

Hierbei sieht ENoLL Living Labs als lebensnahe neue Innovationsstrategie, um die kommerziellen Chancen eines Produktes zu erhöhen. Es können in dieser offenen Umgebung nutzerzentriert und -getrieben Technologien, Produkte und Dienste entwickelt werden. Hierzu haben verschiedene Forscher [PaMa01, ENOK05, BaPD05, SCHK⁺07, BKHS09] den methodischen Hintergrund von Open Innovation, Lead User und Crowdsourcing entsprechend erweitert und an die neuen Konzepte angepasst. Sie folgen den CORES-Prinzipien (Kontinuität (Continuity), Offenheit (Openness), Realismus (Realism), Ermächtigung des Nutzers (Empowerment of users), Spontanität (Spontaneity)) [Core07] für die Living Lab-Forschung.

Zusammen mit den Fokusgruppen sind die Living Labs ein ständiger Innovationsbegleiter für das Konfigurationsframework. Im Zuge dieser Arbeit werden fünf Living Labs verwendet:

1. CIAMI Living Lab – TSB (Spanien)
2. CERTH-HIT und CERTH-IBBR Labs (Griechenland)
3. Laboratory of Environment Intelligence – UPM (Spanien)
4. Smart Home – AIT (Österreich)
5. FZI Living Lab AAL (Deutschland)

¹<http://www.openlivinglabs.eu/> Stand: „Fourth Wave“ (Zugriff am 21.11.2014)

²<http://www.livinglabs-global.com/> (Zugriff am 21.11.2014)

³http://www.ami-communities.eu/wiki/LivingLabs@Work_SIG (Zugriff am 21.11.2014)

Nummer	Beschreibung	# Teilnehmer
FG01	Akteure und ihre Anforderungen	5
FG02	Lebenszyklus eines AAL-Service	6
FG03	Living Labs und AAL-Plattformen	5

Tabelle 2.1: Übersicht zu den in Phase 1 beteiligten Fokusgruppen.

2.1 Phase 1: Analyse

Entsprechend der DSR wird mit der Analysephase begonnen. Um die Anforderungen der Konfiguration für ein flexibles Management richtig zu untersuchen, muss zuerst herausgefunden werden, welche die hierfür relevanten Akteure in einer AAL-Umgebung sind. Anschließend erfolgt die Analyse der Anforderungen der verschiedenen Akteure. Beides findet sich in Kapitel 2.1.1.

Im zweiten Teil dieses Unterkapitels (siehe Kapitel 2.1.2) wird die Frage nach dem Konfigurationsbedarf bei AAL-Plattformen gestellt. Hierbei können besonders die zuvor analysierten Akteure und deren Anforderungen als Grundlage genutzt werden. Das Ergebnis ist eine Bewertung aktueller Plattformen auf ihren Konfigurationsbedarf hin sowie ihr Potenzial im Bezug auf flexibles Management.

Abschließend wird in Kapitel 2.1.3 die Methode zur Analyse des Konfigurationsbedarfs von AAL-Services vorgestellt. Das Ergebnis dieses Kapitels ist eine Beschreibung der notwendigen Konfigurationseingriffe und ihrer Komplexität, um einen AAL-Service entsprechend der Akteuranforderungen in die AAL-Umgebung zu integrieren.

Die an Phase 1 beteiligten Fokusgruppen sind in Tabelle 2.1 dargestellt und werden in den jeweiligen Unterkapiteln genauer vorgestellt.

2.1.1 Analyse der Akteure und ihrer Anforderungen

Die Akteure sowie ihre Anforderungen im Bereich der AAL-Umgebungen unterscheiden sich durch heterogene Kontexte und schnell wechselnde Bedürfnisse von bekannten Anforderungsprofilen aus der Hausautomatisierung und dem Smart Home [KaNF08]. Die Vorgehensweise zur Erfassung der Akteure folgt der Fokusgruppen-Methode.

Für die strukturierte Diskussion in der Fokusgruppe dienten sieben Projekte aus dem AAL-Bereich. Sie wurden vorab entsprechend aufbereitet. Im Folgenden werden die einzelnen Quellen vorgestellt.

Die ersten relevanten Akteure werden vom Eberhardt et al. [Eber09] beschrieben. In ihrem Dokument ist die Rede von unmittelbar profitierenden Zielgruppen und mittelbar profitierende Zielgruppen. Alle nicht in diese Gruppen passenden Rollen werden als Sonstige und sonstige profitierende Zielgruppe bezeichnet. Jede dieser vier Gruppen wird auf ein bis zwei Ebenen weiter verfeinert und bildet Personen sowie Institutionen ab. Anforderungen werden nur implizit in der Akteurbeschreibung dargestellt. Eine explizite Analyse und Darstellung der Anforderungen der Zielgruppen liegt nicht vor.

Das Projekt Amigo [Amig14] hat das Ziel die Lebensqualität der Menschen in dieser Umgebung zu steigern. Dabei liegt der Fokus auf Netzwerke im häuslichen Umfeld. Der Fokus dieses Projektes liegt hauptsächlich auf der Entwicklung prototypischer medizinischer Dienste. Die Akteure von Amigo (im Projekt Stakeholder genannt) sind in sieben Rollen gegliedert. Für diese Arbeit können die Beschreibungen zu den technischen Akteuren und ihren Anforderungen [Geor05, S.205ff] verwendet werden.

Im Projekt MPOWER [MPOW14] wurde eine offene Plattform entwickelt, die zur Unterstützung von geistig Behinderten und älteren Menschen dient. Es wurden in diesem Projekt verschiedene Akteure identifiziert, die dem MPOWER Service Lifecycle zugeordnet werden. Dieser teilt sich grob in eine „pre-release“- und „release“-Phase und hat seinen Schwerpunkt auf der Entwicklerperspektive und den damit verbundenen Anforderungen [Wald07, S.20ff][Benc06, S.18f]. Die in dem Projekt entwickelten Anforderungen werden in dieser Arbeit für die Fokusgruppen aufbereitet.

Mit dem Projekt OASIS [OASI14] wurde eine offene, standardisierte Architektur für die Integration von AAL-Services für die Unterstützung des Alltags von älteren Menschen entwickelt. Die hierbei entwickelte Akteurdefinition setzt im Kern auf die älteren Menschen mit ihren verschiedenen Bedürfnissen und Wünschen, die sie an das System stellen [IsGVG08b, S.8ff]. Hieraus konnten Aspekte für die Endanwender entnommen werden.

PERSONA hatte zum Ziel eine offene und erweiterbare AAL-Plattform für Dienstleistungen zu entwickeln. Es sollen AAL-Services unterstützt werden können, die sich auf alltägliche Aktivitäten und Aufgaben beziehen. Weiterhin wurden AAL-Services zum Thema Sicherheit und Mobilität behandelt. Die Definition der Akteure erfolgt in PERSONA entsprechend ihren Geschäftsbeziehungen untereinander. Hierbei wurden sieben Rollenbilder definiert, die sich mit ihren Unterrollen in über 100 Akteure aufgliedern. Auch die Beziehungen sowie Anforderungen zwischen ihnen wurden erarbeitet [MuRW08].

Das Projekt SOPRANO entwickelt eine serviceorientierte und semantische Plattform zur Unterstützung älterer Menschen in ihrem täglichen Leben. Der Fokus wurde entsprechend auf die „Assisted Person“ – die zu unterstützende ältere Person – gelegt. Diese Person wurde in vier Dimensionen genau untersucht und eingeteilt [Empi07, S10ff].

VAALID [Vaal14] beschäftigte sich mit Musikerziehung im Feld von AAL. Es sollte eine integrierte Entwicklungsumgebung (VAALID IDE) geschaffen werden, welche die Nutzbarkeit durch ältere Menschen anstrebt. Hierzu teilt VAALID die Akteure in drei Gruppen (Beneficiaries, Solution Designers, Secondary Sources), in denen sich verschiedene Akteure wiederfinden [Scha08, S.48ff].

Die Ergebnisse aus diesen Quellen wurden so aufbereitet, dass sie initial in der Fokusgruppe 01 weiter analysiert werden konnten. Die Fokusgruppe wurde mit fünf Experten der AAL-Domäne besetzt, um in einer strukturierten Diskussion über sechs Stunden die folgenden Aufgaben zu bearbeiten:

- Identifizierung der Akteure einer AAL-Umgebung
- Bestimmung der Anforderungen der Akteure

- Identifizierung der Pflichtanforderungen

Anschließend erfolgte eine Konsolidierung der Ergebnisse. Das Resultat wurden den Teilnehmern der Fokusgruppe in einer zweistündigen Sitzung zur Validierung der Akteure, Anforderungen und Pflichtanforderungen erneut vorgelegt. Die Akteurgruppen sind an der von der AALIANCE Standardisierung [vdBCW10] vorgeschlagenen Einteilung orientiert.

2.1.2 Analyse des Konfigurationsbedarfs bei AAL-Plattformen

Ausgehend von den bisher identifizierten Akteuren und Anforderungen werden existierende AAL-Plattformen auf ihre Konfigurationsmöglichkeiten untersucht und ihren Unterstützungsgrad der identifizierten Akteure und den Erfüllungsgrad der Anforderungen untersucht. Die Analyse erfolgt in drei Schritten:

1. Erfassen existierender AAL-Plattformen
2. Konsolidierung der AAL-Plattformen gegen die Pflichtanforderungen einer flexiblen AAL-Umgebung im Rahmen der Fokusgruppe 02. Bei Rückfragen zu den Akteuren oder Anforderungen werden sie von Fokusgruppe 01 unterstützt. Die Fokusgruppe 02 bestand aus fünf AAL-Experten, die sich in einem initialen persönlichen Treffen und 4 Telefonaten von mindestens einer Stunde austauschen. Das Ergebnis dieses Schrittes ist eine Liste von konsolidierten Plattformen.
3. Testen der in Schritt 2 konsolidierten Plattformen gegen die identifizierten Anforderungen einer flexiblen AAL-Umgebung

Im ersten Schritt werden existierende AAL-Plattformen in einer umfassenden Marktanalyse erfasst. Hierbei teilt sich der Markt in drei Gruppen: Industrie, Forschung und Open Source Gemeinschaft. Die AAL-Plattformen werden durch folgendes Vorgehen extrahiert:

- Es werden 22 ausgewählte Domänenexperten der Hausautomatisierung, des Ambient Intelligence und AAL per E-Mail-Umfrage gebeten, alle ihnen bekannten AAL-Plattformen aus Industrie und Open Source Community anzugeben, die aus ihrer Sicht für eine AAL-Plattform in Frage kommen.
- Parallel wird eine Suche mit Hilfe von Anfragen in etablierten Internetforen in den Bereichen der Hausautomatisierung und Ambient Intelligence durchgeführt. Hierzu werden digitale Anfragen in folgenden Foren platziert: (haustechnikdialog.de, fs20-forum.de, knx-user-forum.de, elektrikforum.de).
- Ergänzend wird eine Internetrecherche mit Hilfe der Internetsuchmaschine Google⁴ durchgeführt. Es werden mit den folgenden Begriffen Anfragen gestellt und ausgewertet: Middleware, Hausautomation, Haussteuerung, Ambient Intelligence, Ambient Assisted Living, House Automation, House Remote Control.

⁴siehe <http://www.google.de> (Zugriff am 1.12.2012)

- AAL-Plattformen aus der Forschung ließen sich zusätzlich durch Sichtung der geförderten EU-Projekte erfassen. Innerhalb des 5ten bis 7ten Rahmenprogramms der EU wurden über 1000 Forschungsvorhaben unterstützt⁵. Aus dieser Quelle werden die Forschungsplattformen erfasst.

Im zweiten Schritt wird die konsolidierte Liste der AAL-Plattformen gegen die Pflichtenforderungen geprüft. Alle AAL-Plattformen, welche die Pflichtenforderungen erfüllen, werden im dritten Schritt nach folgendem Vorgehen weiter untersucht:

1. Sichtung und Analyse weiterer Informationsquellen zur AAL-Plattform (z.B. Foren, Dokumentationen, Quellcode)
2. Installieren der AAL-Plattform in der Testumgebung
3. Konfigurieren der AAL-Plattform mit Sensoren und Aktoren
4. Integration und Konfigurieren von Anwendungsfällen (mindestens ein Erinnerungsanwendungsfall)
5. Nutzen der AAL-Plattform mit ihren Sensoren und Aktoren sowie Anwendungsfällen für mindestens 2 Stunden
6. Rückbau der AAL-Plattform mit all ihren Komponenten

Abschließend findet eine Bewertung der Plattform gegenüber allen Anforderungen an AAL-Plattformen (vgl. Kapitel 3.2) statt. Gemäß des Testverhaltens der AAL-Plattform wird die Bewertung durch zwei AAL-Experten vorgenommen. Die Tests wurden in zwei verschiedenen Living Labs durchgeführt. Neben der AAL-Tauglichkeit durch die Anforderungen werden auch die Potenziale und Defizite in Bezug auf Konfigurationsaspekte während der tieferen Analyse beleuchtet. Die Erfüllung der Anforderungen werden hierbei wahlweise auf zwei Arten dargestellt:

1. Ordinaler Skalar: „4“ sehr schlecht erfüllt; „3“ schlecht erfüllt; „2“ gut erfüllt; „1“ sehr gut erfüllt; „0“ für nicht auswertbar
2. Binärer Skalar: „ja“ Anforderung erfüllt; „nein“ Anforderung nicht erfüllt; „0“ für nicht auswertbar

2.1.3 Analyse des Konfigurationsbedarfs bei AAL-Anwendungsfällen

In diesem Kapitel wird der Konfigurationsbedarf der AAL-Anwendungsfälle analysiert. Hierzu wird die Fokusgruppe 03 zur Extraktion des Konfigurationsbedarfs eingesetzt. Für diese Gruppe werden sechs AAL-Experten einbezogen, die über einhalb Stunden teilstrukturiert und moderiert zwei Hypothesen prüfen.

Hypothese 1: Die 10 AAL-Anwendungsfälle (AALS I - X) sind Repräsentanten für AAL-Services in AAL-Umgebungen.

⁵siehe <http://cordis.europa.eu/fp5/> , <http://cordis.europa.eu/fp6/> und <http://cordis.europa.eu/fp7/> (Zugriff am 1.12.2012)

Projekte	# AAL-Anwendungsfälle
PERSONA	14
SOPRANO	10
MPOWER	13
OASIS	6
AMIGO	6
Andere Projekte	6

Tabelle 2.2: AAL-Projekte mit den dazugehörigen Anwendungsfällen. CAPSIL, EasyLine, Eldergames, Hermes, Vital, ePal, Netcareity, SENIOR sind unter dem Stichwort „Andere Projekte“ zusammengefasst.

Hypothese 2: Der Konfigurationsbedarf von AAL-Services besteht aus „einfachen“ sowie „komplexen“ Konfigurationsparametern und Wissensobjekten.

Die Ergebnisse der einzelnen Hypothesenprüfungen werden in Ergebnisprotokollen festgehalten. Entsprechend der einzelnen Hypothesen erfolgte im ersten Schritt eine Literaturanalyse. Im Folgenden wird beschrieben, welche Literatur gesichtet wurde und wie welches Material als Diskussionsgrundlage erstellt und den Teilnehmern vorgelegt wurde.

Zu Hypothese 1

Zum Erstellen dieser Hypothese werden geeignete AAL-Services herausgearbeitet und diese auf ihre Repräsentativität bezüglich der Nutzung in AAL-Umgebungen geprüft. Aus verschiedenen AAL-Projekten existiert eine große Anzahl von AAL-Anwendungsfällen, die in Abstimmung mit ihren verschiedenen Nutzergruppen entwickelt und evaluiert wurden. Je nach Projekt befinden sie sich in verschiedenen Stadien der Entwicklung. Es lag aber mindestens eine finale Servicebeschreibung vor. Zusammen wurden für diese Hypothese 55 Anwendungsfälle (alle Anwendungsfälle siehe [Guil13]) aus 11 Projekten ausgewählt, um als Basis für die Referenzanwendungsfälle zu dienen (vgl. Tabelle 2.2).

Der gesamte Auswahlprozess der Repräsentanten wird aus Konfigurationssicht begleitet. An dieser Stelle soll nur auf die Auswahlkriterien, die im Zusammenhang mit der Flexibilität stehen, eingegangen werden⁶. Jedes der vier Kriterien ist zusätzlich mit einem Gewichtungsfaktor versehen, um sie entsprechend ihrer Wichtigkeit im AAL-Kontext in das Ergebnis aufzunehmen. Eine genauere Beschreibung der nicht konfigurationsrelevanten Kriterien findet man in [Guil13, S. 95f]. Die folgende Aufzählung beschreibt die für die Konfigurationsbedarfsanalyse relevanten Kriterien genauer:

Personalisierbarkeit misst, wie schwer es ist, den AAL-Service zu adaptieren und zu personalisieren, um ihn in die AAL-Umgebung zu integrieren. Je konfigurierbarer ein AAL-Service, desto besser fällt die Bewertung aus. Dieser Parameter geht in die Bewertung mit einfacher Gewichtung ein.

⁶Zur Bewertung der AAL-Services standen 11 Kriterien zur Verfügung, von denen vier direkt relevant für die Flexibilität sind [Guil13, S. 98f].

Abhängigkeit zur Originalplattform misst, wie gut sich der AAL-Service aus seiner Originalumgebung herauslösen lässt. Je besser ein Service hierbei bewertet wird, umso flexibler lässt er sich in einer AAL-Umgebung einsetzen. Dieser Parameter wurde in der Bewertung mit einfacher Gewichtung belegt.

Skalierbarkeit des Services gibt an, wie einfach sich der Service durch Erweitern oder Löschen von Features anpassen lässt. Je flexibler ein Service hierbei abschneidet, umso besser fällt seine Bewertung aus. Dieser Parameter wurde einfach gewichtet.

Bezahlbarkeit und Integrierbarkeit bewertet, wie einfach und mit welchen Kosten ein Service sich in der AAL-Umgebung integrieren lässt. Ein hoher Wert wird vergeben, wenn ein Service sich leicht und kostengünstig integrieren lässt. Dieser Parameter wurde in der Bewertung doppelt gewichtet.

Als Nächstes ist sicherzustellen, dass die ausgewählten Anwendungsfälle wirklich alle alltäglichen Unterstützungsszenarien von hilfsbedürftigen Menschen abbilden und somit als Referenzanwendungsfälle gelten können. In der Literatur gibt es hierzu verschiedene Kategorisierungen. Zur Vorbereitung der Gruppendiskussion wurden die folgenden Kategorisierungssysteme analysiert, das Passendste ausgewählt und mit den Referenzanwendungsfällen abgeglichen.

Die erste Aufteilung von Anwendungsfällen in Kategorien findet sich in der Definition von Weiss et al. [Weis09]. Es wird unterschieden zwischen: Gesundheit und HomeCare, Haushalt und Versorgung, Sicherheit und Privatsphäre sowie soziales Umfeld. Diese Aufteilung ist sehr ungenau und wird nicht näher ausformuliert. Eine genauere Definition liefert das Positionspapier des VDE [Eber08, S.7ff] mit den Kategorien: Informationsversorgung, Kommunikation, Entertainment, Sicherheit und Datenschutz, Ernährung, Gesundheit und Dienste, um den Nutzer an Termine und Tätigkeiten erinnern zu können. Im Jahr 2011 wurde eine darauf aufbauende weiter verfeinerte und mit anderen Quellen vereinte Kategorisierung veröffentlicht [Eber11]. In dieser Arbeit finden sich 37 Beispielszenarien, welche 6 Kategorien (genannt Anwendungsfelder) nach Andrushevich et al. [AKBK09] zugeordnet werden: Informationsassistentz; Intelligentes Umgebungsverhalten; Vorhersehen von Notfallsituationen; Erkennen von Notfallsituationen; Sicherheit und Privatsphäre. Diese Einteilung deckt sich mit der anderer europäischer Projekte [Denk14]. Neben den für AAL entwickelten Kategorien existieren noch Kategoriensysteme aus den Pflegewissenschaften (z.B. AEDL⁷, AEBDL⁸, ICF⁹) und Medizin, die sich bestenfalls partiell auf die AAL-Domäne übertragen lassen und in die Entwicklung von AAL-Anwendungsfallkategorien nur peripher eingeflossen sind. Somit wurden die Kategorien von Andrushevich für die Fokusgruppe übernommen und dienen als Grundlage für die Anwendungsfallrepräsentanten.

Zu Hypothese 2

Die zweite Hypothese beruht auf den Untersuchungen des Konfigurationsbedarfs für die AAL-Referenzservices. Diese Services wurden im Projekt universAAL zusammen mit den verschiedenen an dem jeweiligen Service beteiligten Akteurguppen

⁷Aktivitäten und existenzielle Erfahrungen des Lebens

⁸Aktivitäten, Beziehungen und existenziellen Erfahrungen des Lebens

⁹International Classification of Functioning, Disabilities and Health <http://www.dimdi.de/static/de/klassi/icf/> (Zugriff am 21.11.2014)

Nummer	Beschreibung	# Teilnehmer
FG04	Akteure und ihre Anforderungen	5
FG06	Lebenszyklus eines AAL-Service	7

Tabelle 2.3: Übersicht zu den in Phase 2 beteiligten Fokusgruppen.

über drei Jahre hinweg konzipiert, implementiert und evaluiert. In enger Absprache mit den Entwicklern wurden die einzelnen AAL-Services entsprechend des Konfigurationslifecycles untersucht und im Sinne einer flexiblen AAL-Umgebungsgestaltung weiterentwickelt. Hierbei wurden potenzielle Konfigurationsparameter in Abstimmung mit den Entwicklern identifiziert.

Jeder Anwendungsfall wird nach folgendem Schema untersucht:

1. Lesen der Dokumentation des Anwendungsfalls
2. Ggf. weiteres Studium des Quellcodes und Rücksprache mit den Entwicklern
3. Formulieren von Konfigurationspotenzial
4. Identifizieren von Konfigurationsparametern in der Use Case Beschreibung und dem Quellcode
5. Verifizieren des gefundenen Konfigurationspotenzials

Nach anschließender Konsolidierung der Ergebnisse wurde die finale Liste den Teilnehmern der Fokusgruppe erneut vorgelegt und durch diese validiert.

2.2 Phase 2: Design

Die Designphase der DSR entwickelt einen Lösungsvorschlag für die in der Analysephase identifizierten Anforderungen. Hierzu beinhaltet die DSR eine konzeptionelle Komponente, die in Kapitel 2.2.1 methodisch aufgearbeitet und dargestellt ist, und in Kapitel 2.2.2 finden sich die dazugehörige Implementierung zur Prüfung der entwickelten Konzepte.

Die an Phase 2 beteiligten Fokusgruppen sind in Tabelle 2.3 dargestellt. Die Ergebnisse der ersten Konzeptionsphase werden von der Fokusgruppe 04 (mit fünf Teilnehmer) validiert. Auf Basis der ersten Frameworkiteration wird die Anforderungsgewichtung vorgenommen und erstes Feedback zum Framework und Lifecycle einzuholen. In der zweiten Iteration wird das finale Framework erstellt und nach der Validierung durch Fokusgruppe 06 (sieben Teilnehmer) evaluiert. Die Validierung der Konzeptionsergebnisse erfolgte jeweils parallel zu Konzeption und Implementierung. Hierbei werden den Teilnehmern die erstellten Unterlagen und Prototypen bereitgestellt zum Testen im Living Lab und einmal pro Woche in Telefonkonferenzen das Feedback eingeholt. Hierbei werden vom Moderator Diskussionsschwerpunkte gesetzt. Weiterhin trifft sich jede Fokusgruppe am Ende der Iteration einmal persönlich.

2.2.1 Konzeption des Konfigurationsframeworks

Als Ordnungsrahmen für die Konfiguration wird ein Konfigurationsframework konzipiert, das für die spätere Evaluation implementiert wird. Die Konzeption erfolgte in zwei Iterationen und beruht auf den Anforderungen der Analyseergebnisse.

Die Entwicklung des Frameworks ist an den Richtlinien der AAL-Referenzmodellierung [TFFVH⁺12] ausgerichtet. Somit wird die Kompatibilität des entwickelten Konfigurationsframework zum AAL-Referenzmodell von universAAL sichergestellt, und kann als Teil der universAAL-Plattform eingesetzt werden. Hierzu werden alle Konzepte in concept maps dargestellt und entsprechend beschrieben. Concept maps ist eine grafische Darstellung, um Sachverhalte anschaulich zu modellieren. Sie finden beispielsweise Verwendung in der Beschreibung von serviceorientierten Architekturen [MLMB⁺06] oder beim Überblick für OWL-S Servicebeschreibungen [MBOM⁺06]. Concept maps sind einfache Graphen, deren Knoten die Konzepte repräsentieren und deren gerichtete Verbindungen mit ihren Beschriftungen Relationen zwischen diesen Konzepten ausdrücken.

Im Rahmen der Erstellung des Frameworks werden ein Lifecycle der Konfiguration entwickelt. Dieser versteht sich in der Definition von Gu und Lago [GuLa07] als neuer Ansatz, die traditionellen Programmierparadigmen aufzubrechen und einen „Stakeholder-driven Service Life Cycle“ zu entwickeln. Im Sinne der Kompatibilität zu serviceorientierten Plattformen reiht sich diese Phase in das „Application design“ und die „Appication implementation“ nach [GuLa07] ein und erweitert sie um Konfigurationsaspekte.

Als Grundlage der semantischen Konfigurationskonzepte im Bereich von AAL wurde neben der durchgeführten Analyse der Akteure und Anforderungen der aktuelle Stand der Forschung im Bereich der wissensbasierten Konfiguration betrachtet. Sie ist ein Teil der künstlichen Intelligenz und das häufig verwendete Beispiel für Expertensysteme [Stum97, SaWe98].

Norbisrath et al. [NoMo06] entwickelten einen Konfigurator für sechs Smart Home Anwendungsfälle. Dieser wurde mit zwei Demonstratoren in einer echten Umgebung erfolgreich getestet. Der vorgestellte Prozess zerfällt in drei Phasen: die Spezifikation, die Konfiguration und das Deployment. Die Konzepte und Werkzeuge sind ausschließlich auf diese sechs Anwendungsfälle zugeschnitten und lassen keine Generalisierung zu.

In der Arbeit von Kowar et al. [KaNF08] wird eine Smart Home Plattform präsentiert, in der Endanwender die Anwendungsfälle selbst installieren, starten, stoppen und deinstallieren können. Hierzu haben Anwendungsfälle ein Profil, welches relevanten Informationen für diese Aktionen enthält. Hierbei funktionieren alle Anwendungsfälle vollständig unabhängig voneinander, da sie sich keine Ressourcen teilen. Die Evaluation erfolgte mit vier Aufgaben und 25 Teilnehmern an einem Modelldemonstrator.

Im Jahr 2009 wurde von Cetina et al. [CTPRC09] der Artikel „Mass Customisation along Lifecycle of Autonomic Homes“ veröffentlicht. Er beschreibt mit dem Wort Lifecycle die verschiedenen Situationen, in denen sich die intelligente Umgebung befinden kann. Seine „Customisation“ ist die geeignete automatische Anpassung an die Lifecycle-Situation. Das System konfiguriert sich selbstständig entsprechend des

neuen Kontextes. Konfiguration heißt hierbei, dass nach vordefinierten Regeln bestimmte Anwendungsfälle aktiviert oder deaktiviert werden. Es handelt sich hierbei um einfache Anwendungsfälle, wie dass das Licht in einer intelligenten Umgebung bei Anwesenheit von Personen ein- und ausschaltet werden kann. Alle notwendigen Informationen zur Anpassung der Anwendungsfälle liegen im System vor und werden in der Arbeit nicht weiter besprochen. Die Arbeit konzentriert sich lediglich auf die automatische Aktivierung einer Situation, abgestimmt auf die Benutzerbedürfnisse.

Larson et al. [LaSm10] fassen den Konfigurationsbedarf passend zusammen mit ihrer Aussage, dass es nur wenige Arbeiten auf dem Feld der Konfiguration von Wohnumgebungen existieren und keine Arbeiten, die komplexe Kontexte eingeben. Bisherige Arbeiten beschränken sich ihrer Meinung nach darauf, einfache vordefinierte Komponenten zu kombinieren.

Insgesamt zeigen sich die Arbeiten im Bereich der Anwendungsdomäne Smart Home als nicht flexibel genug, da sie nicht die breite Anzahl von Hardware und Akteuren unterstützen müssen, wie es in AAL der Fall ist. Auch die Komplexität der Anwendungsfälle im Smart Home ist im Vergleich zu AAL eingeschränkt. Die Anwendungsfälle müssen hier nur einfache Komfortfunktionen erfüllen mit einer begrenzten Anzahl an Bewohnern der Umgebung. Auch die Bedürfnisse dieser Personen unterliegen keinen schnellen Änderungen.

Im Bereich der semantischen Konfiguration werden Ontologien verwendet, um konfigurationsrelevante Daten abzulegen und zu prozessieren. Verschiedene Ansätze [YMWZ09, VRLT10, DoYS11] haben Ansätze entwickelt, die mit OWL¹⁰ auf Basis von SWRL¹¹ und Jess¹² gültige Konfigurationen für verschiedene Probleme finden können. Andere Arbeiten [BoCo08, ABBC⁺11] binden zusätzlich verschiedene Datenquellen ein, die über eine Mapping-Ontologie in eine Konfiguration zusammengeführt werden.

Neben der Literatur zur Konfiguration von intelligenten Umgebungen und zur semantischen Konfiguration existieren Vorarbeiten bzw. Normen von Standardisierungsorganisationen zu AAL, die als Richtlinien zum Modellierungsprozess in der Konzeption dienen.

In der Norm des VDE [VDE 12b] werden Kriterien ermittelt, nach denen eine AAL-Umgebung eingerichtet werden sollte. Begonnen wird hierbei mit dem Erheben der Anforderungen an das System. Weiterhin werden die verschiedenen Auswahl- und Installationsmöglichkeiten vorgestellt. Anschließend folgt die Analyse der Sachkenntnisse über die AAL-Komponenten und eventuell die entsprechend notwendige Beratung. Kern ist ein Kriterienkatalog mit verschiedenen Faktoren (z.B. Informationsdarstellung oder Verpackung), um den Prozess der Auswahl und Installation von AAL-Anwendungsfällen besser zu unterstützen.

Die zweite vom VDE verabschiedete Norm [VDE 12c] bezieht sich auf die Qualitätskriterien für Komponenten einer AAL-Plattform. Hier werden zuerst die verschiedenen Nutzergruppen von AAL-Umgebungen und ihre einzelnen Qualitätskriterien vorgestellt. Der zweite Teil der Norm beschäftigt sich dann mit allgemeinen Anforderungen an eine AAL-Umgebung und direkten Anforderungen an AAL-Produkte

¹⁰Web Ontology Language <http://www.w3.org/TR/owl2-overview/> (Zugriff am 26.12.2014)

¹¹Semantic Web Rule Language <http://www.w3.org/Submission/SWRL/> (Zugriff am 26.12.2014)

¹²Rule Engine for Java <http://www.jessrules.com/> (Zugriff am 26.12.2014)

(z.B. technische Sicherheit oder Gebrauchstauglichkeit und Ergonomie). Hieraus lassen sich Eckpunkte in das Referenzmodell übernehmen.

Nach diesen beiden Normen erschienen im gleichen Monat eine Normierungs-Roadmap der Deutschen Kommission Elektrotechnik Elektronik Informationstechnik (DKE) mit weiteren notwendigen Schritten [VDE 12a]. Es wird beschrieben, wo nach Meinung des DKE die Entwicklung von AAL-Umgebungen zukünftig hinführen wird. Hierbei nehmen „anwendungsfallzentrierte Integrationsprofile“ eine wichtige Rolle ein. Hinter diesen Integrationsprofilen steht die Notwendigkeit, eine einheitliche Personalisierung für AAL-Umgebungen zu ermöglichen.

2.2.2 Implementierung des Konfigurationsframeworks

Die Implementierung des Konfigurationsframeworks erfolgt in der Sprache Java mit Hilfe des Frameworks OSGi¹³ und nutzt die Middleware aus dem Projekt universAAL als AAL-Plattform. Die technische Umsetzung folgt der ISO/IEC/IEEE 42010:2011¹⁴, damit das Framework zu der Referenzimplementierung von universAAL kompatibel ist.

Wie die Konzeption wurde auch die Implementierung in zwei Iterationen umgesetzt. Das Ergebnis der ersten Iteration der Implementierung des Konfigurationsframeworks trägt den Arbeitstitel „Jambi“. Dieser Frameworkprototyp war der zentrale Bestandteil der Anforderungsevaluation, um den Teilnehmern die Konfigurationskonzepte mit der Living Lab Methode vorstellen zu können.

Die zweite finale Version des Prototyps zum Konfigurationsframework trägt den Arbeitstitel „Vadiin“ und dient als Basis für die Frameworkevaluation. In ihm sind alle erarbeitete Konzepte des Konfigurationsframeworks implementiert.

2.3 Phase 3: Evaluation

Die dritte Phase der DSR prüft im Rahmen der Evaluation, ob die Konzeption und Implementierung die Anforderungen der Analyse erfüllt. Außerdem adressiert sie Frage 2 nach den Vor- und Nachteilen. Es schließt sich entsprechend der zwei Iterationen in der Designphase je eine Evaluation an. Kapitel 2.3.1 beschreibt, wie die Anforderungen und erste Konfigurationkonzepte auf Basis eines ersten Prototyps evaluiert und gewichtet werden. Kapitel 2.3.2 beschreibt, wie mit Hilfe eines zweiten Prototyps die finale Evaluation der Konzepte des Konfigurationsframeworks vorgenommen wird.

Die Evaluation ist nach Hevner et al. [HMPR04, S. 85] die entscheidende Komponente in einer Forschungsarbeit. Nur durch die Evaluation ist seiner Meinung nach die Überprüfung der auf den Anforderungen entwickelten Konzepte möglich. Der Wahl der Evaluationsmethoden kommt hierbei eine zentrale Rolle zu. Die Methode hängt von dem zu evaluierenden Objekt und den Evaluationszielen ab.

Es können nach Hevner et al. [HMPR04, S. 86] fünf grundlegende Evaluationsmethoden unterschieden werden.

¹³Open Service Gateway initiative

¹⁴Systems and software engineering — Architecture description, the latest edition of the original IEEE Std 1471:2000, Recommended Practice for Architectural Description of Software-intensive Systems

Nummer	Beschreibung	# Teilnehmer
FG05	Living Lab Anforderungsevaluation	4
FG07	Living Lab Frameworkevaluation	5
FG08	Auswertung Frameworkevaluation	3

Tabelle 2.4: Übersicht zu den in Phase 3 beteiligten Fokusgruppen.

Empirisch: Hierzu werden in Fall- oder Feldstudien die erstellten Konzepte in großer Tiefe oder Breite untersucht.

Analytisch: Diese Evaluation bietet die Möglichkeit, die Konzepte analytisch zu zerlegen und Stück für Stück gegen quantitative messbare Faktoren zu evaluieren.

Experimentell: Durch Experimente oder Simulationen werden die Konzepte auf bestimmte qualitative Faktoren wie Benutzbarkeit hin evaluiert.

Testend: Hierbei werden funktionale und strukturelle Tests unterschieden. Der Erfolg der Tests spiegelt das Evaluationsergebnis wider.

Beschreibend: Diese Art der Evaluation stützt sich auf andere Arbeiten und Argumente oder arbeitet mit Szenarien.

Das angewendete Vorgehen für diese Evaluationen entspricht einem Methodenmix. Living Labs geben bei beiden Evaluationen den Teilnehmern die Möglichkeit, wie in einer Fallstudie tief in den Sachverhalt einzutauchen. Auf Basis der Konzepte und des Prototypen kann ein tiefes Verständniss des Konfigurationsframework entstehen. Gleichzeitig wird in einem kontrollierten Experiment die Qualität in einer echten Umgebung studiert. In genau definierten Aufgaben müssen die Teilnehmer Probleme lösen und erfahren so den direkten Umgang mit dem Framework und den Werkzeugen

Die an Phase 3 beteiligten Fokusgruppen sind in Tabelle 2.4 dargestellt und werden in den jeweiligen Unterkapiteln genauer vorgestellt.

2.3.1 Qualitätskriterien

Diese Evaluation hat das Ziel, die Anforderungen anhand von Qualitätsmerkmalen zu bewerten und zu gewichten. Nur vereinzelt wird auf Punkte wie Flexibilität und Konfigurierbarkeit bei der Qualitätssicherung von AAL-Anwendungsfällen hingewiesen [WMSSr⁺12]. Eine etablierte Basis für diese Art von Evaluationen sind Qualitätsmodelle. Sie bilden eine Merkmals-hierarchie mit Qualitätskriterien auf verschiedenen Ebenen ab. Viele dieser Qualitätsmodelle behandeln im Kern die gleichen Kriterien. Beispiele hierfür sind das etablierte FURPS Modell [GrCa87] und das viel genutzte Product Quality Model (PQM) [ISO/10].

Für die Evaluation wurde das PQM gewählt, da es mit seinen acht Kriterien auf zwei Ebenen nicht nur funktionale sondern auch nicht-funktionale Qualitätsanforderungen abdeckt. Weiterhin lässt es sich an eine Anwendungsdomäne adaptieren und ist unabhängig von der Wahl der Methoden zur Evaluation.

Dieses Kapitel ist in drei Teile unterteilt: Zuerst wird der Pretest der Evaluation beschrieben, anschließend die Durchführung der Qualitätsanforderungsevaluation und abschließend die dazugehörige Methodik.

Pretest

Ziel des Pretests ist es, die Konzeptions- sowie Werkzeugebene des Konfigurationslifecycle zu testen. Mit Hilfe des Pretests können eventuelle Fehler vorab beseitigt werden und eine verständliche Demonstration sichergestellt werden. Im Evaluationstest wurde der Prototyp Jambi in fünf¹⁵ verschiedenen europäischen Living Labs getestet. In jedem Living Lab wird ein zuständiger Deployer benannt. Dieser erhält detaillierte Anweisungen (vgl. Anhang A.1), wie ein AAL-Service zu integrieren ist. Als Beispiel für ein AAL-Service wird der AAL-Service "Nutrition Advisor" (Ernährungsberater) verwendet. Die Deployer der Living Labs haben dem geplanten Ablauf in der Demonstration zu folgen:

1. Planen des Unterstützungsbedarfs für den Endanwender und herunterladen des AAL-Services aus dem Online Store
2. Installation des AAL-Services in der AAL-Umgebung
3. Konfiguration des AAL-Services entsprechend der Endanwenderwünsche und Umgebungsanforderungen
4. Verifizieren der korrekten Integration des AAL-Services

Abschließend beantwortet jeder Deployer einen Fragebogen (vgl. Anhang A.2), in dem Erfolge und Schwierigkeiten während der Prozesse und Nutzung der Werkzeuge des Konfigurationslifecycle abgefragt werden. Der Fragebogen beinhaltet 12 Fragen mit vorgegebenen und offenen Antwortfeldern. Die Deployer wurden bei Rückfragen telefonisch kontaktiert.

Durchführung

Das Design dieser Studie wird mit Fokusgruppe 05 mit vier Teilnehmern erarbeitet. Es basiert auf einer Kombination von einer Schulung in Form einer Demonstration des Konfigurationslifecycle und einer anschließenden Befragung der Teilnehmer mit einem Fragebogen. Die Fokusgruppe soll die verschiedenen Fragestellungen des Fragebogens mitentwickeln und ihre Verständlichkeit prüfen. Weiterhin wird in der Gruppe der Ablauf der Demonstration abgestimmt und der passende AAL-Service für die Konfiguration ausgewählt.

Der Ablauf der Evaluation zerfällt in die folgenden Schritte:

1. Demonstration des gesamten Konfigurationslifecycle anhand des AAL-Service „Nutritional Advisors“.
 - (a) Vorstellen der Prozesse und Werkzeuge zur Entwicklerunterstützung bei der Implementierung eines konfigurierbaren AAL-Services

¹⁵CIAMI Living Lab – TSB (Spanien), CERTH-HIT und CERTH-IBBR Labs (Griechenland), Laboratory of Environment Intelligence – UPM (Spanien), Smart Home – AIT (Österreich), FZI Living Lab AAL (Deutschland)

- (b) Planen des Unterstützungsbedarfs für den Endanwender und Herunterladen des AAL-Service aus dem Online Store
 - (c) Installation des AAL-Services in der AAL-Umgebung
 - (d) Konfiguration des AAL-Services entsprechend der Endanwenderwünsche und Umgebungsanforderungen. Anschließend wurde der „Nutritional Advisor“ in seiner konfigurierten Form vorgeführt.
 - (e) Vorstellen der Wartungsmöglichkeiten für AAL-Services
 - (f) Deinstallation des AAL-Services
2. Klärung von Verständnisfragen zur Demonstration
 3. Beantwortung des Fragebogens durch die Teilnehmer wahlweise in Papierform oder online

Für die Demonstration wird der Prototyp mit dem Arbeitstitel „Jambi“ verwendet (für technische Konzeption und Implementierung vgl. Kapitel 4.1 und 5). Er ist das finale Ergebnis der ersten Iteration des Konfigurationslifecycle. Dieser Prototyp basiert auf den 63 durch die Fokusgruppe 05 validierten Anforderungen (vgl. Kapitel 3.2). Hierbei sind in „Jambi“ bereits alle fünf Kernkonzepte (vgl. Kapitel 4.1) umgesetzt: Entwicklung, Planung, Installation, Konfiguration und Wartung. Der AAL-Service „Nutritional Advisor“ ist als einer der Referenzanwendungsfälle ein repräsentativer Kandidat für die Demonstration. Weiterhin unterstützt er in seinem theoretischen Konstrukt sowie in der vorliegenden Implementierung alle Phasen des Konfigurationslifecycle.

Bevor die Teilnehmer den Fragebogen ausfüllten, bestand im Nachgang zur Demonstration die Möglichkeit Verständnisfragen zu klären. Ziel des Fragebogens ist es, die Qualitätsanforderungen an die Konfiguration zu erfassen und zu gewichten. Die Beantwortung der Fragen soll aus Deployersicht geschehen. Es wird angenommen, dass Teilnehmer, die selbst nicht zur Gruppe der Deployer gehören, aber mehr als zwei Jahre in der AAL-Domäne aktiv waren, die Rolle eines Deployers für die Befragung einnehmen können. Die Teilnehmer wurden mehrfach daran erinnert, die Fragen aus Sicht der Deployer zu beantworten.

Der Fragebogen konnte hierzu in Papierform oder online beantwortet werden. In 21 Fragen (Fragebogen siehe Anhang A.3) entschieden die Teilnehmer über die Priorisierung der relevanten Qualitätsmerkmale. Hierbei wurden Blöcke von Fragen gebildet, die auf einzelne Aspekte der Live Demonstration bzw. des Konfigurationslifecycle genauer eingehen. In der Evaluation wurden die Fragen den Teilnehmern auf Englisch vorgelegt, da davon auszugehen ist, dass die Teilnehmer der Fokusgruppen, die aus dem EU-Projekt universAAL rekrutiert werden, keine Deutschkenntnisse besitzen.

1. Stammdaten (*Personal Information*)

Q1: Welche primäre Rolle nehmen Sie in der AAL-Domäne ein? (*What is your primary role in regards to AAL?*)

Diese Frage dient der Identifikation der Art des Hintergrundwissens der Teilnehmer.

- Q2: Wie lange arbeiten Sie bereits im Themengebiet AAL? (*How long have you been working in the AAL domain?*)
Diese Frage dient der Identifikation der Tiefe des Hintergrundwissens der Teilnehmer.
- Q3: Was ist Ihr Geschlecht? (*What gender are you?*)
Diese Frage soll sicherstellen, dass die Anforderungen geschlechterunabhängig sind.
- Q4: Wie alt sind Sie? (*How old are you?*)
Es soll untersucht werden, ob das Alter bei der Gewichtung der Anforderungen eine Rolle spielt.
2. Im Online-Geschäft einen AAL-Service kaufen (*Shop for AAL-Services in the online store*)
- Q5: Als Deployer wünsche ich eine One-Click-Installation von AAL-Services aus dem uStore¹⁶, z.B. die komplette Integration ist mit einem einzigen Mausklick zu erledigen. (*As a deployer I would like to have a one-click-installation of AAL applications from the uStore, i.e. the complete integration in the environment is done by one click.*)
Diese Frage dient zur Kontrolle der Gewichtung der Effizienz und Benutzbarkeit.
- Q6: Ich, der Deployer, möchte immer einen manuellen Weg (ohne den uCC¹⁷) zum Bearbeiten der Phasen im Lifecycle (z.B. Installation, Konfiguration usw.) haben. (*I the deployer would always like to have a manual way (without using the uCC) to process a task (e.g. install, configure etc.)*).
Diese Frage dient zur Kontrolle der Gewichtung der Funktionalität.
- Q7: Für mich als Deployer ist es nicht notwendig, den uStore direkt mit dem uCC ohne externen Browser zu erreichen. (*For me as a deployer it would not be necessary to access the uStore directly in the uCC without an external browser.*)
Diese Frage dient zur Kontrolle der Gewichtung der Benutzbarkeit.
3. Installation eines AAL-Services (*Installation of AAL-Services*)
- Q8: Ich, der Deployer, würde Flexibilität im Konfigurationsprozess zugunsten der Installationsgeschwindigkeit mit dem uCC aufgeben. (*I the deployer would give up some flexibility in the configuration process to optimize the required service installation time with the uCC.*)
Eine hohe Bewertung auf der Likert-Skala befürwortet Effizienz gegenüber Funktionalität, da die Flexibilität zugunsten einer schnelleren Installation beschränkt wird.

¹⁶Der uStore ist ein Online Repository aus dem AAL-Services bezogen werden können. Die Namensgebung entstammt seiner Verwendung im universAAL-Projekt.

¹⁷Der uCC (universAAL Control Center) ist die Administrationssoftware für die Integration und Wartung von AAL-Services. Die Namensgebung entstammt seiner Verwendung im universAAL-Projekt.

- Q9: Ich, als ein Deployer, habe eher geringe Ressourcenanforderungen an den uCC, als dass ich komplexe Fallbackmechanismen benötige (z.B. Datenwiederherstellung bei Systemausfall während einer AAL-Service-Integration) (*I as a deployer prefer low resource requirements for the uCC over complex fallback mechanisms (e.g. data recovery in case of a system failure during the application integration).*)
Eine hohe Bewertung auf der Likert-Skala befürwortet Effizienz gegenüber Zuverlässigkeit. Geringen Ressourcenanforderungen ermöglichen ein effizientes Arbeiten auf Kosten Zuverlässigkeit des Gesamtsystems.
- Q10: Ich, in der Rolle als Deployer, möchte mobile Geräte (z.B. Handy, Tablet-PC) für den Deploy-Prozess nutzen. (*I in the role of a deployer would like to use mobile devices (e.g. cell phone, tablet-pc) for the deployment process.*)
Diese Frage dient zur Kontrolle der Gewichtung der Funktionalität.
4. Konfiguration und Personalisierung des AAL-Service auf die Bedürfnisse des Endbenutzers (*Configuration and personalization of AAL-Services to the end-user's needs*)
- Q11: Um mich in meiner Rolle als Deployer zu unterstützen, sollte es immer eine Standardkonfiguration geben. (*There should always be a default configuration to support me in my role as a deployer.*)
Diese Frage dient zur Kontrolle der Gewichtung der Benutzbarkeit und der Zuverlässigkeit.
- Q12: Während des Konfigurationsprozesses würde ich (der Deployer) Freiheiten in der Auswahl von Optionen gegen Fehlerschutz (z.B. vorausgefüllte Werte usw.) tauschen. (*During the configuration process, I (the deployer) would trade some limitation in my choice of features for error protection (e.g. pre-filled values etc.).*)
Eine hohe Bewertung auf der Likert-Skala befürwortet Funktionalität gegenüber Zuverlässigkeit. Eine Vielzahl von Optionen im Konfigurationsprozess erlaubt es dem Programmierer eine hohe Funktionalität einzubauen. Diese macht es schwerer die Zuverlässigkeit des Gesamtsystems sicherzustellen.
- Q13: Ich denke, der uCC muss die Benutzereingaben zu den Konfigurationsdetails eines Service nicht validieren. (*I think the uCC is not required to validate the user-set configuration details of a service.*)
Eine hohe Bewertung auf der Likert-Skala befürwortet Effizienz gegenüber Benutzbarkeit. Ohne die Validation jeder Benutzereingaben kann die Effizienz gesteigert werden. Im Falle eines Fehlers bei der Benutzereingaben treten Probleme auf, die die Benutzbarkeit einschränken.
5. Deinstallation eines AAL-Services (*Uninstallation of AAL-Services*)
- Q14: Ich bin der Deployer. Ich möchte die Möglichkeit haben, Deinstallationen rückgängig zu machen. (*I am the deployer. I would like to have the possibility to revoke the uninstallation of AAL services.*)
Diese Frage dient zur Kontrolle der Gewichtung der Funktionalität und der Zuverlässigkeit.

6. Allgemeine Fragen über den uCC und die dazugehörige AAL-Service-Integration (*General questions about the AAL-Service integration in the AAL-Environment*)

Q15: Der uCC darf schwierig und komplex sein, da der Benutzer eine große Funktionalität braucht. (*It is okay for the uCC to be difficult and complex, since the user needs extensive functionality.*)

Eine hohe Bewertung auf der Likert-Skala befürwortet Funktionalität gegenüber Benutzbarkeit. Eine hohe Mächtigkeit in den Funktionen während der Konfiguration ist wichtiger als eine einfache Benutzbarkeit des uCCs.

Q16: Neben Installieren, Managen und Deinstallieren von Services sollte sich der uCC in seiner Funktionalität mit Plug-Ins (z.B. Log-Konsole, Benutzer- und Umgebungsprofile editieren) erweitern lassen. (*The uCC should also be extensible with plug-in functionality (e.g. log view monitoring, editing user and space profiles) apart from installing, managing and uninstalling services.*)

Diese Frage dient zur Kontrolle der Gewichtung der Funktionalität.

Q17: Die Ressourcenanforderungen des uCCs sind unwichtig. (*The uCC system resources requirements are of little importance.*)

Diese Frage dient zur Kontrolle der Gewichtung der Effizienz.

Q18: Zum ordnungsgemäßen Gebrauch des uCCs ist ein gewisses Maß an Schulung des Nutzers notwendig. (*To use the uCC properly, the user needs a particular amount of training.*)

Diese Frage dient zur Kontrolle der Gewichtung der Benutzbarkeit.

Q19: Die Benutzbarkeit des uCCs (GUI, Fehlertoleranz usw.) ist wichtiger als Performanzeffizienz. (*The uCC usability (GUI, error tolerance etc.) is more important than the performance efficiency.*)

Eine hohe Bewertung auf der Likert-Skala befürwortet Benutzbarkeit gegenüber Effizienz.

Q20: Der uCC sollte auf die Verwaltung von menschlichen Dienstleistungen und Hardware erweitert werden; selbst wenn das mit Verlust von Bedienkomfort und Kernfunktionalität verbunden ist. (*The uCC should be extended to manage human- and hardware resources, even if this means losing some usability and focus on core features.*)

Eine hohe Bewertung auf der Likert-Skala befürwortet Funktionalität gegenüber Benutzbarkeit. Die Erweiterung des uCC um neue Funktionen macht seine Bedienung nicht einfacher.

7. Allgemeine Kommentare (*General comments*)

Q21: In diesem Freitextfeld konnten die Teilnehmer weitere Kommentare notieren.

Dieses Feld bietet die Möglichkeit für die Teilnehmer, abschließend auf Aspekte einzugehen, die vom Fragebogen nicht erfasst wurden.

Erfassung und Bewertung der Qualitätskriterien

Nach der Darstellung des grundlegenden Durchführung dieser Evaluation werden im zweiten Teil dieses Kapitels die verwendeten Methoden und Modelle zur Erfassung und Bewertung der Qualitätskriterien beschrieben.

Die Bewertung von Anforderungen kann in Softwareprodukten von Qualitätsmodellen unterstützt werden. Für diese Evaluation bietet sich ein Modell aus der ISO/EIC 25010 [ISO/10] an. Es zerfällt in das Product Quality Model zur Anforderungsevaluation und das Quality in Use Model zum Bewerten des Softwareprodukts. Das Qualitätsmodell Product Quality Model wurde für die AAL-Domäne leicht angepasst und in dieser Form von den Fokusgruppen 05 als gute Evaluationsbasis befunden. Es zerfällt auf erster Ebene in acht Charakteristika, die sich jeweils in weitere Subcharakteristika aufteilen. Diese verschiedenen Charakteristika und ihre weiteren Ausprägungen werden in der Norm noch genauer spezifiziert. Sie helfen, die verschiedenen Softwareeigenschaften umfassend zu beschreiben und dabei alle notwendigen Qualitätskriterien zu berücksichtigen.

Das Product Quality Model (PQM) wird gemäß den Evaluationszielen an die AAL-Domäne und den Konfigurationslifecycle angepasst. Die Anzahl der zu untersuchenden Qualitätskriterien hat sehr großen Einfluss auf die Menge der Fragen, die an die Teilnehmer zu stellen sind. Deshalb müssen aus dem PQM die wichtigsten Charakteristika herausgearbeitet werden (vgl. Abbildung 2.2). Funktionalität (Functional Suitability), Effizienz (Performance Efficency), Benutzbarkeit (Usability) und Zuverlässigkeit (Reliability) nehmen nach Ansicht der Fokusgruppen 05 und basierend auf den Pretests für den Deployer eine wichtige Rolle ein. Kompatibilität (Compatibility), Sicherheit (Security), Wartbarkeit (Maintainability) und Portierbarkeit (Portability) spielen eher eine untergeordnete Rolle und werden in den Fragebögen nicht weiter betrachtet. Im Zuge der zweiten Iteration des Konfigurationslifecycle fließen sie mit geringer Priorität ein.

Anhand der Beantwortung verschiedener Fragen drücken die Studienteilnehmer ihre Präferenzen zwischen den unterschiedlichen Qualitätskriterien aus. Damit die Evaluation geeignete Ergebnisse erzeugt, sind neben Repräsentativität und statistischer Signifikanz die Auswahl von geeigneten Teilnehmern und eine ausreichende Größe des Teilnehmerkreises wichtig. Nach Nielsen [Niel12] reichen bereits fünf Teilnehmer das beste Kosten-Nutzen-Verhältnis. Sollen quantitative Ergebnisse für die Benutzbarkeit aus der Evaluation abgeleitet werden, bedarf es nach Nielsen dagegen mindestens 20 Teilnehmer. Hieraus ergibt sich als Ziel, mindestens 20 vollständig beantwortete Fragebögen zu erhalten.

Der Kern der durchgeführten Qualitätsanalyse ist ein Ranking. Dieses basiert auf verschiedenen Merkmalen, die mit Verfahren der Entscheidungstheorie analysiert werden. Eingesetzt wird hierzu der Analytic Hierarchy Process (AHP) [Saat80]. Er ist eine Methode aus der Entscheidungstheorie und hilft durch systematisches Vorgehen, Entscheidungsprobleme strukturiert zu lösen. Dieses wird durch einen paarweisen Vergleich und ein Ranking erreicht. Für eine kleine Anzahl von Merkmalen ist dieses Verfahren sehr gut geeignet. Bei größeren Merkmalsräumen zeigt sich durch den quadratischen Anstieg der Vergleiche eine Schwäche. Die hierbei erreichte höhere Redundanz macht das Verfahren dann bezüglich Bewertungsfehlern relativ anfällig. Weiterhin kann diese Eigenschaft aber genutzt werden, um Aussagen über die Konsistenz eines Datensatzes zu treffen [KaRy97].

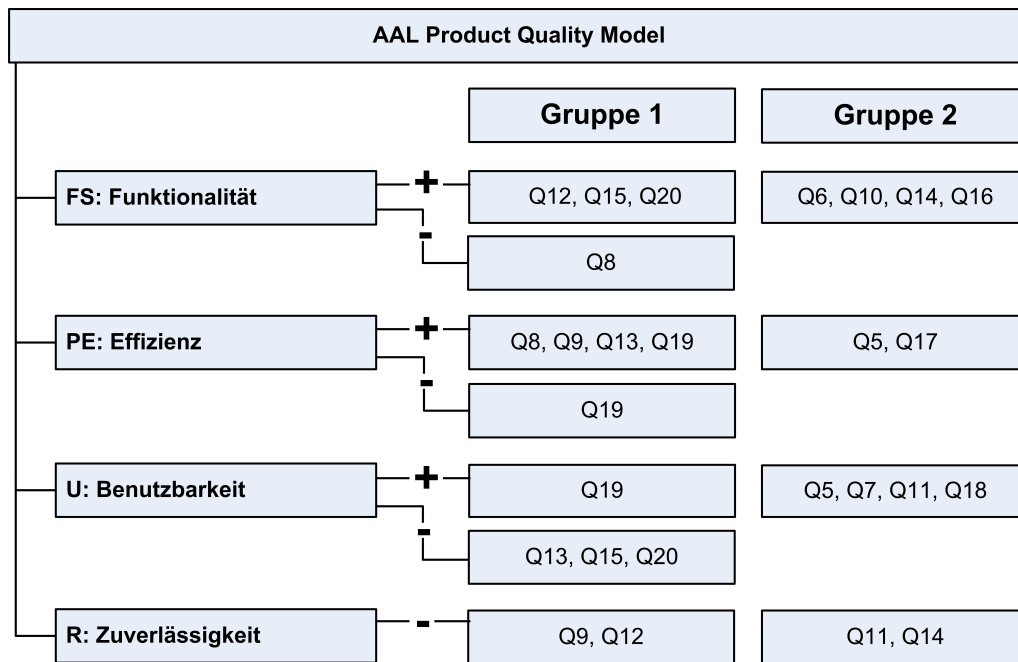


Abbildung 2.2: Angepasstes Product Quality Model mit Fragen für AAL

In Abbildung 2.2 sind die Fragen des Fragebogens in zwei Gruppen unterteilt und den Qualitätskriterien zugeordnet. Fragen der Gruppe 1 werden direkt im AHP verwendet, indem sie positiv oder negativ auf die dementsprechenden Qualitätskriterien wirken. Die Fragen der Gruppe 2 dienen als Kontrollfragen für den AHP. Sie werden außerdem mit dem Ziel formuliert, eine fundierte Einschätzung zu bestimmten Fragestellungen zu erhalten, welche im Zuge verschiedener Fokusgruppendifkussionen als interessant betrachtet wurden oder divergent diskutiert wurden.

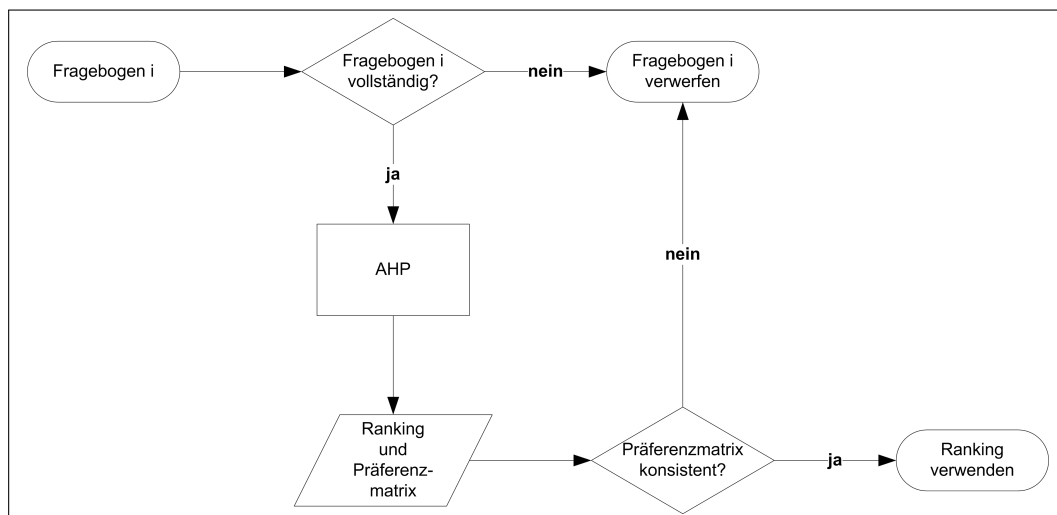


Abbildung 2.3: Vollständigkeits- und Konsistenzprüfung

Abbildung 2.3 stellt die notwendigen Schritte dar, wie man aus den einzelnen Fragebögen der Teilnehmer zum finalen Qualitätsranking gelangt. Hierbei gilt ein Datensatz der Gruppe 1 als vollständig, wenn höchstens drei Fragen nicht beantwortet

sind. Somit sind höchstens zwei Werte paarweise vergleichbar, beziehungsweise ihre vier Werte (zwei Koeffizienten und die dazugehörigen reziproken Werte) der AHP unbestimmt. In diesem Fall wird der paarweise Vergleich auf indifferent gesetzt (Wert 1). Aus den unterschiedlichen Antworten jedes Teilnehmers wird ein Prioritätsvektor erzeugt. Auf den vollständigen Datensatz wird die AHP-Methode angewendet und ein Ranking sowie das Konsistenzmaß berechnet. Nach Alonso und Lamata [JVSC⁺06] muss das Konsistenzmaß bei $\alpha = 0.15$ erfüllt werden, damit Konsistenz gegeben ist und das Ranking verwendet werden kann.

Aus allen Datensätzen, die erfolgreich auf Konsistenz geprüft wurden, wird auf Basis der einzelnen Prioritätsvektoren jedes Datensatzes ein globales Ranking erstellt. Das wichtigste Qualitätskriterium landet hierbei auf Rang 1 und das unwichtigste auf Rang 4. Die Einzelbewertungen werden anschließend nach Borda-Count [dCus14] zu einem Gesamtranking zusammengefasst.

Im Folgenden sollen die einzelnen an diesen Fragebogen angepassten Teilschritte des AHP genauer beschrieben werden:

1. Aufstellen der AHP-Matrix: Basierend auf den von der Fokusgruppe 05 gewählten vier Qualitätscharakteristika (Functional Suitability (FS), Performance Efficiency (PE), Usability (U) und Reliability (R)) wird die AHP-Matrix erstellt:

$$AHP = \begin{pmatrix} 1 & FS \langle \rangle PE & FS \langle \rangle U & FS \langle \rangle R \\ (FS \langle \rangle PE)^{-1} & 1 & PE \langle \rangle U & PE \langle \rangle R \\ (FS \langle \rangle U)^{-1} & (PE \langle \rangle R)^{-1} & 1 & U \langle \rangle R \\ (FS \langle \rangle R)^{-1} & (PE \langle \rangle R)^{-1} & (U \langle \rangle R)^{-1} & 1 \end{pmatrix}$$

Hierbei beschreibt $\langle \rangle$ einen paarweisen Vergleich zwischen den zwei Qualitätskriterien.

2. Auffüllen der Matrix: Zum Befüllen der AHP-Matrix stehen sieben Fragen aus der Gruppe 1 zur Verfügung. Die entsprechenden Antworten in der verwendeten Lickert-Skala müssen hierzu durch geeignete numerische Werte ersetzt werden. Der neutrale Wert der AHP-Matrix ist die 1 und stellt eine indifferente Aussage der befragten Person zu diesem paarweisen Vergleich dar. Hierzu entsprechend werden die anderen Werte -1 bis 3 zugeordnet. Die -1 entspricht hierbei der stärksten Ablehnung einer Aussage und die 3 der stärksten Zustimmung. Weiterhin sind zwei Besonderheiten beim Befüllen zu beachten:

- Fragen, die sich auf den gleichen, paarweisen Vergleich beziehen, werden mit ihrem Mittelwert in die Matrix eingepflegt.
- Die negativen Werte bei den Antworten werden auf den Zahlenraum $[1/2; 1]$ abgebildet. Hierbei ergibt sich der Wert für die AHP-Matrix durch $1/(2 + q)$ wobei q für den Antwortwert der Frage steht.

3. Eigenwerte der Matrix schätzen: Durch Berechnung der Eigenwerte der Matrix erhält man in diesem Schritt die Verteilung der Merkmale. Verwendet wird die Methode von Saaty [Saat80] zur Schätzung der Eigenwerte:

- Berechnung der Summe jeder Spalte
- Division der Elemente der AHP-Matrix mit der dazugehörigen Spaltensumme
- Berechnung der Zeilensumme

Die einzelnen Zeilensummen stellen die geschätzten Eigenwerte da.

4. Prioritätsvektor aufstellen: Der Prioritätsvektor P ergibt sich aus dem normierten Zeilensummenvektor Z.

5. Konsistenz prüfen: Der letzte Schritt ist die Konsistenzprüfung nach Alonso und Lamata [JVSC⁺06]. Ihre Methode baut auf dem von Saaty [Saat80] eingeführten Consistency Index und Consistency Ratio auf. Nach Alonso und Lamata ist eine AHP-Matrix hinreichend konsistent, falls gilt:

$$\lambda_{max} \leq n + \alpha(1.7699n - 4.3513) \quad (2.1)$$

α ist der Parameter für die benötigte Konsistenz und n die Anzahl der Merkmale. Das λ_{max} stellt den maximalen Eigenwert der AHP-Matrix dar und kann wie folgt genähert werden:

- Multiplikation der AHP-Matrix mit dem Prioritätsvektor P
- Elementweise Division des Ergebnisses mit dem Prioritätsvektor
- Mittelwertbildung über den Elementen

2.3.2 Evaluation des Konfigurationsframeworks

Die finale Evaluation des Konfigurationsframeworks hat das Ziel, die entwickelten Konzepte zu validieren und ihre Effizienz, Effektivität, Nützlichkeit und Vollständigkeit zu zeigen. Sie erfolgte in einem halbtägigen Workshop¹⁸. Eingeladen zur Teilnahme am Workshop waren Domänenexperten, die das Konfigurationsframework kennenlernen und anwenden. Über verschiedene Evaluationsmethoden wurden sie durch den Workshop begleitet und ihre Meinung eingeholt. Die Experten wurden gebeten, sich sowohl in die Rolle der Entwickler als auch in die Rolle der Deployer hineinzuversetzen, um alle Komponenten des Konfigurationsframeworks testen und bewerten zu können.

Zur Bewertung des entwickelten Konfigurationsframeworks und seiner Komponenten wird die Quality in Use-Methode [ISO/10] zu Grunde gelegt. Sie unterstützt die Evaluation mit fünf verschiedenen Qualitätskriterien, welche den Anwendungsraum abdecken. Diese fünf Kriterien helfen, über bestimmte Untermerkmale und Fragestellungen, die Qualität in der Benutzung der Konzepte und des Prototyps zielführend zu analysieren. Hierbei sind die einzusetzenden Evaluationsmethoden von Seiten der Norm freigestellt.

Nach näherer Betrachtung werden vier Qualitätskriterien gewählt. In Abstimmung mit der Fokusgruppe 07 (fünf Teilnehmer) wurde die Kategorie „Freedom from Risk“ wegen mangelnder Relevanz für die Konfiguration nicht betrachtet. Somit wird für

¹⁸im Begleitprogramm des 6. AAL Kongress in Berlin

Fragen		FB	D	KLM	V	P
E2.1.0	Effektivität					
E2.1.1	Konnten die Aufgaben gelöst werden?			X		
E2.1.2	Ist die Leistungsfähigkeit der Konzepte angemessen?	X	X			
E2.1.3	Wie viele und welche Probleme traten bei der Durchführung auf?				X	X
E2.2.0	Effizienz					
E2.2.1	Ist der gemessene und wahrgenommene Aufwand für die Durchführung der Aufgaben akzeptabel?	X		X	X	
E2.2.2	Sind die Konzepte zu komplex?	X	X			
E2.2.3	Kann ein Lernerfolg festgestellt werden?			X	X	
E2.3.0	Zufriedenheit					
E2.3.1	Wie komfortabel ist die Umsetzung der Konzepte für Sie?	X	X			X
E2.3.2	Halten Sie die Konzepte im Kontext der AAL-Serviceskonfiguration für nützlich?	X	X			X
E2.4.0	Vollständigkeit					
E2.4.1	Ist das Konzept komplett? Fehlen Funktionen oder gibt es überflüssige?	X	X			X

Tabelle 2.5: Kernfragen des Evaluationsworkshops. (Fragebogen (FB), Diskussionsrunde (D), Keystroke Level Modeling (KLM), Videoanalyse (V), Protokoll (P))

den Workshop eine Evaluation konzipiert, welche die Effektivität (Effectiveness), Effizienz (Efficiency), Zufriedenheit (Satisfaction) und Vollständigkeit (Context Coverage) der vorgestellten Lösung bewertet. Weiterhin validiert die Fokusgruppe 07 die Fragebögen und den Gesamt Ablauf der Evaluation. Die Fokusgruppe 08 (drei Teilnehmer) diskutiert nach der Evaluation die Implikationen für das Framework und erarbeitet wenn nötig Änderungsvorschläge.

Zu den vier Qualitätskriterien in Tabelle 2.5 werden neun Fragen definiert, die im Laufe des Evaluationsworkshops beantwortet werden sollen. Jede dieser Fragen wird durch mindestens eine Methode (Fragebogen (FB), Diskussionsrunde (D), Keystroke Level Modeling (KLM), Videoanalyse (V) oder Protokoll (P)) evaluiert. Die meisten der Fragen werden sowohl aus der Sicht des Entwicklers als auch des Deployers evaluiert. Hierfür war der Workshop so angelegt, dass sich Aufgaben für Entwickler mit Aufgaben für Deployer abwechseln und somit eine Bewertung aus beiden Blickwinkeln zuließ. Weiterhin wurde gezielt der eher theoretische Teil des Lifecycles mit praktischen Teilen der Frameworknutzung kombiniert. Das fertige Konzept für die Evaluation wird in der Fokusgruppe 08 vorgestellt und von ihr validiert.

Die Effektivität (E2.1.0) wird über die Anzahl der gelösten Aufgaben und die dabei gemachten Fehler bewertet. Die einzelnen Fehler der Teilnehmer werden in einem zweiten Schritt genauer analysiert. Die Informationen hierfür werden aus dem Videomitschnitt gewonnen. Weiterhin werden die Teilnehmer gebeten, die Leistungsfähigkeit des Konfigurationslifecycles in einem Fragebogen zu bewerten. Die Effizienzbewertung (E2.2.0) stützte sich auf den wahrgenommen Aufwand, die Einschätzung

der Komplexität der Konzepte und die Untersuchung des Lernerfolgs. Zu diesem Zweck werden neben dem Fragebogen und der Diskussion am Ende des Workshops auch die KLM und das Video eingesetzt. Die Zufriedenheit (E2.3.0) der Teilnehmer mit den Konzepten und dem Framework sowie die Vollständigkeit (E2.4.0) derselben wird in Form von Fragebögen geprüft und in der Gruppe diskutiert. Im Folgenden werden die eingesetzten Methoden dieser Evaluation ausführlicher vorgestellt.

FB - Fragebögen

Die Erfassung der Stammdaten und einige Themengebiete, die sich anders schwer erfassen lassen, werden über einen Fragebogen (vgl. Anhang A.5) abgebildet. Es handelt sich um einen Einseiter, der mit insgesamt acht Fragen und einem Freitextfeld die Meinungen in einer 5er Likert-Skala abfragt. Da nicht auf standardisierte Fragebögen zurückgegriffen werden konnte, ist der Fragebogen durch eine Bestandsaufnahme aller relevanten Prozesse und Komponenten im Gegenstandsbereich von der Fokusgruppe 07 entwickelt worden (vgl. [BoDo06, S. 253ff]).

1. Informationen zu Ihrer Person

Q1: Was ist Ihre Rolle im AAL-Umfeld?

Diese Frage dient der Identifikation der Art des Hintergrundwissens der Teilnehmer.

Q2: Wie lange arbeiten Sie schon im AAL-Umfeld?

Diese Frage dient der Identifikation der Tiefe des Hintergrundwissens der Teilnehmer.

2. Feedback

Q3: Umsetzung: Wie finden Sie die technische Umsetzung der Konfigurationskonzepte (z.B. ConfigItem, Validator, Listener)?

Diese Frage dient der Bewertung der Effizienz der vorgestellten Konfigurationskonzepte.

Q4: Funktionalität: Wie schätzen sie den Funktionsumfang der vorgestellten Konzepte ein?

Diese Frage dient der Bewertung der Vollständigkeit der vorgestellten Konfigurationskonzepte.

Q5: Leistungsfähigkeit: Können Sie Ihre AAL-Anwendungsfälle mit den vorgestellten Konfigurationskonzepten umsetzen?

Diese Frage dient der Bewertung der Leistungsfähigkeit der vorgestellten Konfigurationskonzepte.

Q6: Nachvollziehbarkeit: Wie verständlich sind die vorgestellten Konzepte?

Diese Frage dient der Bewertung der Zufriedenheit der Teilnehmer mit den vorgestellten Konfigurationskonzepten.

Q7: Komplexität: Wie schätzen Sie die Vielfalt der vorgestellten Konfigurationsmerkmale ein?

Diese Frage dient der Bewertung einer angemessenen Komplexität und Effizienz der vorgestellten Konfigurationskonzepte.

Q8: Gesamteindruck: Wie finden Sie die vorgestellten Konzepte und Umsetzungen insgesamt?

Diese Frage dient der Bewertung der Zufriedenheit der Teilnehmer mit den vorgestellten Konfigurationskonzepten.

3. Weitere Kommentare

Q9: In diesem Freitextfeld konnten die Teilnehmer weitere Kommentare notieren.

Dieses Feld bietet die Möglichkeit für die Teilnehmer, abschließend auf Aspekte einzugehen, die vom Fragebogen nicht erfasst wurden.

D - Diskussion

Eine Sonderform des Gruppeninterviews ist das Gruppendiskussionsverfahren. Es bietet die Möglichkeit, weitere Aspekte aus der Gruppe heraus einzubringen. Es wurde durch den Evaluator geleitet. Somit ergibt sich die Möglichkeit, einzelne Aspekte auch vertieft zu diskutieren oder Nachfragen zu stellen und so weitere Erkenntnisse zu gewinnen. Zur Persistierung der Ergebnisse ist hierbei ein Protokollant anwesend (vgl. [BoDo06, S. 243]).

KLM - Keystroke Level Modeling

Die Messung der Ausführzeiten beim Lösen einer Aufgabe kann Auskunft geben über den Aufwand derselben oder ob ein Lerneffekt bei den Teilnehmern auftritt. Zur Auswertung dieser Daten bedarf es einer Vergleichsbasis, dem Keystroke Level Modeling (KLM) [CaNM83]. Dieses Verfahren ermöglicht es, eine Ausführungszeit für eine gegebene Aufgabe auf 10 bis 30% zu schätzen [Saur09]. Hierbei ist zu beachten:

- Die Ausführungszeitschätzung gilt für erfahrene Benutzer.
- Für die Schätzung wird angenommen, dass alle Ausführungsvorgänge fehlerfrei sind.

KLM zerlegt dabei alle Aufgaben in Elementaroperationen (z.B. Mausbewegung, Mausklick oder Tastendruck). Diese Elementaroperationen wurden in Experimentreihen mit durchschnittlichen Ausführzeiten belegt. Die Addition dieser Zeiten für die Elementaroperationen ergibt die geschätzte Gesamtausführzeit einer Aufgabe. Der genaue Ablauf zeigt der folgende Algorithmus [Kier01]:

1. Auswahl von geeigneten Aufgaben
2. Spezifizieren der Rahmenbedingungen, um sicherzustellen, dass das KLM anwendbar ist
3. Optionalen Lösungsweg definieren
4. Zuordnung der Elementaroperationen zum Lösungsweg
5. Gegebenenfalls Warteoperatoren einfügen, um Antwortzeiten des Systems abzubilden

Kürzel	Name	Beschreibung	Zeit [sec]
K	Tastenanschlag	Drücken einer Taste auf der Tastatur	0,28
T(n)	Textsequenz	Eingabe einer Textsequenz mit Länge n	n*0,28
P	Mausbewegung	Mit dem Mauszeiger auf eine Stelle zeigen	1,1
B	Halber Mausklick	Drücken oder loslassen einer Maustaste	0,1
BB	Mausklick	Klicken einer Maustaste (drücken und loslassen)	0,2
H	Wechsel Eingabegerät	Wechsel zwischen Maus und Keyboard oder umgekehrt	0,4
M	Mentale Operation	Verarbeiten von Inhalten und/ oder nachdenken innerhalb normaler Arbeitsaufgaben (nicht für komplexe, lange dauernde Problemlösungen)	1,2
W(t)	Warteoperation	Auf Antwort des Systems warten (t = Wartezeit in Sekunden)	t

Tabelle 2.6: KLM-Operatoren mit dazugehörigen Zeiten nach [Kier01]

6. „Mentale Operatoren“ für Denkpausen einplanen
7. Standardzeiten für jeden Operator nachschlagen und zuordnen
8. Addieren aller Ausführungszeiten, um den geschätzten Gesamtaufwand zu erhalten

Die Standardzeiten und die verschiedenen Operatoren weisen in der Literatur leichte Unterschiede auf. Im Folgenden werden die Operatoren und Zeiten von Kieras [Kier01] verwendet (vgl. Tabelle 2.6).

V - Videoanalyse

Diese Erhebungsart stellt eine vergleichsweise oberflächliche Erhebungsmethode dar. Die Videoaufzeichnungen dienen hauptsächlich zur späteren Analyse bei Unklarheiten einzelner Handlungsabläufe während der Hands-On Phase. Außerdem eignen sich die Aufnahmen, um die genauen Zeitaufwände, Fehler und Probleme der einzelnen Teilnehmer zu analysieren. Die Videoaufzeichnungen stellen auch die Grundlage für die KLM dar. Die Videoaufzeichnungen entsprechen somit nicht der Videographie, die im eigentlichen Sinn hauptsächlich die Personenanalyse und deren Verhalten auswertbar macht [TuSK13].

P - Protokoll

Das Protokoll ist in erster Linie unterstützend zu den anderen Verfahren gedacht. Der Protokollant erfasst hierbei, parallel zu den anderen Methoden, alle Informationen, die mit der jeweiligen Methode nicht direkt abzubilden sind.

Zur Durchführung der Evaluation des Konfigurationsframeworks ist der Workshop in drei Phasen geteilt. Zuerst werden die AAL-Domänenexperten geeignet geschult,

um anschließend in der zweiten Phase das Erlernte praktisch anzuwenden. Anschließend werden mit einem Fragebogen und einer Diskussionsrunde die Evaluation geschlossen. Hierbei nahmen die Experten abwechselnd die Rolle der Entwickler und Deployer ein. Der konkrete Ablauf des Evaluationsworkshops gestaltet sich wie folgt:

0. Begrüßung und Einleitung (5 min)

Die Teilnehmer werden begrüßt, über den Ablauf informiert und erhalten ihre Unterlagen.

1. Theoretische Grundlagen (90 min) und Demonstration (20 min)

Als erstes werden die theoretischen Grundlagen vorgestellt. So werden alle Domänenexperten auf den gleichen Wissensstand zum Konfigurationslifecycle und -framework gebracht. Anschließend wird zur Vertiefung eine Demonstration nach der Living Lab-Methode gezeigt.

- Einleitung in das Thema der Konfiguration von AAL-Services in semantischen Middlewares am Beispiel des Projekts universAAL (15 min)
- Vorstellung des OSGi Unterbaus der universAAL Middleware (15 min)
- Vorstellung der Referenzarchitektur für semantischen AAL-Middleware und ihrer Werkzeugunterstützung (20 min)
- Vorstellung des Konfigurationslifecycle und -frameworks (40 min)
- Demonstration des Konfigurationslifecycle anhand des Konfigurationsframeworks (20 min)

Diese Phase nutzt methodisch Folien, eine Demonstration und zur Persistierung der Ergebnisse ein Protokoll.

2. Vertiefung und Hands-On (60 min)

Die erlernten Konzepte und Prozesse werden in dieser Phase von jedem Teilnehmer praktisch umgesetzt. Unter realitätsnahen Bedingungen wurden die Teilnehmer angeleitet, selber die notwendigen Schritte zur Konfiguration von AAL-Services zu programmieren. Anschließend haben sie den Anwendungsfall in der Plattform installiert und konfiguriert. Mit dem AAL-Service „Inf FRAME“ (siehe Abschnitt 5) bewegen sich die Teilnehmer durch diese Phase. Ihnen wurde ein Arbeitsplatz mit einer Entwicklungsumgebung¹⁹ gestellt. Im Workspace befand sich eine lauffähige Version der universAAL Middleware mit dem installiertem „Inf FRAME“. Auf Grundlage der vorgestellten theoretischen Grundlagen hatten die Teilnehmer unter Anleitung die Aufgabe, Konfigurationsanpassung am Inf FRAME durchzuführen. Da der Inf FRAME eine reine Webanwendung ist, bei der Sensoren über das Internet abgefragt wurden, ist das Ausstatten jedes Teilnehmers mit einem vorbereiteten Laptop für einen realitätsnahen Kontext ausreichend. Die konkreten Teilschritte zu den Konfigurationsaufgaben finden sich in Anhang A.6 und werden an dieser Stelle nur grob dargestellt.

¹⁹Eclipse - quelloffenes Programmierwerkzeug zur Entwicklung von Software

1. Aufgabe Wettervorhersage: Die erste Aufgabe besteht darin, die Anzahl der dargestellten Wettervorhersagen konfigurierbar zu machen. Ziel der Aufgabe ist ein erstes praktisches Kennenlernen des Konfigurationsframeworks. Die Teilschritte sind einfach gewählt und mit einem hohen Maß an Unterstützung gestaltet. Hilfestellung wurde in Form von Quellcode-Kommentaren und teils vordefinierten Methoden umgesetzt. Die Aufgabe besteht aus diesen Teilschritten:

- 1.1 Anlegen eines vorbereiteten `SimpleCongurationItem` und programmatisches Auslesen des konfigurierten Wertes
- 1.2 Integration eines `StandardValidators`
- 1.3 Anpassung eines bestehenden `Listeners`

2. Aufgabe Schriftfarbe: Die zweite Aufgabe zielt darauf ab, die verwendete Schriftfarbe des Infoframes konfigurierbar zu machen. Der Fokus liegt auf der Vertiefung und Festigung des Wissens, welches während der Durchführung der ersten Aufgabe erlangt wurde. Die Aufgabe setzt sich aus den folgenden, selbstständig zu lösenden Schritten zusammen:

- 2.1 Anlegen eines `SimpleCongurationItem` und programmatisches Auslesen des konfigurierten Wertes
- 2.2 Implementierung eines individuellen `Validators`
- 2.3 Implementierung eines individuellen `Listeners`

Zur Protokollierung und Persistierung wurden die folgenden Techniken und Methoden eingesetzt: Vorbereitete Arbeitsstationen, Begleitung der Aufgaben durch Unterlagen und am Beamer, Automatische Datenerhebung durch Video und KLM, Protokoll.

3. Diskussion und Befragung (30 min)

Die dritte Phase beginnt mit der Beantwortung des einseitigen Fragebogens. Anschließend werden verschiedene Aspekte des flexiblen Managements in einer angeleiteten Diskussion mit den Teilnehmern besprochen. Die Diskussion wird in einem Protokoll festgehalten. Es werden eine Diskussionsgruppe, ein Protokoll und Fragebogen eingesetzt.

2.4 Zusammenfassung

Der Methodik dieser Arbeit liegt das DSR-Modell zu Grunde. Das Modell wurde entsprechend der AAL-Domäne angepasst und mit einem methodischen Vorgehen in seinen drei Phasen unterfüttert.

In diesem Kapitel wurde vorgestellt, wie in Phase 1, der Analyse, mit Hilfe von Fokusgruppen in strukturierten Diskussionen alle am flexiblen Management beteiligten Akteure identifiziert wurden. Anschließend wurden für die gefundenen Akteure Anforderungen definiert. Ein weiterer Teil der Analysephase war die Konfigurationsbedarfserhebung bei AAL-Plattformen und -Anwendungsfällen. Hierbei

wurden auch Fokusgruppen eingesetzt, die Analyseergebnisse erarbeiteten oder validierten. Sowohl bei der Analyse der Plattformen als auch der Untersuchung der AAL-Anwendungsfälle kamen immer wieder gezielt Living Labs zum Einsatz, um ein realitätsnahes Verständnis innerhalb der Fokusgruppe zu erhalten.

In Phase 2 wurde die Methode zum Design einer Lösung für das flexible Management dargestellt. Hierzu wurde in Living Labs unter Begleitung von Fokusgruppen ein Framework entwickelt. Dieses ist in seinen Konzepten und Prozessen so aufgebaut, dass es sich leicht auf verschiedene AAL-Plattformen übertragen lässt. Methodisch wurde auf der Basis von Concept Maps und strukturierten Beschreibungen eine Referenz für die Konfiguration in AAL-Umgebungen geschaffen. Weiterhin erfolgte eine Referenzimplementierung nach der ARCADE-Methode.

Die Evaluation schließt dieses Kapitel. Für die Anforderungsevaluation wurde der Ablauf und der Fragebogen zusammen mit einer Fokusgruppe entwickelt. Damit die Probanden die Fragen kompetent und umfassend beantworten konnten, wurde hierbei auf die Präsentation des Konfigurationsframeworks in einem Living Lab zurückgegriffen. Die Nachverarbeitung der Daten aus den Fragebögen mit der AHP-Methode ermöglichte die Bewertung der Qualitätsmerkmale. Die Frameworkevaluation wurde in Zusammenarbeit mit einer Fokusgruppe erarbeitet. Hierbei wurden in einem Workshop zuerst im Rahmen von Schulungen die Konzepte und Prozesse nahegebracht und anschließend in Anlehnung an die Living Lab-Methode von den Teilnehmern implementiert und ausprobiert. Die Bewertung des Frameworks erfolgte über mehrere Kriterien entlang der Quality-in-Use Methodik.

3. Anforderungen an flexible AAL-Umgebungen

Die Flexibilität einer AAL-Umgebung hängt von mehreren Faktoren ab. Ziel dieses Kapitels ist es, diese Faktoren herauszuarbeiten und entsprechende Anforderungen zu definieren. Zuerst wird die Frage beantwortet, welche Akteure direkt an einer flexiblen AAL-Umgebung partizipieren (vgl. Kapitel 3.1). Für die identifizierten Akteure werden jeweils ihre Anforderungen an eine flexible AAL-Umgebung definiert. Einige davon sind Pflichtenanforderungen (vgl. Kapitel 3.2). Basierend auf den gefundenen Anforderungen werden aktuelle Plattformen der Hausautomatisierung auf ihr Potenzial für eine flexible AAL-Plattform untersucht. Das Ergebnis zeigt, welche Defizite behoben werden müssen bzw. welche Anforderungen aktuelle Plattformen nicht bedienen können und welchen Konfigurationsbedarf sie haben (siehe Kapitel 3.3). Abschließend werden AAL-Anwendungsfälle untersucht und ihr Konfigurationsbedarf festgestellt (siehe Kapitel 3.4). Das Kapitel schließt mit einer Zusammenfassung der Analyseergebnisse (vgl. Kapitel 3.5).

3.1 Akteure in einer AAL-Umgebung

In diesem Kapitel werden die Ergebnisse der Fokusgruppe 07 bezüglich der Akteurenanalyse vorgestellt. Hierzu konnten die in der Methodik vorgestellten Quellen teilweise als Diskussionsgrundlage für die Fokusgruppe genutzt werden.

- Eberhardt [Eber09] bietet keine einheitliche Notation der Akteure und die Zielgruppen sind nicht immer trennscharf definiert. Es wurden aus dieser Arbeit nur die genannten Institutionen in die Diskussionsgrundlage übernommen.
- Das Projekt Amigo [Geor05, S.205ff] bietet eine Aufstellung verschiedenen Stakeholder für die weitere Diskussion.
- Das EU-Projekt MPower [Wald07, S.20ff][Benc06, S.18f] ordnet seine Akteure in eine „pre-release“- und „release“-Phase ein und hat seinen Schwerpunkt auf der Entwicklerperspektive und den damit verbundenen Anforderungen [Wald07, S.20ff][Benc06, S.18f], die für die Fokusgruppen aufbereitet wurden.

- Aus OASIS [IsGVG08b, S.8ff] konnten hauptsächlich Endbenutzer und ihre Anforderungen entnommen werden.
- PERSONA [MuRW08] trägt mit einer ausführlichen Aufstellung und Gliederung der Rollen einen großen Anteil zur Diskussionsbasis bei.
- SOPRANO [Empi07, S.10ff] vervollständigt die Diskussionsbasis durch verschiedene Arten von zu unterstützenden älteren Personen.
- Das Projekt VAALID kennt drei Akteurgruppen [Scha08, S.48ff] die aufgenommen wurden.

Gestützt auf diese Daten gestalten sich die Ergebnisse der Fokusgruppe wie folgt. Akteure einer AAL-Umgebung sind Personen, Gruppen oder Organisationen, die Interesse oder Vorbehalte bezüglich der AAL-Umgebungen haben. Im Sinne der ISO/IEC/IEEE 42010:2011¹ ist das Ziel dieser Untersuchung die „Architects“ (Architekten) und die „Acquirer“ (Erwerber) zu identifizieren. Im Laufe der Untersuchung konnten fünf relevante Gruppen identifiziert und zu der AALIANCE Stakeholder-Standardisierung [vdBCW10] in Beziehung gesetzt werden. Es wurde darauf geachtet, nur Akteure zu wählen, die direkt mit der AAL-Umgebung in Kontakt kommen und direkt von einer gesteigerten Flexibilität profitieren würden. Die finalen Akteurgruppen sind in Abbildung 3.1 gegenübergestellt, zu den bekannten Definitionen der AALIANCE und IEEE in Relation gesetzt und mit einer kurzen Beschreibung versehen.

Akteurgruppen	Beschreibung	AALIANCE	ISO/IEC/IEEE 42010:2011
Endanwender	Profitiert direkt von den Vorteilen der AAL-Umgebung.	Primary	Acquirer
Entwickler	Entwickelt Software und Hardware für die AAL-Umgebung.	Tertiary/ Secondary	Acquirer/ Architect
Deployer	Installiert AAL-Umgebungen und integriert AAL-Lösungen bzw. Technologien.	Secondary	Acquirer
AAL-Anbieter	Hilft beim Aufbau einer AAL-Umgebung oder bei deren Erweiterung.	Secondary	Architect
Assistenz Anbieter	Stellt Dienstleistungen für den Endanwender in Kooperation mit Deployern und Entwicklern.	Secondary	Acquirer

Abbildung 3.1: Übersicht zu den einzelnen Akteuren in Relation zu den Definitionen der AALIANCE und ISO/IEC/IEEE 42010:2011.

3.1.1 Endanwender

Endanwender sind die Hauptzielgruppe von AAL-Umgebungen und lassen sich in vier Untergruppen aufteilen. Diesen Untergruppen ist gemein, dass sie direkt von

¹Systems and software engineering — Architecture description, the latest edition of the original IEEE Std 1471:2000, Recommended Practice for Architectural Description of Software-intensive Systems

den Vorteilen der AAL-Umgebung profitieren. Als Endanwender sind sie reine Konsumenten und gehören somit zu den Acquirern. Leider ist die landläufige Definition von Konsumenten als „non-reseller“ in AAL nicht tragbar, da nicht immer der Konsument die Leistung bezahlt oder bewertet bzw. auswählt. Oft erfolgt die Leistungsbewertung durch Fachpersonal mit pflegerischem oder medizinischem Kontext. Auch die Bezahlung der Leistung kann von anderen Akteuren wie der Kranken- oder Pflegekasse vorgenommen werden. In der AAL-Domäne profitieren die Endanwender von AAL-Umgebungen. Im Sinne der AALIANCE entsprechen sie den Primary Stakeholdern.

Der Bewohner ist ein Endanwender in der AAL-Umgebung. Er profitiert direkt von den AAL-Anwendungsfällen, die durch die AAL-Plattform bereitgestellt werden. Die Nutzung der AAL-Anwendungsfälle durch den Bewohner geschieht intuitiv. Weiterhin gibt es keine Einschränkung bezüglich des technischen Verständnisses der Bewohner. Sie sind auch in ihrer Altersklasse nicht weiter eingeschränkt und können jede geistige wie physische Diagnose aufweisen.

Der Nutzen der AAL-Plattform sollte sich für ihn proportional zum Aufwand verhalten. Der Bewohner wird einen nicht nur finanziellen Aufwand betreiben müssen, um die AAL-Plattform anzuschaffen. Außerdem bedarf es einer Gewöhnung des Bewohners an die gesamte AAL-Umgebung. Diesem Aufwand sollte ein möglichst hoher Nutzen gegenüberstehen. Jeder Bewohner erwartet von einer AAL-Umgebung bestimmte Funktionen, die seine Bedürfnisse befriedigen. Dies kann bei einfachen Grundfunktionen (z.B. der Hausautomatisierung) anfangen und endet bei der Erweiterbarkeit durch neue AAL-Anwendungsfälle. Die AAL-Umgebung muss sich im geeigneten Maß an die individuellen Bedürfnisse des Endbenutzers anpassen lassen. Diese Adaption bezieht sich sowohl auf die Möglichkeiten, einzelne AAL-Anwendungsfälle anzupassen, als auch auf die Möglichkeit, das Gesamtsystem durch neue Hard- und Software zu erweitern. Die Installation solcher neuen Komponenten sollte schnell und einfach erfolgen können. Mit diesen Forderungen ist die Zukunftsfähigkeit der AAL-Umgebung eng verknüpft. Die AAL-Plattform verwaltet sensible Daten des Bewohners, so dass für ihn auch der Datenschutz eine Rolle spielt. Neben der Unterstützung im Alltag sollte die AAL-Plattform auch zur Sicherheit des Bewohners beitragen. Hierbei ist es natürlich wünschenswert, ein möglichst robustes System zu haben, welches bestimmte Qualitätsstandards einhält. Weiterhin ist es wichtig, dass die Ästhetik des Gesamtsystems stimmt und es sich dem Bewohner transparent präsentiert. Das System sollte sich auf die Gewohnheiten des Bewohners einstellen lassen und allgemein leicht zu bedienen sein.

Die formellen Betreuer sind Personen, die für ihre Pflegetätigkeit finanziell entlohnt werden. Sie sind meist Angestellte einer Pflegeeinrichtung oder medizinisches Personal. Sie stehen im regelmäßigen Kontakt zum Bewohner und interagieren mit der AAL-Umgebung. Sie vollziehen betreuende Dienste und pflegerische Leistungen.

Für die formellen Betreuer ist es interessant, den Zustand der AAL-Plattform und einzelner AAL-Anwendungsfälle aus der Ferne zu erreichen. Hierbei ist auch die Möglichkeit eines direkten Kontakts zwischen formellem Betreuer und

Bewohner wünschenswert. Auch für sie spielt die Bedienbarkeit der Systeme eine wichtige Rolle. Insgesamt soll eine AAL-Plattform die formellen Betreuer entlasten und ihnen helfen, ihre Betreuungstätigkeit besser zu planen und auszuführen. Es sollte keine Überwachung ihrer Arbeit durch die Arbeitgeber entstehen. Weiterhin sollten alle AAL-Plattform-Komponenten hygienisch und ggf. auch desinfizierbar sein.

Die informellen Betreuer sind meist aus dem direkten Personenumfeld des zu betreuenden Bewohners und stehen in stetigem Kontakt mit ihm. Oft sind es die Familie, Freunde oder Nachbarn, die unentgeltlich die gleichen Aufgaben übernehmen wie ein formaler Betreuer, ohne explizit dafür ausgebildet zu sein.

Auch für sie ist die Zustandsabfrage der AAL-Plattform sowie eine geeignete Bedienung der Plattform und Anwendungsfälle wünschenswert. Die AAL-Umgebung sollte den sozialen Kontakt zwischen informellen Betreuer und Bewohner fördern und für den Betreuer eine Entlastung sein bzw. ihm beim Planen seiner Tätigkeiten unterstützen.

Die Besucher sind Gäste, die den Bewohner unregelmäßig besuchen.

Sie müssen nicht mit der AAL-Umgebung interagieren können, sollten aber auf keinen Fall von ihr eingeschränkt werden. Für sie ist der Datenschutz bezüglich der über sie von der AAL-Plattform gesammelten Informationen wichtig. Weiterhin wollen sie sich gewohnheitsorientiert in der AAL-Umgebung verhalten. So müssen beispielsweise einfache Funktionen der Hausautomatisierung instinktiv bedienbar sein.

3.1.2 Entwickler

Entwickler erstellen AAL-Lösungen für den Endanwender durch die Entwicklung von AAL-Anwendungsfällen und Technologien in Form von Hardware und Software für eine AAL-Umgebung. Sie gehören zu den Acquirers und Tertiary Stakeholdern, wenn sie AAL-Anwendungsfälle implementieren. Arbeiten sie jedoch direkt an der AAL-Plattform, sind sie den Secondary Stakeholdern und Architects zuzuordnen.

Der Software-Entwickler erweitert die AAL-Plattform durch seine Software. Er kann von einem AAL-Anbieter beauftragt sein und aus finanziellem oder aus eigenem Interesse handeln. Die Erweiterungen können sich direkt auf die AAL-Middleware beziehen oder auf AAL-Anwendungsfälle, welche die AAL-Plattform und damit die AAL-Umgebung um zusätzliche Funktionalitäten erweitern. Hierbei treten sie selbst nicht in Kontakt mit der Gruppe der Endanwender.

Für seine Arbeit ist ihm wichtig, dass er einen geringen Implementierungsaufwand hat bzw. ihm eine geeignete Entwicklungsumgebung bereitgestellt wird. Hierzu gehört auch der Wunsch nach einer geringen Einarbeitungszeit und geeigneter Werkzeugunterstützung. Besonders hervorzuheben sind Diagnosemöglichkeiten zur Fehlerfindung und -vermeidung während der Implementierung und des Betriebs sowie eine gute Schnittstellenbeschreibung. Die AAL-Plattform sollte auch die Abstraktion der Hardware beherrschen, so dass der Entwickler sie unabhängig von anderen Parametern (z.B. Protokoll, Hersteller)

verwenden kann. Weiterhin spielt in der Entwicklung Wiederverwendbarkeit von Code eine große Rolle. Auch die Kompatibilität der Entwicklung zu anderen Systemen oder Standards ist wichtig. Insgesamt sollte auch eine gute Transparenz in der AAL-Middleware herrschen, so dass der Entwickler schnell Möglichkeiten, Funktionalität und Konzepte erfassen kann. Weiterhin sollte ein großer potenzieller Kundenstamm für die Implementierung existieren, der über geeignete Distributionswege zu erreichen ist. Nur so kann der Software Entwickler wirtschaftlich arbeiten und seine finanziellen Interessen wahren.

Der Hardware-Entwickler implementiert neue Komponenten für die AAL-Plattform. Hierbei kann er von einem AAL-Anbieter beauftragt sein und aus finanziellem oder aus eigenem Interesse handeln. Normalerweise wird er seine Produkte nicht direkt vertreiben, sondern über einen AAL-Anbieter verkaufen. Sollten zu einer Hardware Softwarekomponenten benötigt werden (z.B. Treiber) ist ein Software-Entwickler einzubeziehen. Da Hardware bis auf vorhandene Standards nicht direkt von der AAL-Plattform abhängt, bewegt sich der Hardware-Entwickler eher auf einer Metaebene.

Natürlich steht für ihn die Wirtschaftlichkeit seiner Entwicklungen im Vordergrund. Hierfür ist eine gute Schnittstellenbeschreibung wichtig. Damit er dies durchsetzen kann, bedarf es weiterhin einfacher Distributionswege und eines potenziell großen Kundenstammes. Um seine Arbeit zu unterstützen, sollte die AAL-Plattform eine hohe Kompatibilität an den Tag legen und sich für den Hardware-Entwickler in Bezug auf Möglichkeiten, Funktionalitäten und Hardware Komponenten transparent präsentieren.

3.1.3 Deployer

Die Gruppe der Deployer besteht aus den Rollen der Handwerker und Hausmeister. Sie gehören zu den Technologieanbietern und sorgen dafür, AAL-Plattformen zu installieren und AAL-Lösungen in eine AAL-Umgebung zu integrieren und die Instandhaltung der AAL-Umgebung. Sie sind nach AALIANCE Secondary Stakeholder und in der IEEE Acquirer, da sie in ihren Aufgaben im Sinne des Endanwenders handeln und durch AAL-plattformsspezifische Werkzeuge unterstützt werden.

Der Handwerker nimmt die Installation der AAL-Plattform beim Bewohner vor. Hierzu wird er vom Bewohner oder pflegenden Instanzen beauftragt. Der Handwerker ist ausgebildet, um eine AAL-Plattform zu installieren und zu konfigurieren. Auch für Reparatur- und Wartungsarbeiten kann er herangezogen werden.

Hierzu sind klare Schnittstellen und definierte Integrationsprozesse wünschenswert. Auch eine geringe Einarbeitungszeit in die AAL-Plattform, mit ihren die AAL-Umgebung aufspannenden Komponenten, wäre zielführend. Um seine Arbeit zu unterstützen, sollte es Werkzeuge geben, die dem Handwerker durch die Integrationsaufgaben begleiten. Hierbei könnten sie Schritte wie die Installation und Konfiguration der Plattform automatisieren und Diagnosemöglichkeiten anbieten. Ein skalierbares System wäre auch für den Handwerker praktisch. Da er hauptsächlich aus finanziellem Interesse agiert, muss die Wirtschaftlichkeit für ihn gegeben und die Haftbarkeit geklärt sein.

Der Hausmeister übernimmt instandhaltende Maßnahmen in Form von Wartungsaufgaben in der AAL-Umgebung. Er ist nicht so gut ausgebildet im Bereich der AAL-Plattformen und übernimmt im Gegensatz zum Handwerker kleinere Aufgaben, die kein besonderes Fachwissen erfordern (z.B. Batterien im Sensor tauschen).

Hierzu wäre ein Fernzugriff auf das System hilfreich, um sich über den momentanen Status informieren zu können. Die anfallenden Wartungsarbeiten sollten aufgrund des Systemaufbaus selten auftreten und einfach auszuführen sein. Die AAL-Plattform sollte robust sein. Der Hausmeister sollte sich schnell in die Plattform einarbeiten können und im Fehlerfall seinerseits einen guten Support erhalten können. Weiterhin ist für den Hausmeister wichtig, dass die Haftungsfragen geklärt sind und es nicht zu einer Leistungsüberwachung durch seinen Arbeitgeber auf Basis der AAL-Umgebung kommt.

3.1.4 AAL-Anbieter

Der AAL-Anbieter hilft beim Aufbau einer AAL-Umgebung oder bei deren Erweiterung. Sie entsprechen der Gruppe der AALIANCE Secondary Stakeholder und sind nach der IEEE als Architect zu sehen. Mit seiner Hilfe werden die passenden Komponenten entsprechend der Bedürfnisse des Bewohners ausgewählt.

Hierbei hat der AAL-Anbieter hauptsächlich finanzielle Interessen. Darum ist die Kundenzufriedenheit für ihn ein wichtiges Thema. Damit er gezielt auf die Bedürfnisse der Kunden eingehen kann, sollte die AAL-Plattform möglichst flexibel sein, so dass sich beispielsweise herstellerunabhängig die Sensorik einbinden lässt. Auch die Integration von medizinischen Geräten nach dem Medizinproduktegesetz (MPG) ist denkbar. Die AAL-Plattform sollte also möglichst mächtig sein in Bezug auf Umsetzungsmöglichkeiten und Erweiterungen durch AAL-Anwendungsfälle und dennoch skalierbar. Die zusammengestellte AAL-Umgebung sollte möglichst robust funktionieren. Weiterhin steht die Sicherheit des Systems im Mittelpunkt. Dies gilt sowohl für den Bewohner, der durch Probleme der AAL-Plattform nicht in Gefahr geraten darf, als auch Sicherheit gegenüber Fremden, wie beispielsweise Schutz vor Einbrechern. Neben einer einfachen und transparenten Nutzung der AAL-Plattform ist für den AAL-Anbieter auch eine Evaluationsmöglichkeit hilfreich. Durch einfaches Feedback über die AAL-Umgebung kann er so auch Fehler identifizieren und die AAL-Plattform stetig verbessern. Wie auch andere Akteure benötigt der AAL-Anbieter gute Distributionswege, um sein Produkt möglichst lange und effizient anzubieten. Weiterhin ist die Frage nach einer geklärten Haftbarkeit (Sach- und Personensachen) durch die AAL-Plattform für den AAL-Anbieter interessant.

3.1.5 Assistenz-Anbieter

Die Assistenz Anbieter unterstützen den Endanwender hauptsächlich durch Dienstleistungen in Kooperation mit anderen Akteuren. Somit gehören sie zur Gruppe der Secondary Stakeholder und zu den Acquirer. Zu dieser Gruppe gehören die Pflegeeinrichtungen, die externen Dienstleister und der Medizinischer Dienst der Krankenkassen (MDK).

Die Pflegeeinrichtungen verwalten das Betreute Wohnen und somit ggf. auch AAL-Plattformen, die vor Ort installiert sind.

Hierbei stehen finanzielle Interessen im Vordergrund und die AAL-Plattform sollte die Möglichkeit lukrativer Finanzierungen (beispielsweise Leasing) bieten. Die Pflegeeinrichtungen legen hierbei Wert auf eine zentrale Verwaltungsmöglichkeit von mehreren AAL-Umgebungen. Hierzu muss das einzelne System gute Schnittstellen bieten, damit beispielsweise eine automatische Erfassung von Notfällen in der Zentrale möglich wird. Es wäre hilfreich, wenn auch Evaluationsmöglichkeiten für die Pflegeeinrichtungen existieren, um die AAL-Plattform und ihre AAL-Anwendungsfälle zu erfassen und zu verbessern. Ebenso eine automatische Abrechnung von erbrachten Leistungen durch die Unterstützung der AAL-Plattform denkbar. Insgesamt sollte die Plattform ressourcenoptimierend sein, sowohl bezüglich der Planungs- und Pflegeprozesse der Einrichtungen als auch direkt auf Arbeitszeit und Ressourcen bezogen.

Externe Dienstleister unterstützen Pflegeeinrichtungen im Bereich des Betreuten Wohnens auf regelmäßiger Basis oder für einzeln anfallende Aufgaben. Beispiele hierfür sind Reinigungsdienste, Catering-Services, Sanitäter oder andere Dienstleistungen.

Für die Gruppe der Dienstleister ist es wichtig, dass die AAL-Plattform ihre Prozessabläufe nicht verändert, sondern sie gewohnt agieren können. Teilweise wären Schnittstellen zur AAL-Plattform hilfreich, über die sich der externe Dienstleister mit der AAL-Umgebung verbinden kann und sich somit besser in sie einfügt. Ein gutes Beispiel hierfür ist ein integrierter Hausnotruf.

Medizinischer Dienst der Krankenkassen (MDK) übernimmt die Prüfung der Pflegebedürftigkeit für die Krankenkassen. Der Kontakt zum Bewohner und der AAL-Umgebung ist dementsprechend gering.

Für ihn steht die Transparenz einer solchen AAL-Plattform und ggf. die Möglichkeit einer Evaluation der Bewohnersituation zur Unterstützung seiner Arbeit im Vordergrund.

3.1.6 Zusammenfassung

Die elf identifizierten Akteure spiegeln alle beteiligten Rollen in einer AAL-Umgebung wider, die direkt von einem flexiblen Management der AAL-Umgebungen betroffen sind. Der Endanwender profitiert direkt von den Vorteilen der AAL-Umgebung und ist darum an einer flexiblen Anpassung der Umgebung an seine Bedürfnisse stark interessiert. Der Entwickler erstellt Soft- und Hardware für die AAL-Umgebung und die darin befindliche AAL-Plattform. Er benötigt eine flexible Umgebung für eine möglichst uneingeschränkte Entwicklung. Der Deployer baut die AAL-Umgebung, indem er verschiedene AAL-Anwendungsfälle und Technologien auf einer AAL-Plattform kombiniert. Je flexibler sich die AAL-Plattform ihm gegenüber darstellt, desto besser kann er auf die Bedürfnisse des Endanwenders eingehen. AAL-Anbieter profitieren von einer flexiblen AAL-Umgebung durch die damit verbundene Steigerung der Möglichkeiten an angebotenen AAL-Anwendungsfällen und anderen Komponenten durch bedürfnisgerechte Lösungen für seine Kunden, die Endanwender. Assistenz-Anbieter können durch eine hohe Flexibilität der AAL-Umgebung besser ihre Dienstleistungen auf sie anpassen und in dieselbe besser integrieren.

Welche Anforderungen genau durch ein flexibles Management für die einzelnen Akteure gefordert sind, wird im folgenden Kapitel genauer beschrieben.

3.2 Anforderungen an flexible AAL-Umgebungen

Nach der Erfassung der verschiedenen Akteure, die an einer AAL-Umgebung beteiligt sind, werden in diesem Kapitel die Anforderungen, die von ihnen ausgehen, beschrieben. Während der Analyse und Zuordnung in den Fokusgruppe 01 wurde darauf geachtet, allen Akteuren in den Anforderungen gerecht zu werden. Die teilweise heterogenen Wünsche der verschiedenen Akteure wurden auf die Anforderungen übertragen und akteurübergreifenden Anforderungskategorien zugeordnet. Die folgenden Kategorien wurden von den Teilnehmern der Fokusgruppe 01 identifiziert:

Anwendungsfälle: Die AAL-Umgebung wird durch AAL-Anwendungsfälle um bestimmte Funktionalitäten erweitert, die hauptsächlich den Endanwender unterstützen. Hierbei verbindet ein AAL-Anwendungsfall sowohl Hardware wie Software und kann auch menschliche Ressourcen beinhalten. Die verschiedenen AAL-Lösungen in einer AAL-Umgebung sind also die Aggregation der auf einer AAL-Plattform laufenden AAL-Anwendungsfälle.

Benutzer-Schnittstelle: Die Benutzer-Schnittstelle (User Interface; kurz UI) wird von jedem der beschriebenen Akteure zur Interaktion mit der AAL-Umgebung und den AAL-Anwendungsfällen genutzt. Dabei ist es egal, ob es sich um den einfachen Lichtschalter an der Wand handelt, eine komplexere UI auf dem Tablet oder um ganz andere (z.B. akustische) Kommunikationswege.

Hardware: Sie beschreibt jede Form von Geräten, die mit der AAL-Plattform in Interaktion stehen. Die häufigsten Vertreter sind Sensoren und Aktoren.

Support: Unter dem Sammelbegriff Support werden Anforderungen in Bezug auf die Supportfähigkeit der AAL-Plattform gesammelt. Hierbei geht die Unterstützung der Akteure nicht von integrierten Anwendungsfällen aus, sondern von externen Informations- und Hilfsquellen, die Nicht-Endanwendern helfen.

Werkzeuge: Diese Kategorie erfasst Anforderungen an Werkzeuge, die von Nicht-Endanwendern genutzt werden. Diese Werkzeuge kommen besonders beim Management der AAL-Plattform zum Einsatz.

Die Fokusgruppe 01 konnte insgesamt 63 Anforderungen mit Flexibilitätsbezug orthogonal zu den Akteuren erarbeiten, welche sich auf die fünf Kategorien verteilen. Weiterhin spezifizierte die Fokusgruppe 01 aus den 63 Anforderungen 10 Pflichtanforderungen, ohne die eine flexible AAL-Umgebung für sie nicht denkbar wäre:

102P Grundkonzepte der Hausautomatisierung

201P Einfach nutzbares UI

211P Nutzt bereits vorhandene Strukturen

302P Schaltbare Sensoren und Aktuatoren

304P Erweiterbare Sensoren und Aktuatoren

405P Gute Dokumentation

409P Passendes Lizenzmodell

501P Sensor und Aktuator-Management

507P Erweiterbarkeit

508P Mächtige Entwicklungssprache

Die folgende Liste beschreibt alle 63 Anforderungen im Detail mit ihren dazugehörigen Akteuren. Die Pflichtanforderungen sind in der folgenden Aufzählung durch ein P hinter der ID gekennzeichnet.

Anwendungsfälle

<i>ID:</i> 101	<i>Titel:</i> AAL-taugliche Anwendungsfälle
<i>Beschreibung:</i> Es sollen Anwendungsfälle vorhanden sein, die den speziellen Bedürfnissen der Endanwender in einer AAL-Umgebung gerecht werden.	
<i>Akteure:</i> Bewohner, formelle Betreuer, informelle Betreuer, AAL-Anbieter	
<i>ID:</i> 102P	<i>Titel:</i> Grundkonzepte der Hausautomatisierung
<i>Beschreibung:</i> Das AAL-System muss den Stand der Technik im Bereich der Hausautomatisierung anbieten. Das beinhaltet grundlegende Interaktion mit Sensoren und Aktoren und deren Verknüpfung über ein Regelwerk.	
<i>Akteure:</i> Bewohner, Besucher	
<i>ID:</i> 103	<i>Titel:</i> Gute Konfigurierbarkeit
<i>Beschreibung:</i> AAL-Anwendungsfälle sollen sich durch Konfigurierbarkeit an die Bedürfnisse des Endanwenders anpassen lassen.	
<i>Akteure:</i> Bewohner, formelle Betreuer, informelle Betreuer, Handwerker, AAL-Anbieter	
<i>ID:</i> 104	<i>Titel:</i> Beachtung des Datenschutzes
<i>Beschreibung:</i> Die AAL Anwendungsfälle sowie das AAL-System gehen mit sensiblen Daten der Endanwender um. Es sind Datenschutzrichtlinien zu befolgen.	
<i>Akteure:</i> Bewohner, formelle Betreuer, informelle Betreuer, Besucher, Hausmeister, Pflegeeinrichtungen	
<i>ID:</i> 105	<i>Titel:</i> Aktive Missbrauchsverhinderung
<i>Beschreibung:</i> Das AAL-System und die AAL-Anwendungsfälle sollten den Missbrauch von Daten verhindern und aktiv entgegenwirken.	
<i>Akteure:</i> Bewohner, formelle Betreuer, AAL-Anbieter	
<i>ID:</i> 106	<i>Titel:</i> Robustheit
<i>Beschreibung:</i> Eine hinreichende Robustheit der AAL-Services ist notwendig.	
<i>Akteure:</i> Bewohner, Hausmeister, AAL-Anbieter	

<i>ID:</i> 107	<i>Titel:</i> Transparenz
<i>Beschreibung:</i> Die AAL-Anwendungsfälle müssen sich gegenüber den Endanwender transparent gestalten.	
<i>Akteure:</i> Bewohner, informelle Betreuer, Hausmeister, AAL-Anbieter, MDK	

<i>ID:</i> 108	<i>Titel:</i> Gewohnheit
<i>Beschreibung:</i> Die Gewohnheiten der Endanwender dürfen nicht verändert werden. Dies schließt das Erlernen neuer Fertigkeiten nicht aus, sondern verlangt, dass gewöhnliche Tätigkeiten und Handlungsabläufe auch unter Nutzung von AAL-Anwendungsfällen beibehalten werden können.	
<i>Akteure:</i> Bewohner, informelle Betreuer, Besucher	

<i>ID:</i> 109	<i>Titel:</i> Unterstützung des sozialen Umgangs
<i>Beschreibung:</i> Es sollte Anwendungsfälle geben, die den sozialen Umgang mit anderen Menschen fördern.	
<i>Akteure:</i> Bewohner, informelle Betreuer	

<i>ID:</i> 110	<i>Titel:</i> Entlastung der betreuenden Akteure
<i>Beschreibung:</i> Die Akteure der Gruppen der formellen und informellen Betreuer sowie ihrer Organisationen sollen durch AAL-Anwendungsfälle entlastet werden.	
<i>Akteure:</i> formelle Betreuer, informelle Betreuer, AAL-Anbieter, Pflegeeinrichtungen, externe Dienstleister	

<i>ID:</i> 111	<i>Titel:</i> Geheimhaltung persönlicher Daten
<i>Beschreibung:</i> Die Geheimhaltung der durch AAL-Anwendungsfälle anfallenden Daten der verschiedenen Akteure in der AAL-Umgebung muss gewährleistet sein. Weiterhin dürfen nur berechtigte Akteure Zugriff auf diese Daten haben. Eine entsprechende Sicherheits- und Managementkomponente in der AAL-Plattform soll vorhanden sein.	
<i>Akteure:</i> informelle Betreuer, AAL-Anbieter, Pflegeeinrichtungen, MDK	

<i>ID:</i> 112	<i>Titel:</i> Offene Schnittstellen
<i>Beschreibung:</i> AAL-Anwendungsfälle sollten offene Schnittstellen besitzen, um mit anderen Komponenten oder Dienstleistern innerhalb und außerhalb der AAL-Umgebung zu kommunizieren.	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler, Handwerker, Hausmeister, Pflegeeinrichtungen, Externe Dienstleister	

Benutzerschnittstellen

<i>ID:</i> 201P	<i>Titel:</i> Einfach nutzbares UI
<i>Beschreibung:</i> Die gesamte Interaktion mit dem System über die verschiedenen UIs muss einfach sein. Dies fängt an bei einfacher Hardware (z.B. Schalter) und endet bei komplexen Möglichkeiten der Interaktion (z.B. UI am Computer).	
<i>Akteure:</i> Bewohner, informelle Betreuer, Hausmeister, AAL-Anbieter, MDK	

<i>ID:</i> 202	<i>Titel:</i> Gewohnheit
<i>Beschreibung:</i> Die Benutzerschnittstellen sollen sich gewohnheitsorientiert verhalten. Dies fordert, dass sich neue Schnittstellen an den Gewohnheiten der Endbenutzer orientieren und sich intuitiv bedienen lassen. Weiterhin sollen auch alte Bedienkonzepte weiter mit und in der AAL-Umgebung funktionieren.	
<i>Akteure:</i> Bewohner, informelle Betreuer, Besucher, AAL-Anbieter	
<i>ID:</i> 203	<i>Titel:</i> Gute Konfigurierbarkeit
<i>Beschreibung:</i> Die Bedienung soll technisch konfigurierbar sein, um auf verschiedene AAL-Anwendungsfälle und die AAL-Umgebung anpassbar zu sein.	
<i>Akteure:</i> Bewohner, Handwerker, Pflegeeinrichtungen	
<i>ID:</i> 204	<i>Titel:</i> Erweiterbarkeit der Benutzerschnittstellen
<i>Beschreibung:</i> Die Benutzerschnittstellen müssen sich auf Sensor- und Aktor-Seite beliebig erweitern lassen. Hierzu zählen neben der Einbindung von neuer Hardware (z.B. Lichtschalter, Tablet) auch die Software-Bedienelemente beispielsweise, auf einem Tablet.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Hardware-Entwickler, Handwerker, AAL-Anbieter	
<i>ID:</i> 205	<i>Titel:</i> Robustheit
<i>Beschreibung:</i> Die Nutzeroberfläche in Hardware und Software muss hinreichend robust sein.	
<i>Akteure:</i> Bewohner, Hausmeister, AAL-Anbieter	
<i>ID:</i> 206	-In der Konsolidierung mit Fokusgruppe 01 gestrichen-
<i>ID:</i> 207	<i>Titel:</i> Ästhetik
<i>Beschreibung:</i> Die Benutzeroberfläche muss optisch ansprechend gestaltbar sein.	
<i>Akteure:</i> Bewohner	
<i>ID:</i> 208	<i>Titel:</i> Bedürfnisse
<i>Beschreibung:</i> Die Benutzerschnittstellen müssen sich auf die Bedürfnisse der Akteure anpassen lassen. Dies gilt sowohl auf allen Ebenen vom technischen Verständnis bis hin zu krankheitsbedingten Einschränkungen.	
<i>Akteure:</i> informelle Betreuer, Software-Entwickler, Handwerker, Hausmeister, AAL-Anbieter, Pflegeeinrichtungen, externe Dienstleister, MDK	
<i>ID:</i> 209	-In der Konsolidierung mit Fokusgruppe 01 gestrichen-
<i>ID:</i> 210	<i>Titel:</i> Richtlinien für Benutzerschnittstellen
<i>Beschreibung:</i> Es muss möglich sein, AAL-Anwendungsfälle über Richtlinien für Benutzerschnittstellen konsistent in die AAL-Umgebung zu integrieren.	
<i>Akteure:</i> Handwerker, Hausmeister, MDK	
<i>ID:</i> 211P	<i>Titel:</i> Nutzt bereits vorhandene Strukturen
<i>Beschreibung:</i> Es muss möglich sein, für das UI bereits vorhandene Strukturen zu nutzen. So können beispielsweise bereits eingebaute Funklichtschalter oder vorhandene Tablet-PCs wiederverwendet werden.	
<i>Akteure:</i> Bewohner, formelle Betreuer, informelle Betreuer	

Hardware

<i>ID:</i> 301	<i>Titel:</i> Bedarf
<i>Beschreibung:</i> Die Hardware sollte im Bezug auf Preis, Komplexität und Funktionalität bedarfsorientiert wählbar sein.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Handwerker, AAL-Anbieter	
<i>ID:</i> 302P	<i>Titel:</i> Schaltbare Sensoren & Aktoren
<i>Beschreibung:</i> Es muss möglich sein Sensordaten zu empfangen und Aktoren zu schalten.	
<i>Akteure:</i> Bewohner, Pflegeeinrichtungen	
<i>ID:</i> 303	<i>Titel:</i> Kosten
<i>Beschreibung:</i> Es muss günstige Hardware verfügbar sein.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Hardware-Entwickler, Handwerker, AAL-Anbieter, Pflegeeinrichtungen	
<i>ID:</i> 304P	<i>Titel:</i> Erweiterbare Sensoren und Aktuatoren
<i>Beschreibung:</i> Eine AAL-Umgebung muss sich jederzeit um weitere Sensoren und Aktuatoren erweitern lassen.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Hardware Entwickler, Handwerker, AAL-Anbieter, Pflegeeinrichtungen	
<i>ID:</i> 305	<i>Titel:</i> Missbrauchsschutz bei der Datenübertragung
<i>Beschreibung:</i> Die Datenübertragung von und zu der Hardware muss vor Missbrauch durch Dritte hinreichend geschützt sein.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Besucher, Hausmeister, AAL-Anbieter	
<i>ID:</i> 306	<i>Titel:</i> Robustheit
<i>Beschreibung:</i> Die verfügbare Hardware muss hinreichend robust sein.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Hausmeister, AAL-Anbieter	
<i>ID:</i> 307	<i>Titel:</i> Qualitätsstandard
<i>Beschreibung:</i> Die Hardware sollte entsprechenden Qualitätsstandards unterliegen und entsprechend zertifiziert sein.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Hardware-Entwickler, Handwerker, Hausmeister, AAL-Anbieter	
<i>ID:</i> 308	<i>Titel:</i> Ästhetik
<i>Beschreibung:</i> Die verfügbare Hardware sollte allen ästhetischen Ansprüchen genügen.	
<i>Akteure:</i> Bewohner, AAL-Anbieter	
<i>ID:</i> 309	<i>Titel:</i> Unauffällige Integration
<i>Beschreibung:</i> Die Hardware sollte in der AAL-Umgebung unauffällig sein.	
<i>Akteure:</i> Bewohner, informelle Betreuer, Besucher	
<i>ID:</i> 310	<i>Titel:</i> Einfache Installation
<i>Beschreibung:</i> Die Sensoren und Aktoren sollen sich einfach in der AAL-Umgebung installieren lassen.	
<i>Akteure:</i> Bewohner, Software Entwickler, Hardware Entwickler, Handwerker	

<i>ID:</i> 311	<i>Titel:</i> Schnittstellenstandards
<i>Beschreibung:</i> Die Hardware sollte gängige Schnittstellenstandards unterstützen.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Hardware-Entwickler, Handwerker, Hausmeister, AAL-Anbieter, Pflegeeinrichtungen, externe Dienstleister	
<i>ID:</i> 312	<i>Titel:</i> Zustandsüberwachung
<i>Beschreibung:</i> Die Hardware sollte die AAL-Plattform über ihren eigenen Zustand (z.B. Batteriekapazität) informieren können, um die Funktionalität zu garantieren.	
<i>Akteure:</i> informelle Betreuer, Hardware-Entwickler, Hausmeister, Pflegeeinrichtungen	
<i>ID:</i> 313	-In der Konsolidierung mit Fokusgruppe 01 gestrichen-
<i>ID:</i> 314	<i>Titel:</i> Breite Auswahl
<i>Beschreibung:</i> Es sollte eine breite Auswahl an Hardware vorhanden sein. Dies gilt sowohl für die Anzahl der Hardwareanbieter sowie für die Funktionen der von ihnen angeboten Hardware.	
<i>Akteure:</i> Hardware-Entwickler, Pflegeeinrichtungen, MDK	
<i>ID:</i> 315	<i>Titel:</i> Wartung
<i>Beschreibung:</i> Die Hardware sollte wartungsarm sein.	
<i>Akteure:</i> Hardware-Entwickler, Hausmeister, Pflegeeinrichtungen	
<i>ID:</i> 316	<i>Titel:</i> Guter Support
<i>Beschreibung:</i> Die Hersteller der Hardware sollten einen guten Support liefern.	
<i>Akteure:</i> Hardware-Entwickler, Hausmeister	
<i>ID:</i> 317	<i>Titel:</i> Gute Wartbarkeit
<i>Beschreibung:</i> Die Hardware sollte sich gut und einfach warten lassen, wenn sie in die AAL-Umgebung integriert ist.	
<i>Akteure:</i> Hardware-Entwickler, Handwerker	
<i>ID:</i> 318	<i>Titel:</i> Verständlichkeit
<i>Beschreibung:</i> Die Hardware sollte gegenüber den Endanwendern verständlich agieren. Sie sollten die Arbeitsweise nachvollziehen können.	
<i>Akteure:</i> Bewohner, Hardware-Entwickler, MDK	

Support

<i>ID:</i> 401	<i>Titel:</i> Günstiger Support
<i>Beschreibung:</i> Für die AAL-Plattform sollte ein günstiger Support verfügbar sein.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Handwerker, Hausmeister, AAL-Anbieter	
<i>ID:</i> 402	<i>Titel:</i> Guter Support
<i>Beschreibung:</i> Für die AAL-Plattform sollte ein guter Support verfügbar sein.	
<i>Akteure:</i> Bewohner, informelle Betreuer, Hausmeister, AAL-Anbieter	

<i>ID:</i> 403	<i>Titel:</i> Langzeitsupport
<i>Beschreibung:</i> Für die AAL-Plattform sollte ein langfristiger Support sichergestellt sein.	
<i>Akteure:</i> Bewohner, Software-Entwickler, Hausmeister, AAL-Anbieter	
<i>ID:</i> 404	<i>Titel:</i> Weiterentwicklung
<i>Beschreibung:</i> Die stetige Weiterentwicklung der AAL-Plattform sollte erfolgen, um auf Fehler zu reagieren oder neue Funktionalitäten einzubauen.	
<i>Akteure:</i> Bewohner, Software Entwickler, Hardware Entwickler, AAL-Anbieter	
<i>ID:</i> 405P	<i>Titel:</i> Gute Dokumentation
<i>Beschreibung:</i> Die AAL-Plattform sollte über eine gute Dokumentation für die verschiedenen Akteurguppen verfügen.	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler, Handwerker, Hausmeister, AAL-Anbieter	
<i>ID:</i> 406	<i>Titel:</i> Referenzfälle
<i>Beschreibung:</i> Für ein besseres Verständnis bei der Eigenentwicklung von AAL-Anwendungsfällen für die AAL-Plattform sollten Referenzanwendungsfälle vorhanden sein.	
<i>Akteure:</i> Software-Entwickler	
<i>ID:</i> 407	<i>Titel:</i> Entwicklungsrichtlinien
<i>Beschreibung:</i> Für die Entwicklung von eigenen Komponenten und AAL-Anwendungsfällen sowie deren Interoperabilität zur AAL-Plattform sind Entwicklungsrichtlinien wichtig.	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler	
<i>ID:</i> 408	<i>Titel:</i> Breite Community
<i>Beschreibung:</i> Eine breite Community sollte vorhanden sein, da sie mit ihren Entwicklungen und Ideen zur Flexibilität und Robustheit der AAL-Plattform beiträgt.	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler, AAL-Anbieter	
<i>ID:</i> 409P	<i>Titel:</i> Passendes Lizenzmodell
<i>Beschreibung:</i> Das Lizenzmodell des AAL-Plattform muss die Vermarktungs- und Nutzungsstrukturen unterstützen. Meist wird die Middleware mit Hardware und weiteren AAL-Services ausgeliefert.	
<i>Akteure:</i> AAL-Anbieter, Pflegeeinrichtungen	

Werkzeuge

<i>ID:</i> 501P	<i>Titel:</i> Sensor- und Aktor-Management
<i>Beschreibung:</i> Die AAL-Plattform muss Werkzeuge bereitstellen, die eine Verwaltung der angeschlossenen Hardware vereinfachen.	
<i>Akteure:</i> Bewohner, Hardware-Entwickler	
<i>ID:</i> 502	<i>Titel:</i> Grundfunktionen der Hausautomation
<i>Beschreibung:</i> Mit Werkzeugunterstützung müssen sich Grundfunktionen der Hausautomatisierung erstellen lassen. Sensoren und Aktoren müssen sich auf einfache Weise miteinander verknüpfen lassen, um einfache Komfortfunktionen zu realisieren.	
<i>Akteure:</i> Bewohner, Hardware-Entwickler	

<i>ID:</i> 503	<i>Titel:</i> Gute Konfigurierbarkeit
<i>Beschreibung:</i> Es muss Werkzeuge geben, die eine Konfiguration der AAL-Plattform und ihre Komponenten sowie AAL-Anwendungsfälle zulassen.	
<i>Akteure:</i> Bewohner, Pflegeeinrichtungen	
<i>ID:</i> 504	<i>Titel:</i> Einfache Installation und Inbetriebnahme neuer Hardware
<i>Beschreibung:</i> Es soll eine Werkzeugunterstützung für die Installation und Inbetriebnahme neuer Hardware geben.	
<i>Akteure:</i> Bewohner, Handwerker, Hausmeister	
<i>ID:</i> 505	<i>Titel:</i> Zugriffsbeschränkung
<i>Beschreibung:</i> Es solle eine Zugriffsverwaltung in der AAL-Plattform durch Werkzeuge realisiert sein.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Besucher, Hausmeister, AAL-Anbieter, MDK	
<i>ID:</i> 506	<i>Titel:</i> Robuste Werkzeuge
<i>Beschreibung:</i> Alle von der AAL-Plattform bereitgestellten Werkzeuge für die verschiedenen Akteurguppen sollen hinreichend robust sein.	
<i>Akteure:</i> Bewohner, Hausmeister, AAL-Anbieter	
<i>ID:</i> 507P	<i>Titel:</i> Erweiterbarkeit
<i>Beschreibung:</i> Die AAL-Plattform muss modular aufgebaut sein, damit sie bei Bedarf leicht zu erweitern ist. Die Erweiterungen beziehen sich hierbei hauptsächlich auf AAL-Anwendungsfälle mit ihren Hard- und Software-Komponenten. Aber auch anderen Erweiterungen mit neuen Komponenten in der Middleware oder über Schnittstellen sind denkbar.	
<i>Akteure:</i> Bewohner, Hardware-Entwickler, Handwerker, AAL-Anbieter	
<i>ID:</i> 508P	<i>Titel:</i> Mächtige Entwicklungssprache
<i>Beschreibung:</i> Die AAL-Plattform muss die Möglichkeit bieten, sich in einer mächtigen Programmiersprache auf neue Funktionalitäten erweitern zu lassen.	
<i>Akteure:</i> Bewohner, formelle Betreuer, Software-Entwickler, externe Dienstleister	
<i>ID:</i> 509	<i>Titel:</i> Schnelle Entwicklung von AAL-Anwendungsfällen
<i>Beschreibung:</i> Die Entwicklung von neuen AAL-Anwendungsfällen sollte durch passende Werkzeugunterstützung schnell vonstatten gehen.	
<i>Akteure:</i> Bewohner, Software Entwickler	
<i>ID:</i> 510	<i>Titel:</i> Einfache und schnelle Installation der AAL-Plattform
<i>Beschreibung:</i> Es sollten Installationswerkzeuge bereitstehen, die eine schnelle und einfache Installation erlauben.	
<i>Akteure:</i> Bewohner, Handwerker	
<i>ID:</i> 511	<i>Titel:</i> Fernwartung
<i>Beschreibung:</i> Die AAL-Plattform sollte die Möglichkeit zur Fernwartung besitzen.	
<i>Akteure:</i> Hausmeister, AAL-Anbieter, Pflegeeinrichtungen	

<i>ID:</i> 512	<i>Titel:</i> Entwicklungsunterstützung
<i>Beschreibung:</i> Den Entwicklern sollen Werkzeuge zur Verfügung stehen, die sie bei der Entwicklung von neuen Komponenten (Software und Hardware) unterstützen.	
<i>Akteure:</i> Software-Entwickler	
<i>ID:</i> 513	<i>Titel:</i> Diagnosetools
<i>Beschreibung:</i> Die AAL-Plattform soll Werkzeuge zu ihrer Diagnose bereitstellen. Diese dienen zur Funktionsüberwachung der Plattform sowie allen ihrer Komponenten (Software und Hardware).	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler, Handwerker, Hausmeister, AAL-Anbieter, MDK	
<i>ID:</i> 514	<i>Titel:</i> Schnittstellen
<i>Beschreibung:</i> Die AAL-Plattform sollte Werkzeuge bieten, welche die Schnittstellennutzung zur Anbindung anderer Dienste außerhalb der AAL-Umgebung unterstützen.	
<i>Akteure:</i> Software Entwickler, Pflegeeinrichtungen, externe Dienstleister	
<i>ID:</i> 515	<i>Titel:</i> Distributionsdienste
<i>Beschreibung:</i> Zur anwenderfreundlichen Distribution von AAL-Anwendungsfällen sollte die AAL-Plattform eine Anbindung an geeignete Dienste haben oder selber bereitstellen.	
<i>Akteure:</i> Software-Entwickler, Hardware-Entwickler, AAL-Anbieter	
<i>ID:</i> 516	<i>Titel:</i> Skalierbare Komplexität
<i>Beschreibung:</i> Die AAL-Plattform sollte im hohen Maß skalierbar sein, um verschiedenen Anforderungen zu genügen.	
<i>Akteure:</i> Handwerker, AAL-Anbieter	

Diese 63 Anforderungen definieren die Basis zur Analyse der AAL-Plattformkandidaten im folgenden Kapitel und bilden den Startpunkt für die Sicherstellung der Flexibilität einer AAL-Umgebung.

3.3 Flexibilität von AAL-Plattformkandidaten

Dieses Kapitel untersucht die Flexibilität von potenziellen AAL-Plattformkandidaten. Dies geschieht mit Hilfe der erarbeiteten Akteure und ihrer Anforderungen. In Frage kommende Kandidaten wurden aus state-of-the-art Middlewares ausgewählt. So kann zum einen erforscht werden, welche wichtigen Eigenschaften für AAL-Plattformen bereits in anderen Middlewares vorhanden sind, zum anderen können die Anforderungen an eine flexible AAL-Umgebung, welche bisher nicht bedient werden, explizit herausgearbeitet werden.

Eine AAL-Plattform ist der technische Mittelpunkt einer AAL-Umgebung (vgl. Abbildung 1.1). An dieser Plattform werden Sensoren angeschlossen, um Informationen aus der AAL-Umgebung aufzunehmen und Aktoren, um auf die AAL-Umgebung einzuwirken. Diese Plattform dient im Sinne einer Middleware weiterhin zur Abstraktion der Hardware zu Softwarediensten. Die AAL-Umgebung kann somit auf Basis der Kontextinformationen interpretiert werden und Reaktionen auf Ereignisse in der AAL-Umgebung auslösen. Somit ist gerade die AAL-Plattform ein virulentes Fundament für Flexibilität in AAL.

3.3.1 Sichtung der AAL-Plattformkandidaten

Dieses Kapitel zerfällt in zwei Teile, deren Methodik in Kapitel 2.1.2 beschrieben ist. Zuerst wurde eine Sammlung potenzieller AAL-Plattformkandidaten angelegt. Diese unterzog man anschließend einer Prüfung gegen die Pflichtenforderungen für eine flexible AAL-Umgebung. Im zweiten Teil wurden die zehn Kandidaten, welche die Pflichtenforderungen erfüllen, ausführlich weiter untersucht.

Es ergaben sich aus dem ersten Teil der Analyse die folgenden 43 Plattformkandidaten:

Industrie (Anzahl 31): Busch-Jäger, Connected Living, Control4, Contronics, ELV Hausautomation, ETS, EZControl, Gira KNX-System, Hager Domovea, HANS, HCAN Hausautomatisierung, HomeMatic, IO-Homecontrol, IP-Symcon, Jung KNX-Software, LocSens, Logic Machine 2, Loxone, myHomeControl, NovaHome, OpenRemote (Open Source/ Industrie)², opnode, Power2Home Insteon, Prosyst mPRM, RWE SmartHome, SmartLiving GmbH, Telefunken, Tunstall ADLife, Vera, ViciOne, Wellaware System

Open Source Gemeinschaft (Anzahl 7): FHEM, HomeAutomationProject, Hydra aka LinkSMART (Forschung/ Open Source)³, i2home aka openURC (Forschung/ Open Source), Open Source Automation, openHAB, universAAL (Forschung/ Open Source)

Forschung (Anzahl 5): Amigo, Genesys, MPower, OASIS, PERSONA

Die Prüfung der Plattformkandidaten gegen die Pflichtenforderungen ergab 10 taugliche Plattformen zur weiteren Analyse. Die genauen Ausscheidungskriterien sind in den Tabellen 3.1, 3.2 und 3.3 niedergeschrieben. Weiterhin ist der erste Eindruck der jeweiligen Plattform notiert. Dieser entstand bei der Prüfung der Pflichtenforderungen durch die Fokusgruppe 02 und bildet ihre subjektive Meinung ab.

Die folgenden 10 Plattformenkandidaten erfüllten die Pflichtenforderungen und werden im zweiten Teil dieses Kapitels bezüglich ihrer Flexibilität weiter untersucht.

Contronics

Die Firma Contronics Automationssysteme hat verschiedene Produkte zur Hausautomatisierung entwickelt [Cont13]. Der Kern ihrer Entwicklungen ist eine hauseigene Plattform zur zentralen Ansteuerung von Hardware.

Die von Contronics entwickelte Plattform erlaubt es, einfache Anwendungsfälle umzusetzen. Die Programmierung erfolgt über die eigene CL-Skriptsprache und ist komplett in Deutsch. Von sich aus bringt die Plattform keine Anwendungsfälle mit. Die Middleware ist über die eigene Softwareoberfläche konfigurierbar und steuerbar. Es lassen sich aber nur Geräte vom Typ HomeMatic, FS20 und slowRF ansprechen und verarbeiten. Insgesamt ist die Middleware über verschiedene Softwarekomponenten einfach zu bedienen. Leider war es im Rahmen dieser Auswertung nicht möglich herauszufinden, ob und wie die Plattform weiterentwickelt wird. Wichtig ist auch hervorzuheben, dass Hardware von der Plattform direkt angesteuert wird und nicht

²OpenRemote vertreibt Teile seiner Middleware unter der Affero GNU Public License.

³Projekte, die als (Forschung/ Open Source) markiert sind, starteten als Forschungsprojekt und sind in Open Source übergegangen.

Plattform	Widerspricht	Ersteindruck	Kommentar
Busch-Jäger	507P	sehr gut	keine Middleware nur Server
Connected Living	405P	sehr gut	kaum öffentliche Informationen
Control4	508P	sehr gut	luxusorientiert
ELV Hausautomatisierung	201P, 507P, 508P	ausreichend	
ETS	507P	gut	
EZControl	201P, 501P	befriedigend	
Gira KNX-System	507P	sehr gut	keine Middleware nur Server, nur KNX
Hager Domovea	405P, 508P	sehr gut	keine Middlewaredokumentation
HANS	405P, 501P, 507P, 508P	befriedigend	eher Hausnotrufsystem
HCAN	201P, 211P, 405P, 409P, 507P, 508P	gut	
HomeMatic	507P, 508P	gut	
IO-Homecontrol	211P, 409P, 501P, 507P, 508P	befriedigend	
Jung	507P, 508P	sehr gut	KNX-Anbieter
LocSens	409P, 507P	ausreichend	fertiges, nicht erweiterbares Produkt
myHomeControl	507P, 508P	gut	
NovaHome	405P	befriedigend	nicht erhältlich
opnode	405P	mangelhaft	letztes Update 2009
Power2Home	507P, 508P	gut	
Prosyst MPRM	501P	befriedigend	
RWE SmartHome	507P	befriedigend	
SmartLiving	102P, 302P, 501P	befriedigend	Set-Top Box
telefunken	507P, 508P,	gut	
Tunstall	405P, 507P	befriedigend	
ViciOne	405P	sehr gut	erfüllt alles laut Webseite, kein Kontakt herstellbar
Wellaware System	405P	befriedigend	eher Aktivitätserkennung

Tabelle 3.1: Verstöße der Plattformkandidaten der Industrie gegen die Pflichtenforderungen

Plattform	Widerspricht	Ersteindruck	Kommentar
Home Automation Project	211P, 409P, 507P, 508P	ausreichend	
Hydra	501P	sehr gut	bald industriell erhältlich
i2home	501P	gut	

Tabelle 3.2: Verstöße der Plattformkandidaten der Open Source-Gemeinschaft gegen die Pflichtanforderungen

Plattform	Widerspricht	Ersteindruck	Kommentar
Amigo	—	gut	fortgeführt in universAAL
Genesys	405P	ausreichend	
MPower	405P	ausreichend	
OASIS	501P	befriedigend	
PERSONA	—	befriedigend	fortgeführt in universAAL

Tabelle 3.3: Verstöße der Plattformkandidaten der Forschung gegen die Pflichtanforderungen. Amigo und PERSONA wurden nicht auf Verstöße gegen die Pflichtanforderungen untersucht, da sie in das Projekt universAAL eingeflossen sind.

beliebig erweiterbar ist. Weiterhin bietet diese Middleware keine Schnittstellen nach außen.

Insgesamt eignet sich diese Plattform nicht für ein flexibles Management von AAL-Plattformen.

FHEM

FHEM (Freundliche Hausautomatisierung und Energie-Messung) ist ein Open Source-Projekt zur Hausautomatisierung [Kön13]. Das System wird von einer großen Gemeinschaft getragen und weiterentwickelt. Die Middleware ist in Perl geschrieben und findet hauptsächlich im deutschsprachigen Raum Verbreitung. Die Middleware gestaltet sich hierbei sehr ressourcenschonend und plattformübergreifend (z.B. Raspberry Pi, Router).

Die gesamte Logik von FHEM ist in Perl implementiert und lässt sich bei Bedarf einfach erweitern, um neue Anwendungsfälle zu integrieren. Einfache Anwendungsfälle können auch über die Weboberfläche von FHEM realisiert werden, in einer an Perl angelehnten Skriptsprache. Durch die große Nutzergemeinschaft existieren bereits viele Anwendungsfälle. In den Untersuchungen hat sich das System robust gezeigt und war über die Perl-Konsole oder die Weboberfläche gut zu konfigurieren und zu steuern. Leider ist die Weboberfläche technisch veraltet und gestaltet sich sehr einfach. Die Open Source-Gemeinschaft hat neben den Anwendungsfällen auch eine Vielzahl an Perl Modulen geschrieben, die das Ansteuern verschiedener Sensoren und Aktoren übernehmen. Hierzu wurden auch Entwicklungsrichtlinien herausgegeben, die allerdings nicht immer eingehalten wurden. Dies hat zur Folge, dass nicht alle Steuer- und Konfigurationsbefehle modulübergreifend funktionieren. Eine tiefgreifende Konfiguration von FHEM erfordert viel Erfahrung. Zum Erweitern stellt FHEM eine Vielzahl an Schnittstellen (z.B. HTTP/JSON, Telnet) zur Verfügung.

FHEM gestaltet sich als mittelmäßig flexibel und robust (Test über 6 Monate) wird aber bei komplexen Anwendungsfällen schnell unübersichtlich und kompliziert. Eine Verwendung als AAL-Plattformen ist darum nur eingeschränkt möglich.

iP-Symcon

Die Firma iP-Symcon GmbH entwickelt eine Hausautomatisierungsmiddleware, welche auf kleinere Unternehmen und den privaten Anwender zielt [Symc14]. Hierbei ist hervorzuheben, dass sie ausschließlich auf Microsoft Windows-Betriebssystemen arbeitet. Die iP-Symcon Plattform wurde im Auswertungsprozess über einen Zeitraum von einem Monat betrieben.

Die Logik der iP-Symcon-Middleware wurde durch die Skripte in der Sprache PHP realisiert. Es wird ein eigener Editor mit einer guten Auto-Vervollständigung bereitgestellt. Jeder Funktionalität (egal ob virtuell, von einem Sensor oder Aktor) wird eine Variable zugeordnet, die in der Konfiguration verwendet werden kann. Weiterhin gibt es Objekte (z.B. Geräte, Räume), welche sich in logische Strukturen einteilen lassen und automatisch in der grafischen Steuerungsoberfläche eingesetzt werden können. Diese ist als Weboberfläche umgesetzt und auch in einer nativen Implementierung als App zu erwerben. Die Weboberfläche lässt sich durch einen integrierten Konfigurator weiter individualisieren und auch mit externen Inhalten per PHP verbinden. Es existiert ein Profildienst, der es erlaubt die Steuerungsoberfläche individualisiert für mehrere Nutzer bereitzustellen. Die Flexibilität dieses Systems zeigt sich in der Sensor- und Aktoranbindung. So werden die meisten gängigen Protokolle und Hardwarehersteller unterstützt. Die Integration von neuen Hardwarekomponenten wird durch entsprechende Assistenten angeleitet. Um auf dem aktuellen Stand der Software zu sein, bietet iP-Symcon ein Abo-Modell an. Die Plattform selber läuft sehr robust (Test über einen Monat) und hat eine große Community. Es ist einfach zu installieren und konfigurieren. Es bietet Schnittstellen über SOAP, PHP und Delphi an.

Diese Middleware zeichnet sich im Besonderen durch eine hohe Konfigurierbarkeit und Flexibilität aus. Bis auf ihre Beschränkung auf Windows-Betriebssysteme als Laufzeitumgebung ist sie zum Einsatz in flexiblen AAL-Umgebungen sehr geeignet.

Logic Machine

Logic Machine ist von der Firma embedded systems und aktuell in der dritten Version verfügbar [esUG14]. Sie ist auf das Anwendungsgebiet der Hausautomatisierung ausgerichtet und bietet neben dem Ansteuern von Sensoren und Aktoren verschiedener Hersteller eine Vielzahl an Einbindungsmöglichkeiten für Unterhaltungselektronik. Die Logic Maschine-Hardware konnte leider nicht getestet werden. Die Bewertung bezieht sich auf die Software, Dokumentation und andere frei verfügbare Quellen.

Anwendungsfälle lassen sich mit der Scriptsprache Lua für die Logic Machine schreiben. Es existiert hierfür ein Konfigurator mit Editor, der erlaubt Anweisungen per Mausklick einzufügen. Leider bietet die Middleware nach außen keine Programmierschnittstellen. Die Visualisierung und Interaktion mit dem Nutzer erfolgt über eine Weboberfläche oder nativ auf mobilen Geräten. Beide Interfaces lassen sich komplett personalisieren. Die gesamte Konfiguration der Hardware läuft über die Webschnittstelle. Hierbei werden die gängigen Protokolle der Hausautomatisierung unterstützt (z.B. KNX/EIB, EnOcean, ModBus).

Die Nutzung der Logic Machine im AAL-Kontext wird als eher schwierig eingeschätzt, da zwar ein ansprechender Konfigurationseditor und -prozess vorhanden ist, aber durch die fehlenden Schnittstellen eine Erweiterbarkeit um neue und komplexere Anwendungsfälle schwer fällt. Auch die Anbindung externer Ressourcen gestaltet sich dadurch schwierig.

Loxone

Loxone Electronics GmbH vertreibt seit 2008 die Plattform Loxone [Loxo14]. Der Kern ist ein kleiner Server, dessen Middleware diverse Hardware ansteuern kann. Das Produkt wird derzeit von einer großen Gemeinschaft im Bereich der Hausautomatisierung eingesetzt und zeigt sich entsprechend robust. Die Hardware des Servers konnte leider nicht getestet werden.

Zur Konfiguration des Loxone-Systems gibt es verschiedene Softwareprogramme sowie die Möglichkeit, die Konfiguration über die Weboberfläche vorzunehmen. Die Möglichkeiten hierbei sind sehr vielseitig und erfordern entsprechendes Wissen. Durch Erweiterungen kann der Server mit vielen Protokollen umgehen (z.B. KNX/EIB, 1Wire, EnOcean). Die Steuerung der Hardware geschieht wahlweise über die App oder die Weboberfläche. Die Konfigurationssoftware erlaubt es, einfache Anwendungsfälle in Form von Skripten direkt in die Middleware zu laden. Komplexere Anwendungsfälle können über Schnittstellen integriert werden (z.B. XML und JSON über HTTP). Das System ist sehr übersichtlich dokumentiert, wirkt robust und wird stetig weiterentwickelt.

Benötigt man eine AAL-Plattform mit einem vorab definierten beschränkten Leistungsumfang, in dessen Rahmen man flexibel agieren kann, eignet sich Loxone sehr gut.

Open Remote

Diese Plattform teilt sich in eine Open Source-Komponente sowie zwei Erweiterungen, die kostenpflichtig vertrieben werden [Open14b]. Zum Open Source-Teil gehören die Middleware sowie der Open Remote Designer und ein Konfigurator zum Einrichten der Middleware. Kostenpflichtig kann Open Remote Professional erworben werden. Dieser Konfigurator stellt eine Vielzahl an weiteren Vereinfachungen und Funktionen für die Konfiguration der Middleware sowie die Möglichkeit, die einzelnen Middlewareinstanzen über die Cloud zu managen.

Die Weboberfläche und die nativen Apps von Open Remote sind klar und benutzerfreundlich strukturiert. Sie können vollständig personalisiert werden. Die Hardware der gängigen Hersteller kann einfach integriert werden und entsprechende Sensoren und Aktoren werden über den Designer mit Hilfe von Makros eingebunden. Einfache Anwendungsfälle lassen sich per Drag&Drop erstellen und mit Makros kombinieren. Es lässt sich jedoch keine eigene Logik über Makros definieren und in den Anwendungsfall einbauen. Um komplexere Anwendungsfälle zu erstellen, müssen diese als Erweiterung der Java Middleware implementiert werden. Auf diesem Weg können auch neue Protokolle sowie Hardware eingebunden und das System beliebig erweitert werden. Hierzu ist eine entsprechende Anleitung vorhanden. Insgesamt wirkt die Dokumentation des Projektes vollständig und gut aufbereitet, weiterführende Fragen werden im Forum zeitnah beantwortet. Die Steuerung der Hardware erfolgt ohne Abstraktion.

Open Remote eignet sich wegen der fehlenden Abstraktion nur mäßig für den Einsatz als flexible AAL-Plattform.

openHAB

Die openHAB Plattform stammt aus dem Jahr 2011 und setzt mit Technologien wie Java und OSGi komplett auf Open Source [open14a]. Das System besteht aus zwei Teilen: der eigentlichen Middleware und einem Designer, der die Konfiguration derselben unterstützt.

Die Realisierung von einfachen Anwendungsfällen ist in openHAB mit Hilfe von Xtend Skripten möglich. Komplexere Anwendungsfälle dagegen werden als Java OSGi Bundle implementiert. Der openHAB-Designer hilft beim Erstellen der Xtend Skripte und kann auch eine Vielzahl von Konfigurationen an der Middleware vornehmen. Steuern lässt sich openHAB durch eine Weboberfläche und native Apps. Hierbei sind eine Personalisierung und die Nutzung von Profilen für mehrere Nutzer möglich. openHAB unterstützt derzeit alle wichtigen Hardware-Protokolle, viele davon aber nur teilweise. Den regelmäßigen Updates ist zu entnehmen, dass die Open Source-Gemeinschaft die Plattform permanent weiterentwickelt. Die Dokumentation ist gut und es sind zahlreiche Schnittstellen vorhanden.

openHAB ist als AAL-Plattform sehr gut geeignet, muss aber um AAL spezifische Aspekte und Anwendungsfälle erweitert werden.

Open Source-Automation

Dieses Open Source Projekt will eine Hausautomatisierungslösung schaffen, welches als Kernziel auf Erweiterbarkeit setzt [RuWo14]. Deshalb wird während der Weiterentwicklung durch die Open Source Community auf eine möglichst einfache API geachtet.

Die Middleware ist in C# auf Basis des .NET Frameworks für Windows geschrieben. Diese Middleware (Server) kann mit Hilfe von Bundlen erweitert werden, um mit Hardware zu kommunizieren. Von Haus aus werden nur sehr wenige Protokolle durch Bundles unterstützt. Die Wichtigsten sind Insteon, ZWave, X10 und ZigBee. Einfache Anwendungsfälle können aber mit Hilfe der Clientkomponente und dazugehörigen Skripten erstellt werden. Weiterhin kann mit dem Client die Konfiguration der Hardware und der Nutzer vorgenommen werden. Die Dokumentation empfiehlt mit dem Konfigurator im ersten Schritt, die gesamte Umgebung zu modellieren. Hierbei zeigt sich der nur windowstaugliche GUI Client als sehr komplex und nicht intuitiv. Konfigurationen können komplett klickbasiert zusammengestellt werden, bevor sie auf der hardwareunabhängigen Middleware ausgeführt werden. Das System wird regelmäßig weiterentwickelt und besitzt neben einem Forum eine gute Dokumentation. Dieses bietet Informationen zur Installation, Konfiguration, Implementierungsrichtlinien, Referenzanwendungsfälle und eine API-Beschreibung in einem Wiki. Die Erstinstallation des Servers wird von einem Installationsassistenten unterstützt. Allerdings funktionierte er trotzdem auf keinem der Testrechner, so dass manuell einige Korrekturen des SQL-Dienstes notwendig waren. Es existiert keine Zugriffsbeschränkung von Seiten der Middleware gegenüber der Clients.

Das System setzt auf die Verwendung durch Experten und wird für die Verwendung im Bereich von AAL nur mäßig empfohlen.

universAAL

Das Projekt universAAL ist von der EU im siebenten Rahmenprogramm gefördert worden [univ14]. Es ist ein IP-Forschungsprojekt mit 22 Partnern und dem Ziel, eine AAL-Referenzarchitektur sowie Implementierung für AAL-Plattformen zu schaffen.

Die universAAL-Plattform steht seit Anfang 2014 als Open Source (MIT und Apache Lizenz) in Version 3.0.0 bereit. Die Middleware ist durch eine ontologiebasierte Abstraktion hardwareunabhängig. Programmiert wurde sie in Java auf Basis von OSGi. Bisher werden relativ wenige Hardwareprotokolle unterstützt (FS20, KNX, Bluetooth, Zigbee). Andererseits existieren viele Softwarewerkzeuge, welche die Programmierer nicht nur während des Einrichtens, sondern auch während der Weiterentwicklung der Middleware unterstützen. Diese Werkzeuge sind an den speziellen Kontext von AAL ausgerichtet. So existiert für universAAL ein eigener Internetdienst zur Distribution von Hard- und Software. Weiterhin ist die Konfiguration und Personalisierung der Benutzerschnittstelle durch einen komplett dynamischen Aufbau derselben möglich. Aktuell existieren native UIs für mobile Endgeräte und eine Weboberfläche. universAAL ist als verteilte Middleware entwickelt, die während des Testlaufs über drei Monate stabil auf mehreren Knoten funktionierte. Die Dokumentation und die Weiterentwicklung wurden nach der Projektlaufzeit von der AALOA⁴ übernommen und werden dort neben der Verwendung in weiteren Projekten⁵ vorangetrieben. Die Dokumentation stellt sich als ausführlich und zielführend beschrieben dar. Auch Beispiele zur Implementierung weiterer Protokolle und Anwendungsfälle werden gegeben. Aktuell existieren zwölf Anwendungsfälle, die AAL-spezifisch sind. Ihre Einrichtung in der Plattform übernimmt ein Konfigurator, der auch Fernwartungsszenarien unterstützt. Als Schnittstellen gibt es neben einem generischen Gateway die Möglichkeiten, eigene Anbindungen über Java bzw. OSGi Bundle zu schaffen. Die Installation und Konfiguration des Systems gestaltet sich einfach und es läuft in seinen Kernkomponenten robust.

UniversAAL wurde mit dem Ziel entworfen, eine AAL-Plattform zu stellen. Seine Stärke liegt in der hohen Flexibilität und Konfigurierbarkeit. Allerdings ist die Plattform, abgesehen von den Kernkomponenten, teilweise unausgereift. Insgesamt ist der Gebrauch von universAAL als AAL-Plattform sehr zu empfehlen.

Vera

Das System Vera ist ein Smart Home-Server von der Firma Mi Casa Verde [MiCa14]. Ziel ist es, eine kostengünstige und einfache Hausautomatisierung anzubieten. Hierbei setzt Vera auf MiOS (ein embedded Betriebssystem), welches unterschiedliche Protokolle unterstützt und einen nativen Internetdienst zu der Erweiterung durch Module bietet. Die Hardware des Servers konnte leider nicht getestet werden.

Derzeit sind an die 100 Module zum Download über den Internetdienst verfügbar (z.B. Nutzerinteraktion für Blinde, Support für verschiedene Endgeräte). Es sind anteilig wenige AAL-Anwendungsfälle vorhanden. Über diese Module lässt sich aber nicht die Logik der Plattform erweitern. Hierzu müssen eigene Skripte mit der Luup-Engine (Lua-UPnP) geschrieben werden. Diese Engine wurde von Mi Casa Verde entwickelt, um mittels Lua UPnP Geräte zu steuern. Das System unterstützt einige

⁴AAL Open Association: <http://aaloa.org/> (Zugriff am 21.11.2014)

⁵z.B. REAAL: <http://www.cip-reaal.eu/home/> und openAAL: <http://www.openAAL.de> (Zugriff am 21.11.2014)

Anbieter und Protokolle (z.B. ZWave, Zigbee), schließt aber andere wichtige Standards aus (z.B. KNX/EIB, EnOcean). Das System wird ständig weiterentwickelt und kann über die Vera-Konfigurationskonsole (auch Fernwartung) middlewareseitig konfiguriert werden. Hierüber ist auch die nicht intuitiv erscheinende grafische Oberfläche für die verschiedenen Endgeräte zu personalisieren. Die Erweiterung der Middleware über Schnittstellen ist nur über das Einpflegen von eigenen Lua-Skripten über den Internetdienst möglich und somit auf die Mächtigkeit von Luup beschränkt.

Durch die Beschränkungen bei den Protokollen und Schnittstellen wird von einer Nutzung als AAL-Plattform eher abgeraten.

3.3.2 Auswertung der Plattformkandidaten gegen die Anforderungskategorien

Dieses Kapitel analysiert die zehn Plattformkandidaten, welche die Pflichtanforderungen erfüllt haben, gegen alle 63 Anforderungen an eine flexible AAL-Umgebung. Es werden zum einen die verschiedenen von den Kandidaten nicht adressierten Aspekte zum flexiblen Management dargestellt. Zum anderen werden aus den Plattformkandidaten Konzepte und Werkzeuge herausgearbeitet, welche die Flexibilität in besonderer Weise unterstützen. Hierzu werden plattformübergreifend die Ergebnisse anhand der fünf verschiedenen Anwendungsfallkategorien diskutiert. Die Pflichtanforderungen (102P, 201P, 211P, 302P, 304P, 405P, 409P, 501P, 507P, 508P) werden von allen aufgeführten Plattformen erfüllt und an dieser Stelle nicht mehr weiter besprochen. Zu jeder der fünf Anforderungskategorien findet sich in den Tabellen 3.4, 3.5, 3.6, 3.7 und 3.8 die exakte Bewertung der Fokusgruppe 02.

Anwendungsfälle sind das zentrale Element einer jeden Plattform zur Erweiterung und Nutzung durch verschiedene Akteure. Sie implementieren neue Funktionalitäten in der Plattform und ermöglichen eine flexible Anpassung der Plattform an die Umgebung und Bedürfnisse der Nutzer. Im Bereitstellen von *hilfreichen Anwendungsfällen* (101) im AAL-Kontext sticht universAAL hervor. Diese Plattform bietet zum einen für den AAL-Gebrauch konzipierte sowie implementierte Anwendungsfälle und zum anderen eine AAL-taugliche und semantische Prozessmodellierung für den flexiblen Umgang mit derselben. Weiterhin sind Softwarewerkzeuge vorhanden, welche die Erstellung und den Umgang mit Anwendungsfällen vereinfachen. Im Bereich der *guten Konfigurierbarkeit* (103) sind zu nennen universAAL, OSA und Loxone, die alle über leistungsstarke und grafische Tools verfügen, welche zum Einstellen von Skripten und Eigenschaften verwendet werden. iP-Symcon bietet ähnliche Werkzeuge in Form von Assistenten an. *Datenschutz* (104) und *Missbrauchsmöglichkeiten* (105) korrelieren bei der Auswertung in den meisten Fällen. Die meisten Systeme bieten ein Verfahren, um den Zugriff auf alle relevanten Komponenten der Plattform zu kontrollieren. Insgesamt ist dieser Bereich aber in allen untersuchten Plattformen gleichermaßen schwach ausgebaut. Die *Robustheit* (106) der Anwendungsfälle ist in allen Systemen gegeben, besonders in den skriptbasierten Anwendungsfällen durch automatische Prüfungen. Die *Transparenz* (107) der Anwendungsfälle ist besonders bei OSA und OpenRemote zu finden, da diese durch eine gute Skriptgenerierung und -visualisierung den Anwender *gewohnheitsorientiert* (108) unterstützen. Loxone bietet die gleiche Unterstützung, zielt jedoch auf den Experten-anwender. Eine *Förderung des sozialen Umgangs* (109) ist bei fast allen Systemen möglich (z.B.

Anforderungen	Contronics	FHEM	iP-Symcon	Logic	Loxone	OSA	openHAB	OpenRemote	universAAL	Vera
101 AAL-taugliche Anwendungsfälle	2	2	2	2	2	2	2	2	1	2
102P Grundkonzepte der Hausautomatisierung	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
103 Gute Konfigurierbarkeit	4	3	2	3	2	2	3	3	1	3
104 Beachtung des Datenschutzes	3	1	1	1	1	1	2	1	3	1
105 Aktive Missbrauchsverhinderung	4	1	1	1	1	3	1	1	3	1
106 Robustheit	2	1	1	0	0	1	3	1	0	0
107 Transparent	3	3	2	3	3	1	2	1	3	0
108 Gewohnheitsorientiert	2	2	1	1	1	1	1	2	1	2
109 Unterstützung des sozialen Umgang	Nein	Nein	Ja	Ja	Ja	Nein	Ja	Nein	Nein	Nein
110 Entlastung der betreuenden Akteure	3	1	1	1	1	2	1	2	2	1
111 Geheimhaltung persönlicher Daten	Nein	Ja	Ja	Ja	Ja	Nein	Ja	Nein	Nein	Ja
112 Offene Schnittstellen	Nein	Ja	Nein	Nein	Nein	Ja	Ja	Nein	Ja	Ja

Tabelle 3.4: Bewertung der Plattformkandidaten gegen die Anforderungen der Kategorie Anwendungsfälle.

Webcam Unterstützung). Direkt als Kommunikationsmedium konzipierte Anwendungsfälle konnten in keiner Plattform identifiziert werden. Alle Systeme können Informationen ausgeben (z.B. Sensor- oder Status-Information) und so *betreuende Organe entlasten* (110). Diese sind meist in einer Weboberfläche einzusehen und in sechs Fällen sogar weitgehend konfigurierbar, um sie an spezielle Bedürfnisse anzupassen. Hierzu stehen verschiedene Visualisierungen zur Verfügung (z.B. zeitliche Verläufe, kumulierte Darstellungen, Graphen). Hierüber können, neben betreuenden Organen, die Informationen bei sechs Plattformen auch anderweitig veröffentlicht werden. Zur flexiblen Gestaltung von Anwendungsfällen bieten alle Open Source-Plattformen *Schnittstellen* (112). Besonders hervorzuheben ist die Plattform Vera, die durch ihren Software-Onlineshop eine Veröffentlichung jeder Erweiterung erzwingt.

Eine möglichst flexible Gestaltung und Verwendung von Anwendungsfällen findet sich in den Open Source-Plattformen. Bis auf universAAL bieten sie alle auch die Möglichkeit, durch skriptbasierte Ansätze schnell zu Ergebnissen zu kommen. Sollen dagegen komplexere Sachverhalte abgebildet werden, erhält man bei ihnen die Möglichkeit, in einer mächtigen Programmiersprache zu arbeiten. Keine der untersuchten Plattformen bietet eine befriedigende Anzahl an einsetzbaren AAL-Anwendungsfällen, erst recht nicht sind diese flexibel einsetzbar. Selbst die AAL-Plattform universAAL bietet nur Referenzanwendungsfälle, die in ihren Kernfunktionen als Referenzimplementierung existieren.

Eine anpassungsfähige **Benutzerschnittstelle** (UI) ist grundlegend für eine flexible AAL-Plattform. Als zentraler Interaktionspunkt für die verschiedenen Akteure mit der Middleware sollten neun Anforderungen erfüllt sein. Hierbei wurden die Anpassbarkeit und die Nutzbarkeit analysiert. Außerdem wurden die digitalen UIs getrennt von den analogen getestet. Eine AAL-Plattform sollte hierbei den Spagat schaffen, zum einen für den älteren Menschen, ohne Schulung intuitiv nutzbar zu sein, und zum anderen dem versierten Nutzer umfangreiche Konfigurationsmöglichkeiten zu bieten. Eine *einfach zu nutzende UI* (201) stellen die vier Systeme iP-Symcon, Logic Maschine 3, openHAB und openRemote. Allen ist eine einfache und intuitiv nutzbare Weboberfläche, in der über Knopfdruck Aktionen ausgelöst werden können gemein. Hierbei wird auf der Oberfläche die Wohnumgebung auf verschiedene Art (z.B. Listen, fotorealistisch) abgebildet. Diese digitalen Elemente lassen sich mit klassischen Bedienelementen (z.B. Schaltern) kombinieren und erzeugen so eine hohe *Gewohnheitsorientiertheit* (202). Hierbei ist es möglich, flexibel die Funktionalität auf verschiedenen Elementen der UI (auch parallel) umzusetzen. Die *Konfigurierbarkeit* (203) der GUI ist besonders bei iP-Symcon, Logic Maschine 3 und OSA ausgeprägt. Der erfahrene Benutzer kann vorgegebene UI-Bausteine beliebig anordnen und so spezielle Bedürfnisse verschiedener Rollen abbilden. Im Bereich der *UI-Erweiterbarkeit* (204) wurden digitale sowie analoge Steuerungen untersucht. Bestnoten wurden hier für eine hohe Flexibilität gegeben. Hier ist besonders universAAL hervorzuheben, das mit seinem Konzept eines UI-Buses dies im höchsten Maße erreicht. Der UI-Bus erlaubt das Übertragen abstrakter UI-Elemente und ihre Deserialisierung sowie Anzeige in einer passenden Visualisierung auf der Client-Seite. Bei den analogen Steuerungen liegen Plattformen mit Hardwareabstraktion klar vorne. Die meisten der getesteten Systeme liefern eine *robuste UI* (205). Die Bewertung bezieht sich auf das Verhalten der UI während der Installation, Konfiguration und Nutzung des Systems. Die *Ästhetik* (207) der Benutzerschnittstellen wurde durch

Anforderungen	Contronics	FHEM	iP-Symcon	Logic	Loxone	OSA	openHAB	OpenRemote	universAAL	Vera
201P Einfach nutzbares UI	2	3	1	1	2	2	1	1	2	3
202 Gewohnheitsorientiert	1	3	1	1	3	2	1	1	3	1
203 Gute Konfigurierbarkeit	2	4	1	1	3	1	2	2	2	4
204 Erweiterbarkeit der Benutzerschnittstellen	4	1	1	2	2	2	1	2	2	3
205 Robustheit	1	1	1	0	2	2	1	1	2	0
207 Ästhetik	3	3	1	3	2	3	2	3	3	3
208 Bedürfnisorientiert	2	4	1	2	4	2	2	1	1	3
210 Richtlinien für Benutzerschnittstellen	Ja	0	Ja	Nein	Ja	Nein	Ja	Ja	Ja	0
211P Nutzt bereits vorhandene Strukturen	2	1	1	1	1	3	1	1	2	1

Tabelle 3.5: Bewertung der Plattformkandidaten gegen die Anforderungen der Kategorie Benutzerschnittstelle.

den Vergleich mit anderen auch außerhalb der Hausautomatisierungsdomäne bewertet. Hierbei ist iP-Symcon hervorzuheben. Ihre UIs lassen sich auf Basis von Widgets im Aussehen individualisieren und weisen trotzdem immer ein schlüssiges Bedienkonzept auf. *Unterschiedliche Nutzerrollen zu bedienen* (208), wurde von folgenden Punkten beeinflusst: Eine Trennung zwischen der Konfigurations- und Nutzeroberfläche wurde als sinnvoll erachtet. Weiterhin war die Skalierbarkeit der Komplexität entsprechend der Rollen ein bewerteter Punkt. Auch die Barrierefreiheit (z.B. Kontrast, Schriftgröße, Text-To-Speech) der UI wurde betrachtet. Bis auf die Plattformen FHEM und Loxone setzen alle diese Anforderung zumindest ansatzweise um. Neben der Flexibilität zum Bedienen der verschiedenen Rollen bedarf es aber auch einheitlicher *UI-Richtlinien* (210). Diese setzen die meisten Systeme indirekt um, indem sie wenig Freiraum bei der Erstellung der UIs lassen. Viele Plattformen erlauben die Nutzung *bereits vorhandener Soft- und Hardware* (211). So unterstützen sie Web-UIs oder Tablets mit iOS oder Android.

Abschließend sollen zur flexiblen Verwendung der UIs die Konzepte von iP-Symcon und universAAL hervorgehoben werden. Durch ihre Abstraktion von UI-Elementen in Kombination mit einem Konfigurationswerkzeug lassen sich alle UI-relevanten Parameter anpassen. Ebenso bieten beide über sogenanntes User Profiling die Möglichkeit, flexibel auf verschiedene Rollen in der UI einzugehen.

Hardware ist eine wichtige Anforderungskategorie, um Informationen aus der Umgebung zu erhalten und ggf. die Umgebung wieder zu beeinflussen. Somit stellt die Hardwarekompatibilität einen zentralen Punkt der Plattformflexibilität dar. Es sei hierbei zu beachten, dass die Middleware nicht direkt Hardware unterstützt, sondern die entsprechenden Protokolle zur Kommunikation mit derselben bereitstellt. Einige Plattformen, darunter FHEM, iP-Symcon, OSA, OpenRemote, OpenHAB und universAAL, unterstützen durch ihre Erweiterbarkeit theoretisch jede mögliche Hardware. Im Zuge dieser Analyse wird aber nur aktuell nutzbare Hardware betrachtet und auf dieser Basis das beste Ergebnis dokumentiert. Die erste Anforderung bezieht sich auf die *Bedarfsorientiertheit* (301) der Middleware an die Hardware. Hierzu wurden verschiedene Faktoren auf ihre Diversität geprüft: Preis, Protokolle, Ästhetik, Datenübertragung und Aufwand für Installation und Konfiguration. Am besten schnitt iP-Symcon und FHEM ab, da sie eine sehr hohe Anzahl von Protokollen und Hardware unterstützen. Die Möglichkeit *günstige Hardware* (303) einzusetzen, bezieht sich auf die Anschaffung, Installation und Konfiguration derselben. Hier stellt sich das SF20 Protokoll als das Günstigste und der weit verbreitete KNX/EIB Standard als der Teuerste heraus. Ein weiterer Anforderungspunkt war, die Datenübertragung vor *Missbrauch zu schützen* (305). Besonders bei funkbasierter Hardware ist dies ein wichtiger Faktor. So bietet beispielsweise FS20 keine Sicherung und kann leicht mitgehört oder entsprechend beeinflusst werden. Alle anderen untersuchten Protokolle (bis auf EnOcean) verfügen über eine Verschlüsselung und meist über bidirektionale Kommunikation. Da keine Middleware nur EnOcean und FS20 unterstützt, konnten alle als sicher eingestuft werden. Die *Robustheit* (306) der untersuchten Hardware korreliert meist mit dem Preis. EnOcean bildet hierbei eine Ausnahme, da es als relativ hochpreisiges Produkt eine eher schlechte Robustheit liefert. KNX/EIB wird als sehr robust eingestuft. Die *Qualitätsstandards* (307) werden bei vielen Produkten (z.B. Z-Wave, Insteon, X10, KNX/EIB) durch Zertifizierung wie ISO 9001 erreicht. Die *Ästhetik* (308) scheint auch mit dem Preis zu korrelieren. So sind kostengünstige Sensoren (z.B. FS20) meist unansehnlicher als die teureren Konkurrenzprodukte

Anforderungen	Contronics	FHEM	iP-Symcon	Logic	Loxone	OSA	openHAB	OpenRemote	universAAL	Vera
301 Bedarfsorientiert	3	1	1	3	2	2	2	2	3	2
302P Schaltbare Sensoren und Aktuatoren	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
303 Günstig	1	1	1	4	4	3	3	2	1	3
304P Erweiterbare Sensoren und Aktuatoren	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
305 Missbrauchsschutz bei Datenübertragung	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
306 Robustheit	3	1	1	1	1	1	1	2	2	1
307 Qualitätsstandards	Nein	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Ja
308 Ästhetik	3	1	1	1	1	2	1	1	3	2
309 Unauffällig	3	1	1	1	1	2	1	1	3	2
310 Einfache Installation	2	1	1	2	2	3	3	3	1	2
311 Schnittstellenstandards	Nein	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Ja
312 Zustandsüberwachung	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
314 Breite Auswahl	2	1	1	1	1	2	1	1	3	2
315 Wartungsarm	3	1	1	1	1	2	2	2	3	1
316 Guter Support	3	1	1	1	1	1	1	1	3	1
317 Gute Wartbarkeit	2	1	1	1	1	2	4	2	1	3
318 Verständlichkeit	1	1	1	1	1	1	1	1	1	1

Tabelle 3.6: Bewertung der Plattformkandidaten gegen die Anforderungen der Kategorie Hardware.

(z.B. KNX/EIB). Während der Analyse ergab sich, dass die Bewertung der *Unauffälligkeit* (309) der Ästhetik folgt. Der *Installationsaufwand* (310) ist bei funkbasierten Geräte natürlich geringer als bei kabelbasierten Lösungen. Weiterhin ist damit auch der Preis einer kabelgebundenen Installation höher. *Schnittstellenstandards* (311) liefert nur KNX/EIB mit ISO 14543-3 bzw. EN50090. Weitere Produkte bieten zwar Schnittstellenbeschreibungen, aber keinen offiziellen Standard. Alle Geräte bis auf FS20 und EnOcean können Auskunft über ihren momentanen *Zustand* geben (312). Die *Mächtigkeit* (314) der Geräte lässt sich wieder auf die Protokollunterstützung reduzieren. Hierbei stellen beispielsweise FS20, HomeMatic, Z-Wave alle Geräte zur Abdeckung von Grund- und Komfortfunktionen. Allerdings bietet KNX/EIB oder EnOcean auch die Möglichkeit, leicht neue Geräte auf dem Protokoll zu nutzen. Die anfallenden *Wartungsarbeiten* (315) beziehen sich auf Aspekte wie den Tausch von Batterie oder defekten Sensoren. Hierbei konnten alle Plattformen bis auf Contronics punkten. Der *Support* (316) wurde über die jeweiligen Händler untersucht. Im hochpreisigen Segment fiel er besser aus. Aber auch die Verbreitung der Hardware spielte eine Rolle. So konnte Support bei hoher Verbreitung oft in öffentlichen Foren oder anderen Quellen gefunden werden. Bis auf Z-Wave zeigten alle Geräte eine gute *Wartbarkeit* (317). Z-Wave ist anspruchsvoller zu warten, da es ein Netz von Hardwareknoten aufbaut, welches für Laien nicht zu überschauen ist. Die *Verständlichkeit* (318) der jeweiligen Geräte war immer gegeben. Jedes Gerät entsprach der erwarteten Funktionalität und war nach der Installation, intuitiv zu nutzen.

Die Auswertung der Hardwareflexibilität zeigt, dass eine große Anzahl von unterstützenden Protokollen wünschenswert ist. Dies bedeutet wiederum eine hohe Anzahl von möglichen Sensoren, um die Benutzerbedürfnisse zu sichern. Im besonderen Maße entsprechen iP-Symcon und FHEM dieser Forderung. Plattformen, die nur auf günstige (Contronics) oder nur hochpreisige (Logic Machine 3) Hardware setzen, sind eher ungeeignet für eine flexible AAL-Plattform.

Support beschreibt im Rahmen der Auswertung nicht nur die direkte Unterstützung bei Wartung oder Ausfällen der AAL-Plattform, sondern auch Aspekte der Weiterentwicklung und des Vertriebs. Eine von vielen Akteuren eingebrachte Anforderung war ein *günstiger Support* (401). Hierbei wurden die anfallenden Kosten im Supportfall bewertet. Open Source-Projekte mit einer großen Community schnitten gut ab, aber auch Industrieprojekte mit Lizenzmodellen konnten punkten (z.B. iP-Symcon). Des Weiteren wurde die *Qualität des bereitgestellten Supports* (402) untersucht. Bewertet wurde der Gesamteindruck vom Online-Support, Antwortgeschwindigkeit und die Güte von Foren, Dokumentationen sowie Tipps zur Fehlerbeseitigung. Hierbei liegen die großen Industrieprojekte vorne. Neben einem guten Support ist es wichtig, dass man sich auch in *Zukunft auf Unterstützung* verlassen kann (403). Es fielen besonders OSA, openHab und universAAL durch einen schlechter werdenden Support auf, der in der Zukunft nicht abzuschätzen ist. Das Gleiche trifft auf die *Weiterentwicklung* (404) der drei Open Source-Projekte zu. Zwar haben alle drei eine Roadmap veröffentlicht und werden teilweise in großen Projekten (z.B. universAAL⁶ im EU Forschungsprojekt ReAAL⁷) verwendet, aber das sagt nichts über ihre Durchsetzungskraft in der Zukunft aus. *Referenzfällen zur Anwendungsfallentwicklung* (406) zur Unterstützung der Software Entwickler fehlen fast allen Plattformen. Dies liegt unter anderem an den oft einfachen Skriptsprachen

⁶<http://www.universAAL.org> (Zugriff am 21.11.2014)

⁷<http://www.cip-reaal.eu/home/> (Zugriff am 21.11.2014)

Anforderungen	Contronics	FHEM	iP-Symcon	Logic	Loxone	OSA	openHAB	OpenRemote	universAAL	Vera
401 Günstiger Support	1	1	2	2	2	1	1	2	1	2
402 Supportqualität	3	1	1	2	2	2	2	2	1	2
403 Zukunftsfähigkeit	2	1	3	2	2	2	2	2	3	2
404 Weiterentwicklung	2	2	1	1	1	2	2	1	3	2
405P Gute Dokumentation	0	2	2	2	2	2	2	2	1	2
406 Referenzfälle	2	2	2	3	2	2	3	2	2	2
407 Entwicklungsrichtlinien	2	2	1	2	2	2	1	2	2	2
408 Breite Community	3	1	1	2	1	1	2	2	3	2
409P Passendes Lizenzmodell	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja

Tabelle 3.7: Bewertung der Plattformkandidaten gegen die Anforderungen der Kategorie Support.

vieler Plattformkandidaten. Nur universAAL und iP-Symcon bieten echte Referenzfälle mit *Entwicklungsrichtlinien* (407). Eine *große Community* (408) bietet gerade bei Internetforen, Wikis und ähnlichen Quellen eine gute Supportbasis. Hier konnten besonders FHEM, iP-Symcon, Logic Machine 3, OSA und openHAB punkten. Ein *passendes Lizenzmodell* (P409) wurde nicht direkt bewertet, da es zu stark von den verschiedenen Nutzerbedürfnissen abhängt. Es wurde lediglich aufgenommen, dass alle Anbieter über ihre Lizenzmodelle Auskunft geben.

Insgesamt konnte bei allen Plattformen ein guter Überblick zu den Supportmöglichkeiten erarbeitet werden. Gerade in Bezug auf die in der Methode beschriebenen Plattfortmtests (vgl. Kapitel 3.3.1) wurden verschiedene Supportmöglichkeiten auch aktiv genutzt. In Bezug auf die Supportflexibilität muss man immer die grundlegende Entscheidung zwischen Industrie und Open Source mit ihren jeweiligen Vor- und Nachteilen treffen. Hat man sich entschieden, sind iP-Symcon und Loxone sowie FHEM und openHAB die besten Kandidaten.

Werkzeuge und die damit verbundene Unterstützung verschiedener Rollen ist ein wichtiger Teil von Plattformen. Sie vereinfachen den Umgang mit der Middleware (z.B. Prozesse automatisieren, Sensorintegration, Konfigurations- und Wartungsaufgaben unterstützen) und steigern die Akzeptanz. Die Pflichtenforderungen haben bereits *Sensor/Aktor Management* (501) und *Entwicklungssprachen* bestätigt (508). Die *Basiswerkzeugunterstützung* (502) war in Kombination mit den Managementwerkzeugen immer vorhanden und konnte im Zuge der ersten fünf Schritte (vgl. Methode Plattform prüfen) genutzt werden. Ein weiterer wichtiger Punkt sind die Werkzeuge zur Konfiguration der Plattform. Der *Konfigurationsbegriff* (503) vereint für diese Studie die Anpassbarkeit der Anwendungsfälle, der UI und der Hardware. Weiterhin wurden optionale Konfigurationsmöglichkeiten (z.B. Unterstützung bei komplexer Umgebungsmodellierung) positiv bewertet. Der beste Kandidat im Test war iP-Symcon. Es werden sehr hilfreiche Konfiguratoren für die Hardware und Nutzeroberfläche bereitgestellt, die darüber hinaus sehr flexibel in ihrer Nutzung sind. Es bedarf etwas Eingewöhnung in die Werkzeuge, um die komplette Umgebung in iP-Symcon zu modellieren, aber die Konfigurationsassistenten führen flexibel durch die einzelnen Schritte und ermöglichen auch automatische Diagnosen zum Erkennen fehlerhafter Zustände. Die *Installation neuer Hardware* (504) vollzieht sich im Fall von Logic Machine 3 und OpenRemote (professional) vollautomatisch. Auch bei FHEM werden neue Geräte spätestens beim ersten Senden von Signalen automatisch angelegt. Alle anderen Plattformen bedurften mehr Handarbeit beim Einbinden neuer Hardware. Die *Zugriffsbeschränkung/ -kontrolle* wurde (505) mit „Ja“ notiert, wenn mindestens ein Login Konfigurations- und Steuerungswerkzeuge sicherte. Sieben der zehn untersuchten Plattformen erfüllen diese Anforderung. Die *Robustheit* (506) wurde während der Testphasen 1-6 bewertet und zeigt bis auf die OpenSource Kandidaten sehr gute Ergebnisse. Vera und Logic Machine 3 konnten aufgrund des nicht zur Verfügung stehenden Miniservers nicht selbst getestet werden. Die Werkzeuge von Loxone konnten teilweise ohne Middleware getestet werden. Im Allgemeinen laufen nach der Installation alle Middlewares robust. Die *Erweiterbarkeit* (507) auf Werkzeugebene wurde auf die Middleware bezogen. Sie kann durch selbst erstellte Skripte (einfache Logik und Abläufe), Module (komplexe Logik und Abläufe) oder Bundles (komplett eigene Softwarepakete integrierbar) erweitert werden. Die Erweiterung auf Bundleebene erlauben FHEM (modularen Aufbau von Perl), OSA (OSGi ähnlicher Aufbau in C#), openHAB (OSGi), OpenRemote (mo-

Anforderungen	Contronics	FHEM	iP-Symcon	Logic	Loxone	OSA	openHAB	OpenRemote	universAAL	Vera
501P Sensor- und Akteur-Management	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
502 Grundfunktionen der Hausautomation	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
503 Gute Konfigurierbarkeit	3	3	1	3	2	2	2	2	1	3
504 Einfache Installation und Inbetriebnahme neuer Hardware	2	2	2	1	3	2	2	2 / 1 ^a	3	2
505 Zugriffsbeschränkung	Nein	Ja	Ja	Ja	Ja	Nein	Ja	Ja	Nein	Ja
506 Robuste Werkzeuge	1	1	1	0	0 / 1 ^b	2	2	1	2	0
507P Erweiterbarkeit	3	1	3	3	4	1	1	1	1	3
508P Mächtige Entwicklungssprache	3	2	3	3	3	1	1	1	1	4
509 Schnelle Entwicklung von Anwendungsfällen	2	2	2	1	2	1	1	1	3	1
510 Einfache und schnelle Installation der AAL-Plattform	1	2	1	1	1	2	1	2	3	1
511 Fernwartung	Nein	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Ja	Ja
512 Entwicklungsunterstützung	4	4	2	2	4	4	2	3	1	2
513 Diagnosewerkzeuge	4	4	1	4	3	4	4	4	4	4
514 Schnittstellen	4	2	2	4	2	2	1	2	1	4
515 Distributionsdienste	Nein	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Ja	Ja
516 Skalierbare Komplexität	Nein	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein

Tabelle 3.8: Bewertung der Plattformkandidaten gegen die Anforderungen der Kategorie Werkzeuge.

^aabhängig von den Komponenten^babhängig von den Komponenten

dularer Ausbau in Java und saubere Schnittstellen) und universAAL (OSGi). In diesem Zuge wurde auch die Mächtigkeit der Entwicklungssprachen bewertet. Von den vorhandenen Sprachen wurde Java und C# als sehr mächtig angesehen. Geht es darum *schnell neue Anwendungsfälle zu erstellen* (509), spielen noch andere Faktoren eine Rolle. Die Hälfte der Plattformen (Logic Machine 3, OSA, openRemote, Vera) unterstützt das Erstellen von Anwendungsfällen mittels einer Skriptsprache, hierunter leidet jedoch auch die Mächtigkeit. Die Skriptsprachen reichen aus, um einfache logische Verknüpfungen zwischen Sensorinformationen der Middleware zu erreichen und ggf. in einer Aktorausführung zu enden. Die *Installation der Middleware* (510) wurde positiv bewertet, wenn ein Installationsassistent vorhanden war. Es ist anzumerken, dass der Installationsassistent von OSA reproduzierbare Fehler auf verschiedenen Computern erzeugte. Die Datenbankeinstellungen mussten manuell konfiguriert werden, um die Plattform unter Windows in Betrieb zu nehmen. Die Möglichkeit, per *Fernwartung* (511) auf die Middleware zuzugreifen, boten sechs der untersuchten Systeme. Hierbei sind alle als „1 zu 1“-Verbindung ausgelegt, so dass nirgends ein Wartungszugriff von mehreren Plattformen möglich ist. Werkzeuge zur *Entwicklungsunterstützung* (512) beschränken sich auf die jeweiligen IDEs oder Editoren. Nur für universAAL sind 12 Eclipse-Erweiterungen verfügbar, die dem Entwickler auf verschiedene Art Arbeit abnehmen können. *Diagnosewerkzeuge* (513) helfen, Fehler in allen Teilen der Plattform aufzuspüren. Nur iP-Symcon bot im Test eine überzeugende automatische Fehlerdiagnose. In der Wartungssoftware von Loxone lassen sich Simulationen der gesamten AAL-Umgebung fahren. Die *Schnittstellen* (514) der Middleware wurden in Bezug auf Funktionsumfang und Mächtigkeit bewertet. Besonders gut wurden openHAB und universAAL bewertet, da sie mit OSGi beide Kategorien erfüllen. Bei den *Softwaredistributionswerkzeugen* (515) liegen OSA und universAAL deutlich vorne, da sie mit einem eigenem Online-Dienst die Middleware und Anwendungsfälle verteilen. So kann die Plattform auf einfache Art und Weise erweitert und aktuell gehalten werden. Der letzte Punkt in dieser Kategorie ist die *Skalierbarkeit* (516). Bis auf Contronics und Vera gibt es hier keine Einschränkungen bei der Anzahl der nutzbaren Geräte.

In der Werkzeugkategorie gab es keinen eindeutigen Favoriten, da die einzelnen Plattformen auf verschiedene Schwerpunkte bei der Unterstützung setzen und somit bestimmte Anforderungen sehr gut erfüllen und andere gar nicht. Insgesamt können aber FHEM, iP-Syncom und universAAL im vorderen Feld gesehen werden.

3.3.3 Zusammenfassung

Keine der untersuchten Plattformen kann alle Anforderungen an eine flexible AAL-Umgebung zufriedenstellend erfüllen. Viele Anforderungen wurden von den Plattformkandidaten erfüllt, andere aber auch nicht. Es ergibt sich hierbei ein sehr divergentes Bild. Da auch AAL-Umgebungen sehr verschiedene Ansprüche haben, kann man zwar für jede Umgebung die Anforderungen gewichten und entsprechend eine Plattform wählen. Dies widerspricht aber einer flexiblen Lösung mit allen ihren Vorteilen der Interoperabilität und Wiederverwendbarkeit.

Für eine flexible AAL-Plattform konnten im Zuge dieser Metastudie die folgenden Aspekte herausgearbeitet werden:

- Es fehlt an tauglichen Anwendungsfällen für den AAL-Bereich. Diese sollten einfach erstellbar und später leicht personalisierbar sein.

- Dafür bedarf es einer Konfigurationsmöglichkeit der Anwendungsfälle, die schon während ihrer Entwicklung eingearbeitet wird.
- Die Anwendungsfälle kommunizieren über Benutzerschnittstellen mit den verschiedenen Akteuren der AAL-Umgebung. Jeder Akteur hat einen anderen Kontext, so dass die Benutzerschnittstelle flexibel an seine technischen Fähigkeiten und seinen speziellen Informationsbedarf angepasst sein muss.
- Die an AAL-Plattformen angeschlossene Hardware kann in den meisten Fällen schon flexibel erweitert werden. Das geschieht aber nur auf den unteren Protokollebenen, so dass nur selten eine Abstraktion im Sinne einer echten Middleware erfolgt. Eine Hardware-Abstraktion auf höheren Ebenen der Plattform ist notwendig.
- Je mehr Flexibilität eine Plattform bietet, desto höher ist der Bedarf an passender Werkzeugunterstützung, um fehlerfrei und schnell bei steigender Komplexität zu arbeiten.

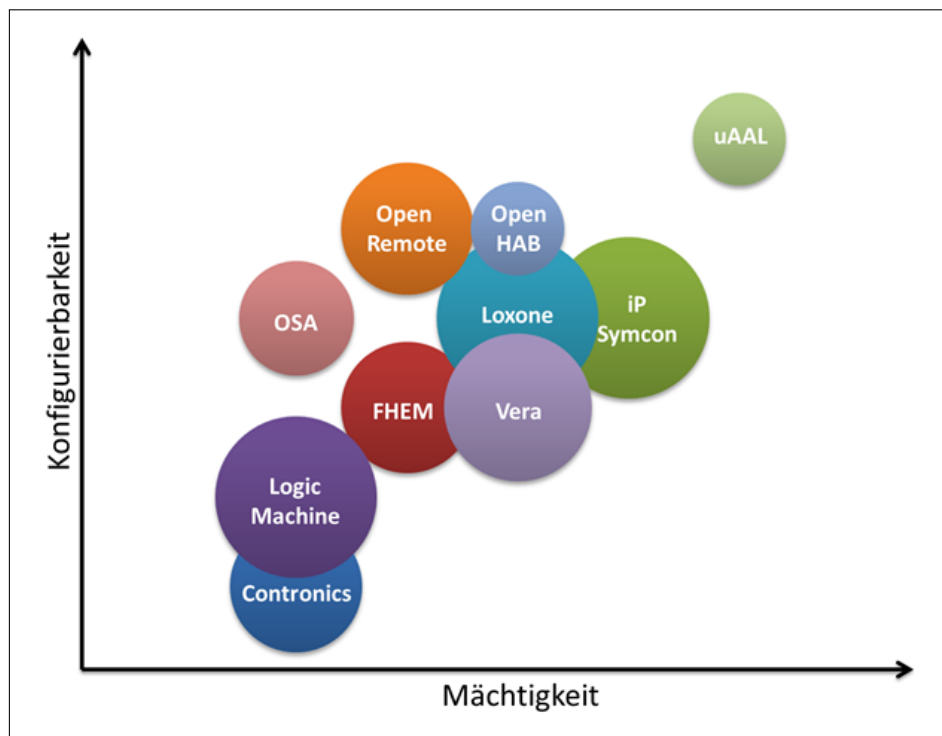


Abbildung 3.2: Bewertung der Plattformkandidaten: Kreisgröße: Robustheit; X-Achse: Mächtigkeit; Y-Achse: Konfigurierbarkeit

In Abbildung 3.2 wurde eine Bewertung der 10 Plattformkandidaten im Bezug auf ihre Flexibilität unternommen.

Hierbei stützen Konfigurierbarkeit, Mächtigkeit und Robustheit zu gleichen Teilen die Flexibilität einer AAL-Plattform. Ein robustes System (Kreisgröße), dem man vertrauen kann, ist die Basis für Flexibilität. Weiterhin kann nur eine hohe Anpassbarkeit durch Konfiguration (Y-Achse), ein flexibles Eingehen auf spezielle Bedürfnisse und Rahmenbedingungen sicherstellen. Außerdem ist eine gewisse Mächtigkeit

(siehe X-Achse) der Plattform gefragt. Nur so kann eine ausreichende Komplexität erreicht werden, um flexibel auf neue Anforderungen einzugehen.

Die Achsen sowie die Kreisgrößen sind mit keinem Skalar versehen, da in dieser Darstellung über die reinen Zahlenwerte der Auswertung (siehe Tabellen 3.4, 3.5, 3.6, 3.7 und 3.8) hinaus gegangen werden soll. In einer Sitzung der Fokusgruppe 02 wurde diese Darstellung im gemeinsamen Konsens erarbeitet. Hierbei ist eher die Positionierung der einzelnen Plattformen zueinander ausschlaggebend als ihre absolute Position im Koordinatensystem. Sie spiegelt den qualitativen Eindruck der Fokusgruppe 02 wider, der während der Analyse in den Living Labs gewonnen wurde.

3.4 Konfigurationsbedarf von AAL-Referenzanwendungsfällen

Im letzten Teil dieses Kapitels werden die Anforderungen an die Konfiguration von AAL-Anwendungsfällen analysiert. Ein AAL-Anwendungsfall besteht aus bis zu drei Teilen. Die Implementierung eines AAL-Anwendungsfalls ist ein AAL-Service. Des dessen Grundlage ist eine Software-Komponente, die auf Basis der AAL-Plattform bestimmte Bedürfnisse der Endanwender unterstützt. Hierbei können AAL-Services als weitere Komponenten Hardware und menschliche Ressourcen über Software in die AAL-Umgebung integrieren. Der Fokus liegt somit auf der Software. Die anderen Komponenten werden soweit vorhanden und notwendig untersucht.

„Für einen einzelnen Anwendungsfall [...] lässt sich oftmals relativ genau festlegen, welche Optionen erforderlich sind und welche nicht.“ [VDE 12a, S.58] Um eine allgemeingültige Aussage zum Konfigurationsbedarf abzuleiten, ist es wichtig, Repräsentanten in Form von AAL-Referenzanwendungsfällen zu untersuchen. In vereinzelt Forschungsprojekten gibt es bereits umgesetzte Beispiele für AAL-Anwendungsfälle. Diese stellen eine gute Basis dar, um die technischen Anforderungen an die Konfiguration zu erheben, die sich bei der Integration dieser Anwendungsfälle in eine flexible AAL-Umgebung ergeben.

Hierfür wurden Anwendungsfallkategorien geschaffen, die alle in AAL-Umgebungen vorkommenden Anwendungsfälle abdecken. Diesen Kategorien wurden in einem weiteren Schritt die verschiedenen, bereits existierenden Anwendungsfälle zugeordnet und unter ihnen Repräsentanten gewählt. Diese Repräsentanten wurden als AAL-Referenzanwendungsfälle im letzten Teil dieses Kapitels auf ihren Konfigurationsbedarf hin untersucht.

3.4.1 Validierung der Anwendungsfallrepräsentanten

Dieses Kapitel präsentiert die Ergebnisse der Fokusgruppe 03, welche die Hypothese 1 aus Kapitel 2.1.3 bestätigen. Unter ständiger Berücksichtigung von Konfigurationsaspekten wurden die 55 Anwendungsfälle aus 11 verschiedenen Projekten zusammengeführt. So entstanden die 10 Repräsentanten und ihre Zuordnung in die AAL-Anwendungsfallkategorien.

Im ersten Teil dieses Kapitels werden die von der Fokusgruppe 03 validierten Repräsentanten in ihrer finalen Version kurz beschrieben. Eine ausführliche Beschreibung der Anwendungsfälle findet sich in [Guil13, S. 260ff].

AALS I: Health Manager (Gesundheitsmanager): Der Health Manager unterstützt Endanwender mit Krankheiten im täglichen Leben durch Überwachung der Vital-Parameter. Diese können einem Arzt live zur Verfügung gestellt werden oder zu Dokumentationszwecken verwendet werden. Weiterhin können diese Informationen auch anderen Akteurgruppen zur Verfügung gestellt werden, um den Bewohner beim Selbstmanagement seiner Krankheit geeignet zu unterstützen.

AALS II: Nutritional Advisor (Ernährungsberater): Auf der Basis von Gesundheitsprofilen hilft dieser Service dem Endbenutzer und besonders dem Bewohner, sich richtig zu ernähren. Weiterhin bindet der Service externe Quellen (z.B. Sensoren, Wetterdaten, Ärzte) ein, um weitere Daten zur Bestimmung der optimalen Ernährungsstrategie zu nutzen.

AALS III: Agenda and Reminders (Agenda und Erinnerungen): Dieser Service hilft, kognitive Schwächen auszugleichen, indem er zum richtigen Zeitpunkt an Ereignisse oder Dinge erinnert. Die Informationen hierzu kann er aus verschiedenen Quellen beziehen. Das Ziel ist es, den Bewohner zu entlasten.

AALS IV: Safety and Security at Home (Sicherheit zu Hause): Die Basis für diesen Service ist ein Benachrichtigungssystem, welches es erlaubt, sicherheitsrelevante Meldungen an den Endanwender auszuliefern. Weiterhin besteht die Möglichkeit, auf diese Meldungen zu reagieren. Dies kann auf Wunsch auch automatisch geschehen.

AALS V: Help when outdoors (Navigation draußen): Durch verschiedene Verfahren der Lokalisierung und verschiedene Hilfestellungen (z.B. Notruf oder nach Hause führen) ermöglicht es dieser AAL-Service, dass der Bewohner sich unabhängig außerhalb seiner Wohnung in der AAL-Umgebung bewegen kann.

AALS VI: Long Term Behavior Analyzer (Langzeitmonitoring): Dieser Service erkennt durch Langzeitanalysen bestimmte Verhaltensmuster und kann auf ungewöhnliches Verhalten schließen und geeignet reagieren. Dies hilft beispielsweise dem Arzt bei der Diagnose von Krankheiten. Weiterhin kann dieser Service den Bewohner im täglichen Leben durch Hinweise und Informationen unterstützen.

AALS VII: AALficiency (Energieeffiziente Haussteuerung): Dieser Service unterstützt den Endanwender durch Komfortfunktionen im täglichen Leben. Hierbei wird im Besonderen auf die Energieeffizienz geachtet.

AALS VIII: Food and Shopping (Lebensmittel- und Einkaufsunterstützung): Basierend auf Gesundheitsprofilen, der Lebensmittelsituation zu Hause und anderen Informationsquellen (z.B. Sensoren) werden Einkaufslisten zusammengestellt. Diese können dann selbstständig oder durch einen Dienstleister abgearbeitet werden.

AALS IX: Personal Safety (Persönliche Sicherheit): Dieser Service nutzt alle Informationen der AAL-Umgebung, um auf mögliche medizinische Risikosituationen aufmerksam zu machen. Weiterhin kann er beim Eintreten einer solchen

Kategorien	AALS									
	I	II	III	IV	V	VI	VII	VIII	IX	X
Informationsassistentz	X	X	X	X	X		X	X	X	X
Intelligentes Umgebungsverhalten	X	X	X	X			X		X	
Vorhersehen von Notfallsituationen	X				X	X	X	X		X
Erkennen von Notfallsituationen	X				X	X		X		X
Sicherheit			X	X	X	X			X	X
Privatsphäre	X	X		X		X			X	

Tabelle 3.9: Zuordnung der Referenzanwendungsfälle zu den Anwendungsfallkategorien. Die grauen Felder zeigen hierbei Differenzen zwischen dem Vorschlag und der validierten Version der Fokusgruppe 03.

Situation entsprechend deeskalieren. Hierzu können verschiedenen Menschen und Aktionen nacheinander beteiligt oder ausgeführt werden.

AALS X: Medication Manager (Medikationsunterstützung): Dieser Service unterstützt den Bewohner bei der Medikamenteneinnahme durch Überwachung und Erinnerung.

Nach der Validierung der Referenzanwendungsfälle durch die Fokusgruppe 03 werden die Anwendungsfälle den Anwendungsfallkategorien von VDE (2011) und Andrushevich et al. [AKBK09] zugeordnet. Wie in der Methodik beschrieben, wurde der Vorschlag als Diskussionsgrundlage von zwei Domänenexperten erstellt. Die Fokusgruppenmitglieder bestanden aus Experten der Anwendungsfallentwicklung und waren zum größten Teil direkt in die Implementierung der 10 Referenzanwendungsfälle eingebunden. Somit ist ihre Kompetenz zur Bewertung der Einordnung gesichert. Die Gruppe diskutierte den Vorschlag der zwei Domänenexperten und kam zum in der Tabelle 3.9 dargestellten Ergebnis. In der Diskussion wurde das genauere Verhalten der Services erklärt und es konnte sich auf dieser Basis auf eine einheitliche Bewertung geeinigt werden. Die grauen Felder zeigen hierbei Differenzen zwischen dem Vorschlag und der validierten Version der Fokusgruppe 03. Bis auf den Anwendungsfall AALS VI herrscht eine hohe Übereinstimmung in der Einschätzung. Die abweichende Bewertung des Anwendungsfall AALS VI ist begründet durch ein teilweise falsches Verständnis seiner Funktionalität bei der ersten Bewertung. Mit Hilfe des Entwicklers konnte in der Validierung aber eine Korrektur vorgenommen werden.

Die zehn Anwendungsfälle können im Weiteren als Repräsentanten genutzt werden, da sie zum einen aus der Aggregation und Konsolidierung von 55 AAL-Anwendungsfällen hervorgegangen sind und zum anderen eine hohe Abdeckung aller AAL-relevanten Kategorien zeigen. Das Ergebnis dieses Kapitels ist somit eine Menge von zehn repräsentativen Anwendungsfällen, die als Grundlage dienen, um den Konfigurationsbedarf bei AAL-Services genauer zu untersuchen.

3.4.2 Bedarfsextraktion

Die Hypothese 2 wurde durch die Fokusgruppe 03 untersucht. Sie bestimmte für die einzelnen Anwendungsfälle jeweils den Konfigurationsbedarf, um den Ansprüchen für ein flexibles Management der AAL-Umgebung zu genügen. Hierzu wurden neben den sechs Teilnehmern der Fokusgruppe 03 insgesamt fünf weitere Entwickler einbezogen. Im Folgenden werden die Ergebnisse dargestellt. Hierbei wurde jeder Referenzanwendungsfall in der Gruppe besprochen und seine Konfigurationsparameter identifiziert.

Diese potenziellen Konfigurationsparameter lassen sich in drei Gruppen einteilen:

Einfache Parameter: In diese Gruppe fallen Parameter, die einfache atomare Werte beinhalten und in der Konfiguration leicht zu behandeln sind. Beispiele hierfür sind Datenwerte vom Typ Boolean, Integer, Double und String. Diese Parameter sollten nur innerhalb des Anwendungsfalls bekannt sein. Meistens ist eine Validierung der Eingabe für diese Parameter einfach.

Komplexe Parameter: Diese Gruppe vereint alle komplexeren Objekte, die für die Konfiguration genutzt werden. Hierunter fallen Strukturen wie Listen, Dateien oder Objekte, die mehrere einfache Datentypen vereinen. Ihre Verarbeitung innerhalb der Konfiguration ist anspruchsvoller. Diese Parameter sollten nur innerhalb des Anwendungsfalls bekannt sein. Meistens ist eine Validierung der Eingabe für diese Parameter komplizierter.

Wissensobjekte: Sie beinhalten Objekte, deren Instanzen auch von andere Komponenten in der AAL-Umgebung genutzt werden können. Sie können eine beliebige Komplexität aufweisen, solange sie sich in Ontologien modellieren lassen. Hier stellt sich die Frage, ob die vorhandenen AAL-Domänenontologien diese Wissensobjekte bereits modellieren oder neue Konzepte erforderlich sind.

Die folgenden Tabellen 3.10-3.19 stellen die Auswertung des Konfigurationsbedarfs der zehn verschiedenen Anwendungsfälle dar. Zu jedem Parameter wurde sein Name notiert sowie ein Beispiel und eine Beschreibung, wozu er dient⁸. Nach dem Doppelstrich ist notiert, ob es sich nach Ansicht der Fokusgruppe 03, um einen einfachen oder einen komplexen Parameter handelt und wie die Validierung des Parameters eingeschätzt wird. Die Daten links vom Doppelstrich sind einzustellende Parameter aus dem jeweiligen Anwendungsfall und werden mit dem dazugehörigen Entwickler erfasst. Die Daten rechts vom Doppelstrich stellen die Bewertung der Parameter durch die Fokusgruppe 03 bezüglich der Konfiguration dar. Dieser Schritt gehört teilweise in die Konzeption und ist nur zum besseren Verständnis an dieser Stellen in den Tabellen mit dargestellt.

Teilweise kann man argumentieren, dass einzelne Parameter besser als Wissensobjekt zu modellieren sind. Beispiele hierfür sind in Tabelle 3.11 „User Email“ oder in Tabelle 3.14 „LCaregiver_name“. Bei Unstimmigkeiten in der Gruppe hatte der Entwickler des Anwendungsfalls das letzte Wort. So waren in den Beispielen die Entwickler der Meinung, aus Gründen der Sicherheit wollen sie die Daten nicht in der gesamten Umgebung abfragbar machen.

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Health_Care_Server	HL7://health.care.co: 7074	URL to which connect when using health care protocols like HL7.	String	Einfach	Einfach
Security_Parameters	Username:Password	for accessing the external healthcare system (username, password, additional parameters if the data is encrypted)	String	Einfach	Einfach
Connection_Policy		Policy if the external server is down (number of retries, timestamp of the last successful transfer of data, retry interval)	LIST	Einfach	Komplex

Tabelle 3.10: Konfigurationsbedarf: AALS I: Health Management

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Behave_rules	{(Kitchen,1,7,30)}	Rules that in case are true an alert is sent. (Kitchen, 1,7,30) indicates that from 1a.m to 7 a.m. if I stay more than 30 minutes in the Kitchen it sends an alert.	String	Komplex	Komplex
Nutritional Web Server IP	158.42.166.101	Web server that contains nutrition services.	String	Einfach	Einfach
Login and password to access Nutritional Web Server		To authenticate users (i.e. informal caregivers) that could access to the nutritional service remotely.	String	Einfach	Einfach
User Email		To notify the user by email.	String	Einfach	Einfach
Relative's email		To notify the relative by email.	String	Einfach	Einfach
User ID		To notify the relative by email.	String	Einfach	Einfach

Tabelle 3.11: Konfigurationsbedarf: AALS II: Nutritional Advisor

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Remote access URL	http://IP:8080/ versAAL	Web (remote) access to the service. IP is IP address of home server. Note: listening port can be changed by changing following VM parameter: - Dorg.osgi.service.http.port=8080	String	Einfach	Einfach
Confadmin	Copy confadmin dir to workspace\runDir	result:workspace\runDir\confadmin Here the main menu and other configuration files are stored.		Komplex	Komplex
Database credentials	Username:password	For accessing the service data.	String	Einfach	Einfach

Tabelle 3.12: Konfigurationsbedarf: AALS III: Agenda and Reminder

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Potential risks	{water leak, gas leak, open window left, open door left, fire}	These potential risks can be configured by the AP in order to be informed when any of them took place.	String	Komplex	Komplex
Devices to be controlled remotely	{central heating, gas heating, water heater, etc}	The device that are allowed to be controlled remotely by AP user	String	Komplex	Komplex
Warning form	{speakers, PC terminal, vibration}	How the warning will be presented to the user. This may take advantage of the Input/ Output handler of the universAAL architecture.	String	Komplex	Komplex

Tabelle 3.13: Konfigurationsbedarf: AALS IV: Safety and Security at Home

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Emergency_tel_number	INT	This is the phone number where the SMS is sent if there is any emergency.	INT	Einfach	Einfach
I_Caregiver_name	LIST	To personalize the notifications send by the service.	LIST	Einfach	Einfach
I_Caregiver_email	LIST	To send notifications by emails to the informal caregiver.	LIST	Einfach	Einfach
I_Caregiver_tel_number	LIST	To call the informal caregiver in case of risky situations.	LIST	Einfach	Einfach
F_Caregive_name	LIST	To personalize the notifications send by the service.	LIST	Einfach	Einfach
F_Caregiver_email	LIST	To send notifications by emails to the formal caregiver.	LIST	Einfach	Einfach
F_Caregiver_tel_number	LIST	To call the formal caregiver in case of risky situations.	LIST	Einfach	Einfach
Safe_area	COORDINATES	To locate the assisted person in a safe area.	String	Einfach	Komplex
Allowed_people_name	LIST	To identify who can access to the assisted person location.	LIST	Einfach	Komplex

Tabelle 3.14: Konfigurationsbedarf: AALS V: Help when Outdoors

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
URL of the Server	Web URL with Webservices to access	The software needs this information in order to connect to the remote web service.	String	Einfach	Einfach
URL of the user	Web URL to allow external access	It is required to allow that external users (relative and technicians) access functionalities as well as the AAL Space aspects.	String	Einfach	Einfach
Basic knowledge	DROOLS format.	Basic knowledge are facts that are used as a starting point for the DROOLS system to interpret the reality.	File	Einfach	Komplex
End user rules set	DROOLS format.	Rules are the basis of the system. The system is able to model user behavior using rules and relate different detected events in order to create knowledge of a higher abstraction level.	File	Einfach	Komplex
Basic set of personalized reporting mechanism	Some rules and DB	The system should be configured to start reporting about some personalized behaviors. There will be a default reporting that can be changed by the end user.	File	Einfach	Komplex
Service started	ON/OFF	This parameter is used when the service starts in order to know if the service starts to collect information or not to be used for the modeling.	Boolean	Einfach	Einfach
User accounts	Encrypted file. Key,value.	ACL with permissions and user/password settings to give access to information and configuration of the system.	String	Einfach	Komplex

Tabelle 3.15: Konfigurationsbedarf: AALS VI: Long Term behaviour Analyser

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
URL of the user	Web URL to allow external access	It is required to allow that external users (relative and technicians) access functionalities as well as the AAL Space aspects.	String	Einfach	Einfach
Basic knowledge	DROOLS format.	Basic knowledge are facts that are used as a starting point for the DROOLS system to interpret the reality.	File	Einfach	Komplex
End user rules set	DROOLS format.	Rules are the basis of the system. The system is able to model user behavior and the electrical consumption based on the data extracted from the devices and the set of rules, to create high level knowledge.	File	Einfach	Komplex
Challenges	DB with the description of the challenges and punctuation of each one	System's operation is based on a series of challenges that the user must complete to obtain certain points. It is necessary to have those challenges described in the DB.	File	Einfach	Komplex
Service started	ON/OFF	This parameter is used when the service starts in order to know if the service starts to collect information or not to be used for the modeling.	Boolean	Einfach	Einfach
User accounts	Encrypted Key,value. file.	ACL with permissions and user/password settings to give access to information and configuration of the system.	String	Einfach	Komplex

Tabelle 3.16: Konfigurationsbedarf: AALS VII: AALfficiency

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Health profile of the AP	{Blood pressure, glucose level, chronic diseases}	In order the shopping list to be personalized towards his/her nutritional needs.	String	Komplex	Komplex
AP's Address	{Address name, Address number, Postal code, City}	The home address of the AP; in order to be delivered shopping from the supermarket to AP's home.	String	Komplex	Komplex
Supermarket's email address	{email address name}	In order to receive the shopping list of the AP.	String	Einfach	Einfach
Informal Caregiver's credentials	{username, password}	In order to be able to edit the shopping list of the AP.	String	Einfach	Einfach

Tabelle 3.17: Konfigurationsbedarf: AALS VIII: Food and Shopping

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
Emergency_telnumbers and priorities	(0,+34 976677888; 1,+34 682148148;)	This is the phone number where the SMS is sent if there is any emergency. In case the first in the list fails, then the alert must be send to the next one in the list.	LIST	Komplex	Komplex
Emails	mother@terra.es; Car-men@hotmail.com	Emails are sent to those accounts.	String	Einfach	Einfach
Active communication means	SMS, reliable SMS, email, web, social networks	The list mention those that should be used in parallel. Priorities could also be established.	String	Einfach	Einfach
Behave_rules	{{(Living,1,7,30), (Bath,1,7,30)}	Rules that in case are true an alert is sent. (Living, 1,7,30) indicates that from 1a.m to 7 a.m if I stay more than 30 minutes in the Living room it sends an alert to the emergency telephone number.	String	Komplex	Komplex
Web users	l: u: mother; p:123;	Users that have permission to access assisted person's data.	LIST	Komplex	Komplex
Public IP address	IP_Address	Having this information helps to provide access from the outside and configure Dynamic DNS accounts.	String	Einfach	Einfach
Report frequency	Daily, weekly, monthly	The report about assisted person's activity sent to the informal caregiver.	String	Einfach	Einfach
Report email address	Email		String	Einfach	Einfach

Tabelle 3.18: AALS IX: Personal Safety

Parameter Name	Beispiel	Beschreibung	Datentyp	Konfigtyp	Validation
drug_db_uri	— ^a	The URL of the online medicine database	String	Einfach	Einfach
drug_db_user	—	The user name used for accessing the online medicine database	String	Einfach	Einfach
Drug_db_password	—	The password needed for accessing the online medication database	String	Einfach	Einfach
users_involved	—	Comma separated list of the URIs to the profiles of the involved assisted persons	LIST	Komplex	Komplex

Tabelle 3.19: Konfigurationsbedarf: AALS X: Medication Manager

^aWurde durch den Entwickler nicht angegeben.

Parameter Name	Beschreibung	Ontologie
Name	To personalize messages I need his name. To put his name in the Motivation Messages.	Instanz
Gender	To personalize motivation and health strategies.	Instanz
Birth date	To personalize motivation and health strategies.	Instanz
Illnesses	To personalize motivation and health strategies.	Konzept
Health Habits	To focus motivation into pursuing healthy habits. If the user smoke the service will motivate the user to quit smoking	Konzept
Formal Caregivers	To grant access to these users, and send then the required information.	Instanz
Informal Caregivers	To grant access to these users, and send then the allowed information.	Instanz
Treatments	Although Health Manager handles the treatments, these can be available for other services to take into account.	Konzept

Tabelle 3.20: Wissensobjekte: AALS I: Health Management

Die Wissensobjekte wurden in gleicher Weise in der Gruppe analysiert und in den Tabellen 3.20-3.29 festgehalten. Hierbei wurde zu jedem Wissensobjekt der Parametername notiert sowie eine Beschreibung seines Inhalts und seiner Funktion. Nach dem Doppelstrich wurde die Bewertung vorgenommen, ob das Wissensobjekt nur in den vorhandenen Domänenontologien instantiiert werden muss oder ob neue Konzepte notwendig sind. Hierbei stellten die Entwickler⁹ die Parameter bereit, die sie in den jeweiligen Anwendungsfällen als Wissensobjekt nutzen wollen und die Fokusgruppe 03 nimmt die Konzeptionelle Designentscheidung vor (rechts vom Doppelstrich).

War keine eindeutige Entscheidung zu treffen, ob es sich um ein Wissensobjekt handelt oder eher um einen Konfigurationsparameter, hatte der Entwickler des jeweiligen AAL-Anwendungsfalls das letzte Wort. So wollten die Entwickler beispielsweise, dass die Logindaten für Facebook (vgl. Tabelle 3.26 „Facebook account details“) oder für den Zugriff auf die Langzeitanalyse (vgl. Tabelle 3.25 „User/password“) von anderen AAL-Anwendungsfällen in der AAL-Umgebung möglich ist.

Neben der reinen Analyse der zu konfigurierenden Parameter brachten die Gruppendiskussionen drei weitere Erkenntnisse, die für die AAL-Anwendungsfallkonfiguration wichtig sind:

Ganzheitliche Konfiguration: Flexibilität ist nur zu erreichen, wenn die Konfiguration von Anwendungsfällen von der Entwicklung über ihre Nutzung bis hin

⁸Der linke Teil der Tabellen wurde von den Entwicklern ausgefüllt. Schreib- und Ausdrucksfehler wurden nicht korrigiert.

⁹Schreib- und Ausdrucksfehler wurden nicht korrigiert.

Parameter Name	Beschreibung	Ontologie
Name	To personalize messages.	Instanz
Last name		Instanz
Sex		Instanz
Relationship status		Instanz
Nutritionist email		Instanz
Email	To receive custom notifications, shopping lists, meal invitations	Instanz
User id	Identify user	Instanz
Username		Instanz
Age	To personalize messages.	Instanz
Language	Tailor content	Instanz
Country	Catering service provider needs it	Instanz
City		Instanz
Postal code		Instanz
Country		Instanz
Home phone number		Instanz

Tabelle 3.21: Wissensobjekte: AALS II: Nutritional Advisor

Parameter Name	Beschreibung	Ontologie
Formal Caregiver	To grant access and send them the required information.	Instanz
Informal Caregiver	To grant access and send them the required information.	Instanz

Tabelle 3.22: Wissensobjekte: AALS III: Agenda and Reminder

Parameter Name	Beschreibung	Ontologie
Name	The name of AP in the house	Instanz
Surname	The surname of AP in the house	Instanz
Address	The home address of the AP; it will be used by Informal caregivers	Instanz
Apartment's Location	The floor where the apartment is.	Instanz
RF-IDs	The RF-IDs are allowed to access the front door mechanism and enter at AP's home. E.g. RF-IDs of (informal) caregivers.	Konzept
Contact (details)	Connected to RF-IDs. These are the contact details of the users (caregivers) are allowed to enter at home. This information is used in order to inform the AP who is entering/accessing his/her home. The mandatory contact details are the name and the surname of the visitor.	Konzept

Tabelle 3.23: Wissensobjekte: AALS IV: Safety and Security at Home

Parameter Name	Beschreibung	Ontologie
User name	Because I want to personalize messages and then I need his name. To put his name in the SMS when an emergency is sent.	Instanz
User address	It is important to be used by the component that manages how the user is guided to their home.	Instanz
Home area	The definition of this area is important because the service needs it to determine if the user is in a controlled place (home area) or not.	Instanz
Safe area	Similar to the Home area parameter but this parameter defines an area biggest than the home area (because probably it includes the home area) where the user is "controlled" and therefore safe.	Instanz

Tabelle 3.24: Wissensobjekte: AALS V: Help when Outdoors

Parameter Name	Beschreibung	Ontologie
User/password	To allow the user to access the application.	–
Basic personal information: address, phone	Basic information useful for the application to know.	Instanz
Subprofile to access the behavior related information	Formal description of behavioral aspects of the user according to his status (mental, physical, social).	Instanz
Health profile	Interesting for the doctor to know more about the overall context of the health status.	Instanz
AAL Space profile	Information about how the devices and placed, nodes and types of nodes, where application parts are running. Important for the technician to access that information.	Instanz

Tabelle 3.25: Wissensobjekte: AALS VI: Long Term behaviour Analyser; Mit – gekennzeichnete Parameter sind nach Meinung der Fokusgruppe 03 nicht als Instanz oder Konzept in die Wissensobjekte aufzunehmen.

Parameter Name	Beschreibung	Ontologie
Name	In order to personalize the messages published on FB.	Instanz
Facebook account details	In order to connect to FB and publish the punctuation messages	–

Tabelle 3.26: Wissensobjekte: AALS VII: AALfficiency; Mit – gekennzeichnete Parameter sind nach Meinung der Fokusgruppe 03 nicht als Instanz oder Konzept in die Wissensobjekte aufzunehmen.

Parameter Name	Beschreibung	Ontologie
Name	It is needed the name of the AP, in order the shopping list to be personalized towards his/her needs.	Instanz
Surname	For the same reason as above.	Instanz
Address	The home address of the AP; in order to be delivered shopping from the supermarket to AP's home.	Instanz

Tabelle 3.27: Wissensobjekte: AALS VIII: Food and Shopping

Parameter Name	Beschreibung	Ontologie
Name	Because I want to personalize messages and then I need his name. To put his name in the SMS when an emergency is sent.	Instanz
The time the user usually wakes up.	It is important to compare with the current wake up time, to check for changes and trends in behavior.	Konzept
Health profile	Information about health aspects: frequency in going to bathroom, diseases that should be considered. . .	Konzept

Tabelle 3.28: Wissensobjekte: AALS IX: Personal Safety

Parameter Name	Beschreibung	Ontologie
Name	Because I want to personalize messages and then I need his name. To put his name in the SMS when an emergency is sent.	Instanz
Used dispenser id	In case the service is provided with pill dispenser(s), I need to know which one is used by the specific assisted person.	Instanz

Tabelle 3.29: Wissensobjekte: AALS X: Medication Manager

zur Deinstallation in der AAL-Umgebung betrachtet wird. Weiterhin bedarf es der Einbeziehung aller an den AAL-Anwendungsfällen beteiligten Akteure.

Anwendungsfallübergreifende Konfigurationsparameter: Viele Konfigurationsparameter lassen sich im Kontext von anderen Anwendungsfällen wiederverwenden. Eine zentrale Wissensbasis zu Konfigurationsparametern in der AAL-Umgebung hätte mehrere Vorteile wie beispielsweise Skalierbarkeit, Wiederverwendbarkeit, Zeitersparnis, Automatisierbarkeit. Dies ermöglicht eine Nutzung von Software- oder Hardwarekomponenten von mehreren AAL-Anwendungsfällen aus. Hierbei sollte das personalisierte Wissen in der eigenen AAL-Umgebung verbleiben.

Regeln für Konfigurationsparameter: In der Gruppendiskussion ist aufgefallen, dass viele Konfigurationsparameter bestimmten Restriktionen unterliegen. Es sollte die Möglichkeit geben, die Parameter durch Regeln einzuschränken, ihre Instanziierung zu prüfen und den Konfigurationsablauf entsprechend zu beeinflussen.

3.4.3 Zusammenfassung

Mit Hilfe der Fokusgruppe 03 konnten zehn Repräsentanten für AAL-Anwendungsfälle identifiziert werden. Durch ihre Abdeckung aller für AAL wichtigen Anwendungsfallkategorien konnte weiterhin die Allgemeingültigkeit dieser Referenzanwendungsfälle erfolgreich gezeigt werden. Abschließend wurde auf Basis der Referenzanwendungsfälle der Konfigurationsbedarf bestimmt. Hierbei wurden drei Komplexitätsgruppen gefunden: „Einfache“ (atomare Parameter), „Komplexe“ (mehrdimensionale Objekte) und Wissensobjekte (Parameter sind auch für andere Komponenten in der AAL-Umgebung nützlich). Außerdem stellte die Gruppe fest, dass auf eine ganzheitliche Konfiguration zu achten ist, die eigene Regeln und Validierungsmöglichkeiten mitbringen muss.

3.5 Zusammenfassung

Dieses Kapitel beantwortete die Frage nach den Anforderungen an ein flexibles Management von AAL-Umgebungen und die damit verbundene Konfiguration. Es konnten sowohl die verschiedenen beteiligten Akteure sowie ihre Anforderungen identifiziert werden als auch der Konfigurationsbedarf von AAL-Plattformen und AAL-Anwendungsfällen.

Es gibt 11 verschiedene Akteurgruppen, die am flexiblen Management beteiligt sind. Zusammen haben sie 63 Anforderungen, die zehn Pflichtanforderungen beinhalten. Die Anforderungen wurden in fünf Kategorien geteilt: Anwendungsfälle, Benutzerschnittstelle, Hardware, Support und Werkzeugunterstützung. Die Beschreibung der Anforderungen ist die Grundlage für die weitere Spezifikation des Konfigurationslifecycle und dem dazugehörigen technischen Anforderungen an das Konfigurationsframework.

Weiterhin wurden in diesem Kapitel vorhandene Plattformen und ihre Eignung als flexible AAL-Umgebung in einer Studie untersucht. Das Ergebnis dieser Analyse zeigt die Bewertung von 43 Plattformen entlang der 63 Anforderungen. Nur zehn

der Plattformen kommen für eine flexible AAL-Umgebung in Frage. Hierbei konnten verschiedene Defizite und bewährte Konzepte herausgearbeitet werden, die in fünf Anforderungen zum Konfigurationsbedarf von Plattformen münden. Es zeigte sich, dass leicht personalisierbare AAL-Anwendungsfälle mit gut anpassbaren Benutzerschnittstellen fehlen. Weiterhin muss der Konfigurationsaspekt schon bei der Anwendungsfallentwicklung beachtet werden und mit geeigneten Werkzeugen die Software- und Hardwareentwicklung unterstützen.

Der dritte Analyseteil widmete sich verschiedenen Referenzanwendungsfällen. Durch die Untersuchung des Konfigurationsbedarfs bei Anwendungsfällen, die sich in der AAL-Domäne bereits bewiesen haben, entstand ein klarer Eindruck, wo und von wem in welcher Mächtigkeit Konfiguration von AAL-Anwendungsfälle benötigt wird, um die angestrebte Flexibilität zu erreichen.

Insgesamt lassen sich aus den vier Bereichen (Akteure, Anforderungen, AAL-Plattformen und AAL-Anwendungsfälle) vier Grundsätze für flexibles Management von AAL-Umgebungen ableiten.

Einfache Integration von neuer Funktionalität: Es muss einen Prozess geben, um einfach neue Funktionen zur Bedürfnisbefriedigung der verschiedenen Akteure in einer AAL-Umgebung einzubringen.

Sammlung und Bereitstellung von Wissen über die AAL-Umgebung:

Das Wissen über die AAL-Umgebung mit ihren Akteuren und verschiedenen Komponenten (z.B. Hardware, Anwendungsfälle, Benutzerprofile) muss in geeigneter Form eingesammelt und bei Bedarf durch definierte Schnittstellen zur Verfügung gestellt werden.

Wartbarkeit der Umgebung: Soll eine Umgebung sich auf aktuelle und neue Anforderungen einstellen, bedarf man Informationen über den Zustand der AAL-Umgebung. Diese Informationen dienen der Wartbarkeit und somit dem flexiblen Reagieren auf Umgebungsveränderungen.

Anpassung an den Endbenutzer: Die AAL-Umgebung soll per Definition das Leben der Endbenutzer, im speziellen des Bewohners, vereinfachen, indem seine Bedürfnisse bedient werden. Alle Bestandteile der Umgebung müssen sich somit an den Endbenutzer anpassen lassen.

Ausgerichtet an diesen vier Grundsätzen und untermauert durch die Erkenntnisse der verschiedenen Analysen, folgt im nächsten Kapitel der Entwurf des Konfigurationsframeworks.

4. Konzeption des Konfigurationsframeworks

In diesem Kapitel wird ein Framework zur Konfiguration als Lösung zum flexiblen Management von AAL-Umgebungen konzeptionell erarbeitet. Wie in der DSR-Methode (siehe Kapitel 2.2) beschrieben, werden auf Basis der verschiedenen Analysen, die benötigten Prozesse und Komponenten in Zusammenarbeit mit den Fokusgruppen 04 und 06 definiert. Auf diese Weise müssen Akteure aus Kapitel 3.1 mit zusätzlichen Aufgaben im Rahmen der Konfiguration betraut werden. Weiterhin sind ihre Anforderungen (siehe Kapitel 3.2) an den entsprechenden Stellen in die Konzeption des Frameworks eingeflossen. Die Ergebnisse zum Konfigurationsbedarf der Plattformen (siehe Kapitel 3.3) und AAL-Anwendungsfälle (siehe Kapitel 3.4) fließen jetzt direkt in die Konzeption ein.

Die Konzeption hat entsprechend der DSR in zwei Iterationen stattgefunden (siehe Kapitel 2) und wird an dieser Stelle in ihrem finalen Ergebnis präsentiert. Somit ist die Gewichtung und Bewertung der Anforderungen (siehe Kapitel 6.1) auf Basis des ersten Frameworkprototypen in die finale Konzeption eingeflossen.

Die Vereinigung der Analyseergebnisse mit den Erkenntnissen der Anforderungsevaluation sind in Abstimmung mit der Fokusgruppe 04 zu sieben technische Anforderungen zusammengefasst. An ihnen ist die Konzeption ausgerichtet.

- R 1:** Das Framework soll Dienste und Werkzeuge zur Installation neuer AAL-Anwendungsfälle in der AAL-Umgebung bereitstellen.
- R 2:** Das Framework soll Dienste und Werkzeuge zur Personalisierung neuer AAL-Anwendungsfälle in der AAL-Umgebung bereitstellen. Diese Anpassungen sollen nach der Initialisierung des AAL-Anwendungsfalls zu jedem Zeitpunkt möglich sein.
- R 3:** Dem Framework soll eine einheitliche Konfigurationssprache zu Grunde liegen, die alle beteiligten Komponenten und Akteure des Frameworks nutzen.

- R 4:** Das Framework soll den Austausch von Verbrauchsgütern oder defekten Komponenten in der AAL-Umgebung unterstützen.
- R 5:** Das Framework soll das Updaten und Deinstallieren von AAL-Anwendungsfällen unterstützen.
- R 6:** Das Framework soll mit der Middleware kommunizieren, um Informationen in der AAL-Umgebung bereitzustellen oder aus ihr zu beziehen.
- R 7:** Das Framework soll verschiedene Akteure mit verschiedenen Rechten unterstützen.

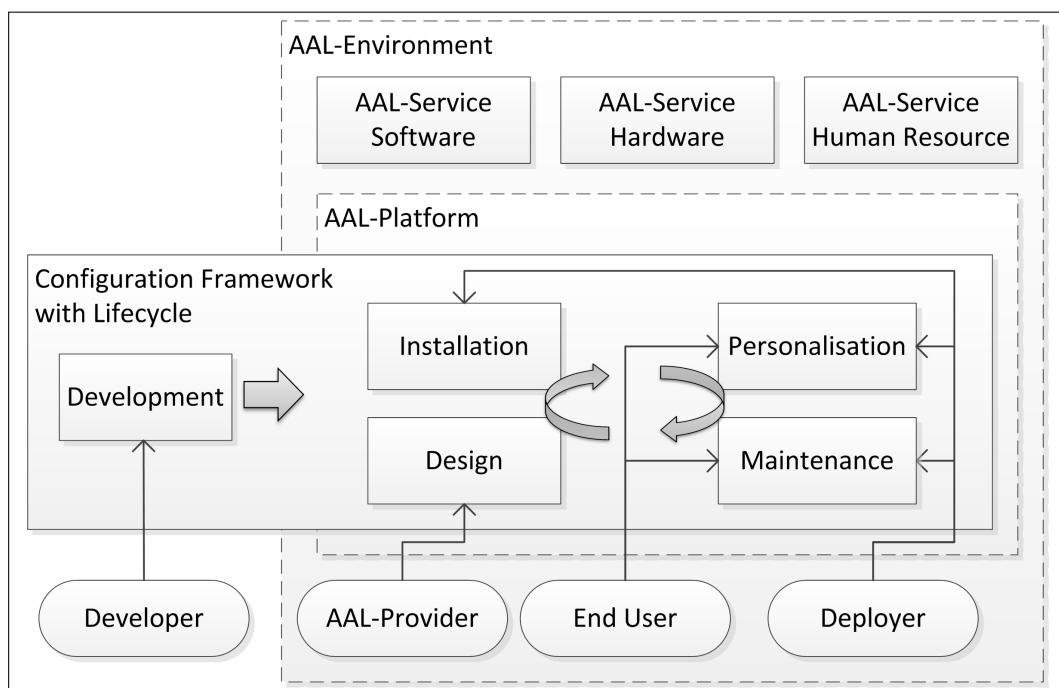


Abbildung 4.1: Überblick zum Konfigurationsframework und den beteiligten Komponenten (eckiger Kasten) und Akteuren (runder Kasten) sowie den Beziehungen dazwischen.

Das flexible Management der AAL-Umgebung soll, wie in der Methodik (siehe Kapitel 2) beschrieben, durch die Konzeption eines Konfigurationsframeworks erreicht werden. Die Konfiguration einer derart komplexen Umgebung kann aber nur gelingen, wenn alle relevanten Komponenten und Akteure miteinbezogen werden. Hierzu wird ein Lifecycle als Teil des Frameworks entwickelt, der den AAL-Anwendungsfall und seine Integration als implementierter AAL-Service in die AAL-Plattform begleitet.

Ein Überblick ist in Abbildung 4.1 zu finden. Man sieht, in welcher Relation das Konfigurationsframework (*Configuration Framework*) zu den Komponenten der AAL-Umgebung (*AAL-Environment*) steht. Dieses Framework stellt Schnittstellen zu den Akteuren und der Middleware sowie die fünf Komponenten des Lifecycles und die damit verbundenen Prozesse bereit. Bis auf die Entwicklung läuft das Framework zum großen Teil auf der AAL-Plattform (*AAL-Plattform*) und wird von Akteuren

in der AAL-Umgebung genutzt. Die Kommunikation zu den AAL-Services mit ihren drei Komponenten (Software, Hardware und menschliche Dienstleistungen (*Human Resource*)) geschieht hierbei auch über die AAL-Plattform innerhalb der AAL-Umgebung.

Die zentrale Komponente des Konfigurationsframeworks ist der Lifecycle. Dieses Kapitel beschreibt, wie nach der Analyse die einzelnen Komponenten des Lifecycles entwickelt wurden, die jeder AAL-Service (siehe Kapitel 4.3) durchläuft. Dieser besteht aus fünf Phasen (siehe Abbildung 4.1, eckige Kästen im Framework), welche in Kapitel 4.1 im Einzelnen beschrieben sind. Beginnend mit der Entwicklung durchläuft den Lifecycle jeder AAL-Service über die Planung, Installation, Personalisierung und Wartung, bis er irgendwann wieder deinstalliert wird. Jede Phase im Konfigurationslifecycle ist Akteuren (siehe Kapitel 4.2 und Abbildung 4.1, runde Kästen) zugeordnet, die daran partizipieren. Hierbei befinden sich nicht alle Komponenten und Beteiligte in der AAL-Umgebung oder nutzen die AAL-Plattform. Kapitel 4.3 geht vertieft auf die technischen Anforderungen im Konfigurationsframework ein und stellt die Konzepte dahinter vor. Abschließend werden in Kapitel 4.4 die Konfigurationsontologien besprochen, die eine Verteilung des Konfigurationswissens in der Plattform und zwischen den verschiedenen AAL-Services zulassen.

4.1 Der Konfigurationslifecycle

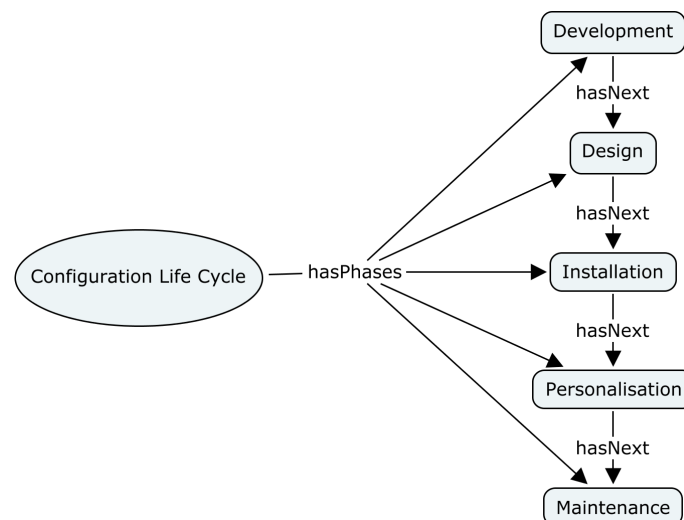


Abbildung 4.2: Überblick zum Konfigurationslifecycle

Basierend auf den Ergebnissen der Analyse der Plattformkandidaten und der Anforderungen der Akteure ist ein Prozess entwickelt worden, um AAL-Services in die AAL-Umgebung unter Berücksichtigung der notwendigen Flexibilität zu integrieren. Das Ergebnis dieser Prozessentwicklung ist der Konfigurationslifecycle. Dieser orientiert sich an dem Service Lifecycle-Modell für SOA von Gu und Lago [GuLa07] und erweitert es um Konfigurationsaspekte. In Abbildung 4.1 ist jede Phase des Lifecycles mit den jeweiligen Akteuren verknüpft. Entsprechend der Methodik ist der Lifecycle (*Configuration Life Cycle*) in Abbildung 4.2 in einer Concept Map dargestellt. Hierbei wurden die Ergebnisse und einzelnen Schritte von den Fokusgruppen 04 und 06 begleitet und validiert. Wie die gesamte Konzeptionsphase wurde auch

der Konfigurationslifecycle in zwei Iterationen erarbeitet. Er besteht aus den fünf Phasen: Entwicklung (*Development*), Planung (*Design*), Installation (*Installation*), Personalisierung (*Personalisation*) und Wartung (*Maintenance*). Jede dieser Phasen zerfällt in verschiedene Schritte, die notwendig sind, um die Konfiguration im Ganzen sicherzustellen. Die einzelnen Teile der Phasen aus Abbildung 4.2 werden in den folgenden Unterkapiteln ausführlich dargestellt.

4.1.1 Entwicklung

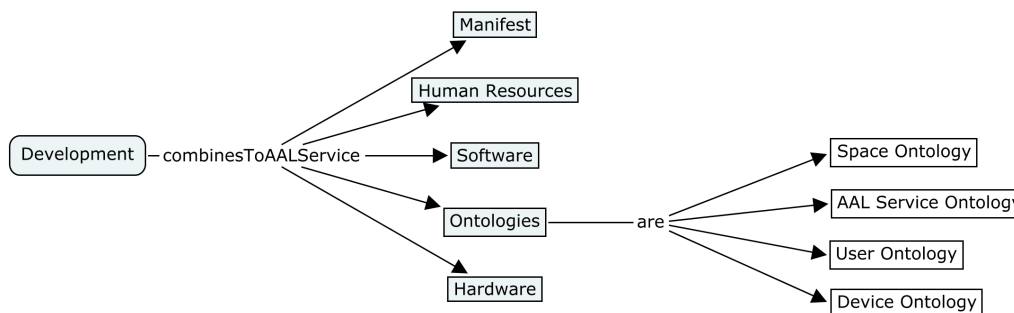


Abbildung 4.3: Überblick zur Entwicklung und ihren Komponenten

Die Entwicklung (siehe Abbildung 4.3) beschäftigt sich mit der Erstellung der AAL-Services und den verschiedenen ihm immanenten Komponenten. Während der Konzeption des AAL-Service in der Entwicklung sollten bereits die zu konfigurierenden Parameter festgelegt werden. Wie im weiteren Verlauf der Implementierung mit ihnen umgegangen wird, hängt von ihrem Gebrauch innerhalb des AAL-Service ab. Es existieren drei grundlegende Funktionsarten, die für Software, Hardware und menschliche Dienstleistungen (Human Resources) genutzt werden. Zur Organisation dieser Komponenten und der damit in Beziehung stehenden Konfigurationen bedarf es zusätzlich eines Manifestes, das in einer strukturierten Form alle zentralen Metainformationen enthält.

Konfigurationsparameter. Diese Art von Parameter wird direkt vom AAL-Service und seinen Komponenten (Software, Hardware und menschliche Dienstleistungen (Human Resources) verwendet und erlaubt die Anpassung des AAL-Service an die AAL-Umgebung. Konfigurationsparameter sind nur innerhalb dieses AAL-Service nutzbar. Im Zuge von Updates des AAL-Service können diese Parameter weiter genutzt werden. Konfigurationsparameter können nicht serviceübergreifend in der AAL-Plattform genutzt werden. Sie stehen nur dem zugehörigen AAL-Service zur Verfügung. Die einzelnen Konfigurationsparameter eines AAL-Service können nach den Wünschen des Entwicklers validiert werden. Weiterhin können Aktionen bei ihrer Instanziierung implementiert werden. Diese Konfigurationsparameter werden in der Softwarekomponente des AAL-Services abgelegt und beinhalten auch konfigurationsrelevante Teile der Hardware und Human Resources.

Wissensobjekte. Sie werden als Objekte mit Relationen zueinander als Ontologie in der Middleware zur Verfügung gestellt. In Abbildung 4.3 sind sie aufgeführt und werden in Kapitel 4.4 genauer vorgestellt. Im Gegensatz zu den

Konfigurationsparametern stehen diese Informationen allen Komponenten der AAL-Umgebung zur Verfügung. Es können sowohl neue Domänenontologien in die AAL-Plattform eingebracht werden als auch vorhandene, entsprechend der Anforderungen des AAL-Service, instanziiert werden. Diese Art der Konfiguration wird somit oft für die Hardwarebeschreibung oder andere Komponenten des AAL-Service verwendet, deren Wiederverwendung explizit gewünscht ist oder sogar im Sinne der SOA ihr primäres Ziel darstellt.

Metadaten. Sie werden in der Entwicklung erstellt und in den späteren Phasen mit allen notwendigen Informationen zur Konfiguration angereichert. Sie werden in Form eines Manifestes mit dem AAL-Service ausgeliefert.

Sind die Konfigurationsparameter und Wissensobjekte festgelegt und den Kategorien zugeordnet, werden sie entsprechend ihrer Kategorie persistiert und mit dem AAL-Service über das Manifest verknüpft. Dies muss durch verschiedene Werkzeuge unterstützt und automatisiert werden.

Human Resource und Hardware beschreiben die echten beteiligten Personen und die verwendete Hardware. Da ihre Abstraktionen zur Konfiguration im Softwareteil oder den Ontologien liegen, sind diese Komponenten nicht weiter zu besprechen.

Im Sinne der Kompatibilität zu SOA-Plattformen reiht sich diese Phase in das „Application design“ und die „Application implementation“ nach [GuLa07] ein und erweitert sie um Konfigurationsaspekte.

4.1.2 Planung

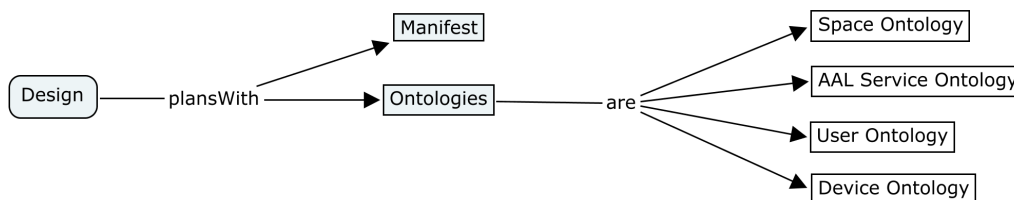


Abbildung 4.4: Überblick zur Planung und ihren Komponenten

Die Planung (siehe Abbildung 4.4) ist einer der wichtigsten Schritte im Konfigurationslifecycle, aus den vorhandenen AAL-Services diejenigen auszuwählen, welche die verschiedenen Bedürfnisse am besten treffen. Die Ontologien unterstützen die Planung indem sie helfen den Auswahlprozess teilweise zu automatisieren. So ist es möglich, unter verschiedenen AAL-Services passende Komponenten zu wählen oder sogar verschiedene AAL-Services zu kombinieren. In diesem Schritt findet der Übergang zwischen der Informationsanreicherung und dem Nutzen der konfigurationsrelevanten Informationen statt. Auf der einen Seite bedarf es bestimmter Informationen über die Bedürfnisse der Bewohner, der Entwicklung und der AAL-Umgebung für die passende Auswahl. Auf der anderen Seite können in der Planung gesammelte Informationen in späteren Phasen genutzt werden. So werden beispielsweise konkrete Sensoren ausgewählt oder ihre Position in der AAL-Umgebung festgelegt. Die wichtigsten semantischen Informationsgruppen der Planung sind:

AAL-Umgebungsinformationen (*Space Ontology*): Diese Gruppe von Informationen beschreibt die AAL-Umgebung und wird von der Plattform bereitgestellt und verwaltet. An sie werden Konzepte der anderen Ontologien angehängen, sofern diese sich in der AAL-Umgebung befinden. In der Planungsphase gibt sie Auskunft über bereits installierte AAL-Services, Hardware oder verfügbare menschliche Dienstleistungen. Weiterhin gibt es auch die Möglichkeit, andere Objekte der AAL-Umgebung zu modellieren. So ist es beispielsweise möglich, Räume, Türen und ihre Relationen zueinander darzustellen.

AAL-Service Informationen (*AAL Service Ontology*): Sie beschreiben AAL-Services und umfassen im Wesentlichen vier Teile. Der erste Teil beschreibt Metainformationen (z.B. die Lizenz, allgemeine Einschränkungen oder Abhängigkeiten), die während dieser Phase bereitgestellt, bewertet und weiterverwendet werden. Der zweite Teil beschreibt alle im AAL-Service konfigurierbaren Parameter, die der Entwickler festgelegt hat. Der dritte Teil stellt alle bereits im Service beinhalteten Komponenten dar. Dies bezieht sich auf Hardware, andere Software oder menschliche Dienstleistungen. Der vierte und letzte Teil beschreibt alle unerfüllten Abhängigkeiten des AAL-Services bezüglich Hardware, anderer Dienste oder menschlicher Dienstleistungen.

Während der Planung kann man sowohl auf alle bereits in der Ziel-AAL-Umgebung installierten AAL-Service-Ontologien zugreifen sowie auf potenziell verfügbare Ontologien aus einem Online Repository.

Akteur Informationen (*User Ontology*): Sie ist eine Beschreibung der Personen, die zur AAL-Umgebung gehören bzw. mit ihr interagieren. Von besonderem Interesse ist hier die Gruppe der Bewohner. Zu allen Personen ist ein bestimmter Grundstamm an Informationen in der Umgebung hinterlegt, passend zu ihren Aufgaben und Bedürfnissen.

Hardware Informationen (*Device Ontology*): Diese Ontologie beschreibt Hardware und ermöglicht ihre Nutzung durch AAL-Services in der AAL-Umgebung.

In der Planung hat man sowohl Zugriff auf die bereits installierte Hardware sowie auf potenziell verfügbare Hardware aus einem Online Repository.

Die beschriebenen Informationsquellen helfen bei der Auswahl des richtigen AAL-Services passend zum Unterstützungsbedarf. Wird ein AAL-Service ausgewählt, wird er als erstes gegen die Akteurinformationen abgeglichen. Durch die Beschreibung der Konfigurationsparameter kann leicht festgestellt werden, ob der Service sich ausreichend anpassen lässt oder wo Probleme bestehen. Danach werden die AAL-Service-Informationen mit den AAL-Umgebungsinformationen abgeglichen. Es wird erkannt, ob eine Integration des Services in die Ziel-AAL-Umgebung möglich wäre oder welche Abhängigkeiten nicht aufgelöst werden können. Sollten weitere Abhängigkeiten zur Hardware bestehen, wird versucht, diese mit Hilfe der Hardwareinformation aufzulösen. Abhängigkeiten zu anderen AAL-Services oder menschlichen Dienstleistungen können wiederum über die AAL-Service-Information iterativ aufgelöst werden.

Am Ende dieses automatischen Prozesses stehen meist mehrere AAL-Services zur Auswahl. In der Planung erfassen integrierte Bewertungsfunktionen bestimmte Parameter (z.B. Kosten, Sensorgenauigkeit) und bewerten sie. Auf dieser Basis trifft

der Mensch die finale Entscheidung, welcher AAL-Service integriert werden soll. Das Ergebnis der Planungsphase ist dann ein Report mit einer Aufstellung aller relevanten Komponenten und erfassten Informationen aus der Planung.

Die Designphase erweitert das „Service Design“ nach [GuLa07] um Konfigurationsaspekte.

4.1.3 Installation

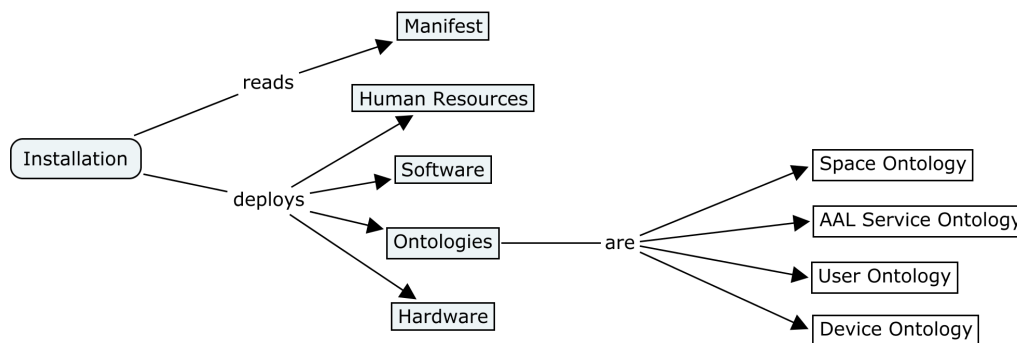


Abbildung 4.5: Überblick zur Installation und ihren Komponenten

Auf Basis des Reports aus der Planungsphase und des Manifests aus dem AAL-Service kann mit der Installation, wie in Abbildung 4.5 abgebildet, begonnen werden. Im Report sind alle benötigten Komponenten verzeichnet, so dass passend die Akteure zur Installation gewählt werden können. Hierbei wird zwischen Soft- und den Hardwareanteil des AAL-Service zu unterscheiden. Die Installation in der AAL-Umgebung ist nur für berechnigte Akteure möglich.

Software/ Ontologien: Im Softwareteil des AAL-Services befinden sich alle weiteren Informationen zur Inbetriebnahme. Ergänzungen aus der Planungsphase werden automatisch in die AAL-Plattform eingespielt. Der AAL-Service beinhaltet ein Manifest, das vor der Installation ausgewertet wird. Hierin befinden sich Informationen, ohne die der Installationsprozess nicht durchgeführt werden könnte. Sie steuern viele Installationsschritte semiautomatisch (z.B. das Nachladen von Abhängigkeiten). Es werden Installationsentscheidungen vorgeschlagen und der Nutzer kann sie, wenn notwendig, beeinflussen. Eine genauere Auflistung aller Parameter im Manifest findet sich in Kapitel 4.3. Da im Manifest alle Abhängigkeiten verzeichnet sind, erfolgt vor der Installation nochmals eine Funktionsprüfung. Anschließend beginnt die Installation aller Softwarekomponenten. Hierbei werden die gegebenenfalls im AAL-Service mitgelieferten Ontologien eingespielt, um die Wissensbasis zu erweitern. Außerdem werden alle Softwarekomponenten des AAL-Services gemäß der Beschreibungen im Manifest installiert. Werden im AAL-Service menschliche Dienstleistungen benötigt, sind diese in den Softwarekomponenten verknüpft und werden in der Personalisierungsphase konfiguriert.

Hardware: Die Hardware wird entsprechend ihren jeweiligen Installationsanweisungen in der AAL-Umgebung installiert. Je nach Installationsanforderungen

der Hardware muss der entsprechende Akteur gewählt werden. An der technischen Installation der Hardware verändert sich nichts. Es können aber zusätzliche Installationsanweisungen aus dem Report der Planungsphase hinzukommen. Diese kommen durch spezielle Anforderungen an die Hardware im Zuge der Nutzung mit dem AAL-Service zustande (z.B. vorgegebene Sensorposition). Die komplette Installation der Hardware ist abgeschlossen, wenn auch ihre Softwarekomponenten installiert sind (z.B. Treiber). Dies geschieht in Form einer AAL-Service-Softwarekomponente.

Menschliche Dienstleistung (*Human Resources*): Sie verlangen oft nach einer Schnittstelle für die Mensch-Maschine-Kommunikation. Diese wird über einen AAL-Service installiert und beinhaltet auch die Informationen für die Personalisierung der Dienstleistung in der nächsten Phase.

Die Installation des AAL-Services in der AAL-Umgebung funktioniert semiautomatisch, da die AAL-serviceeigenen Komponenten so programmiert wurden, dass sie sich flexibel an die Plattform und Umgebung anpassen können. Weitere installationsrelevante, zur Entwicklungsphase unbekannt Informationen wurden in der Planung erfasst. Diese werden als Report für die Installation bereitgestellt. Das Ergebnis dieser Phase ist ein in seiner Standardkonfiguration laufender Service. Die Installationsphase ist mehreren Aktivitäten [GuLa07] zuzuordnen: „Service provision“, „Service discover“, „Service orchestration“, „Service negotiation“ und „Application invocation“.

4.1.4 Personalisierung

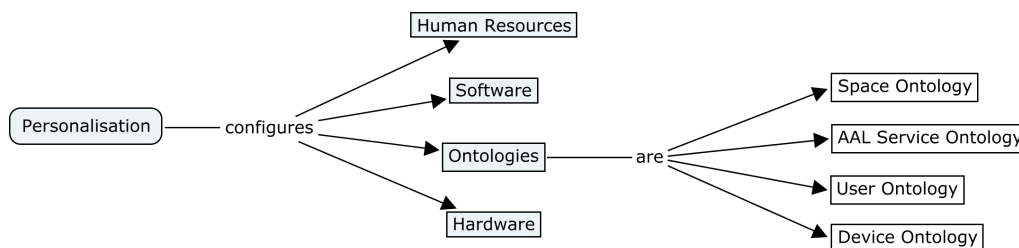


Abbildung 4.6: Überblick zur Personalisierung und ihren Komponenten

Die Personalisierung (siehe Abbildung 4.6) ist die wichtigste Phase im Konfigurationslifecycle. Sie ist den berechtigten Akteuren vorbehalten. Nach der Installation wird der laufende AAL-Service personalisiert und so an den Benutzer und die AAL-Umgebung angepasst. Dies betrifft alle Komponenten des AAL-Services. Jede Komponente kann hierzu auf zweierlei Weise editiert werden. Zum einen über die Anpassung von Konfigurationsparametern und zum anderen über Wissensobjekte. Für jede Personalisierung wird stets eine Standardbelegung durch den AAL-Service mitgeliefert, die eine rudimentäre Lauffähigkeit sicherstellt. Teile der Personalisierung können automatisiert werden, da bereits Instanziierungen für ihre Parameter aus anderen AAL-Services in der AAL-Umgebung bekannt sind.

Software/ Ontologien: Die Einrichtung der Software des AAL-Service geschieht über eine Konfigurationsdatei, welche die zu setzenden Parameter beschreibt.

Hierbei ist eine entsprechende Hilfestellung für jeden Parameter gegeben und es erfolgt eine Überprüfung der Eingabe. Weiterhin können, abhängig von der gewählten Eingabe für den Parameter, weitere Aktionen ausgelöst werden. Die Parameter selber können sich auf einfache Eingaben beziehen oder auf bereits in der AAL-Umgebung vorhandene Ontologiekonzepte verweisen. So können Hardware und menschliche Dienstleistungen in einem AAL-Service verwendet werden. Diese Zuordnung kann aufgrund der semantischen Beschreibung beider Komponenten oft automatisch geschehen. Weiterhin ist es möglich, die im Installationsschritt eingespielten Ontologien zu instanziiieren.

Hardware: Zur Personalisierung von Hardware in der AAL-Umgebung gehört meist eine Softwarekomponente, welche die Anbindung (z.B. Treiber) an die Plattform realisiert. Diese Software wird als AAL-Service ausgeliefert und wie im Abschnitt Software/ Ontologien personalisiert.

Ist keine Software zu der Hardware installiert worden, da beispielsweise schon alle Treiber vorhanden sind, kann die Hardware direkt in der Ontologie bekannt gemacht werden. Hierzu werden Instanzen der Ontologiekonzepte angelegt, welche die Hardware repräsentieren und ihre Nutzung in der AAL-Umgebung ermöglichen. Die genauen Anforderungen an diese Hardwareontologien sind in Kapitel 4.4 zu finden.

Menschliche Dienstleistung (*Human Resources*): Bei den menschlichen Dienstleistungen verhält es sich ähnlich wie bei der Hardware. Man braucht meist eine Softwarekomponente, um Akteure mit ihren Dienstleistungen in der AAL-Umgebung repräsentieren zu können. Hierfür wird ein AAL-Service genutzt und die Personalisierung desselben kann im Abschnitt Software/ Ontologien vollzogen werden.

Weiterhin ist es wie bei der Hardware möglich, auch direkt neue Akteure in der AAL-Umgebung bekannt zu machen. Hierzu müssen die notwendigen Ontologien bereits existieren. Die entsprechenden Konzepte werden dann in der Personalisierungsphase instanziiert.

Nach Abschluss dieser Phase ist der zuvor installierte AAL-Service an die AAL-Umgebung und ihre Akteure angepasst und kann in seiner personalisierten Form genutzt werden. Ein Großteil der Personalisierung kann aufgrund der semantischen Modellierung und durch vorab in die Umgebung eingebrachte Informationen anderer AAL-Services automatisch vorgenommen werden. Der Lifecycle nach [GuLa07] erweitert sich in dem Aspekt der „Service composition“ um Konfigurationsaspekte.

4.1.5 Wartung

Die Wartung ist die letzte Phase des Konfigurationslifecycle. Auch in dieser Phase agieren nur autorisierte Akteure. In der Wartung werden alle im System laufenden *AAL-Services* überwacht und bei Bedarf wird auf Fehlerzustände reagiert (siehe Abbildung 4.7). Zur Überwachung gehört in erster Linie ein Überblick zu allen AAL-Servicekomponenten und ihrer korrekten Funktionsweise. Im Problemfall kann eine Komponente oder ein ganzer AAL-Service editiert werden. Durch Notfallkonfigurationen und Standardeinstellungen ist dies auch vollautomatisch möglich. Natürlich muss nicht erst ein Problem virulent werden, bevor man aus der Wartung heraus

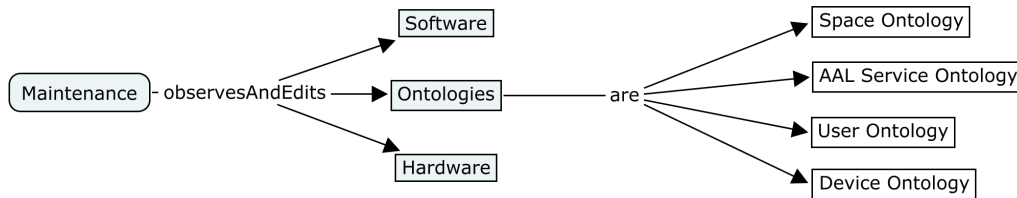


Abbildung 4.7: Überblick zur Wartung und ihren Komponenten

eine Anpassung der Personalisierung der AAL-Umgebung vornimmt. So besteht immer die Möglichkeit, bereits installierte AAL-Services während der Laufzeit neu zu personalisieren und so an neue Bedürfnisse anzupassen. Auch die Anpassung eines alten AAL-Services aufgrund neuer in der Umgebung installierter AAL-Services kann sinnvoll sein. So kann ein alter AAL-Service leicht neue Software, Hardware oder menschliche Dienstleistungen nutzen. Auch die Änderung und Wartung der Ontologieinstanzen ist in dieser Phase angesiedelt. Die Wartungsphase eines AAL-Services kann verlassen werden, indem man den Service deinstalliert.

Die Wartung bietet somit einen kompletten Überblick der AAL-Umgebung und ihrer Funktionen. Sie ist die letzte Phase des Lifecycles und greift vielfach für ihre Aufgaben auf Informationen aus den vorherigen Phasen zurück. Im Sinne der Kompatibilität reiht sich diese Phase in das „Application maintenance“ nach [GuLa07] ein und erweitert sie um Konfigurationsaspekte.

4.2 Akteure des Konfigurationslifecycles

In diesem Kapitel werden die Akteure beschrieben, welche an den verschiedenen Phasen des Konfigurationslifecycles partizipieren. Die im Analysekapitel 3.1 identifizierten Akteure einer AAL-Umgebung bieten eine gute Grundlage, um die konfigurationsrelevanten Rollen unter ihnen zu identifizieren. Vier der fünf gefundenen Akteure lassen sich den entwickelten Lifecyclephasen zuordnen. Der Assistenzanbieter profitiert zwar von einer flexiblen AAL-Umgebung, ist aber nicht an der Konfiguration beteiligt. Nicht alle Akteure können hierbei ihre Rollenbeschreibung beibehalten. Teilweise bedarf es einer Anpassung der Rollenbeschreibung und ihrer Aufgabefelder. Bevor die einzelnen Akteure, unter Berücksichtigung des Konfigurationslifecycles, genauer beschrieben werden, soll ein Überblick zu ihrer Phasenzuordnung gegeben werden. Abbildung 4.8 zeigt im oberen Teil der Grafik die fünf Phasen und darunter die Akteure mit ihren dazugehörigen Gruppen. Die Überschneidung einzelner Akteure bei den Phasenübergängen deutet hierbei eine Interaktion zwischen den verschiedenen Gruppen an.

Die beiden Rollen der **Entwickler** sind im Entwicklungsteil (siehe Kapitel 4.1.1) des Lifecycles angesiedelt und übernehmen anschließend keine Aufgaben mehr. Sie sind die Gruppe, welche die anderen Phasen des Lifecycles unterstützt und selber nur sekundär von der Konfigurierbarkeit ihrer AAL-Services profitiert.

Software-Entwickler: Er setzt die Basis für die Konfigurierbarkeit der AAL-Services. Er entscheidet, welche Parameter sich in den folgenden Phasen des Lifecycles anpassen lassen. Hierbei obliegt ihm die Aufgabe zu wählen, ob Parameter nur für einen Service als Konfigurationen abgelegt werden oder ob

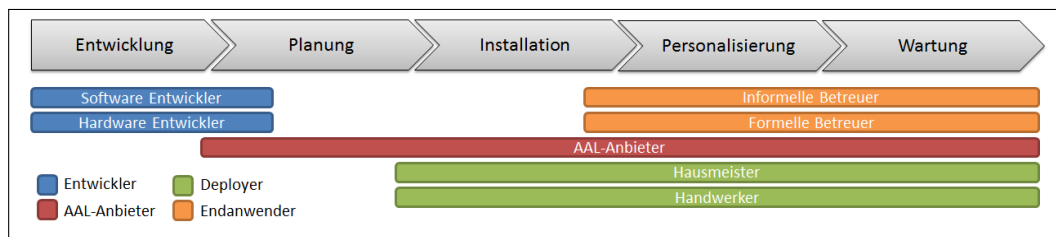


Abbildung 4.8: Übersicht zu den einzelnen Akteuren in Relation zum Konfigurationslifecycle.

sie in einer Ontologie gespeichert werden und somit im gesamten System zur Verfügung stehen. Weiterhin kann er für die folgenden Phasen der Installation und Personalisierung Regeln definieren, die den jeweiligen Nutzern ihre Arbeit erleichtern. Diese Regeln zwischen Konfigurationsparametern ermöglichen eine direkte Reaktion auf Veränderungen der Konfiguration. Außerdem können über Restriktionen die Konfigurationseinstellungen in den späteren Phasen kontrolliert werden. Hierzu programmiert der Entwickler Validatoren. Nach Fertigstellung des AAL-Services wird eine entsprechende Softwarekomponente erzeugt und eine passende Manifestdatei generiert.

All diese Schritte bedeuten für den Entwickler erstmal einen Mehraufwand, der durch Werkzeuge, die ihm zur Verfügung gestellt werden, gemindert werden soll. Sekundär wird er aber von der Konfigurierbarkeit seines erzeugten AAL-Services auch profitieren. Durch die definierte Konfiguration ist eine vielfältigere und flexiblere Nutzung und die Wiederverwendung seines AAL-Services durch andere AAL-Services möglich.

Hardware-Entwickler: Er implementiert Hardwarekomponenten für die Plattform. Durch die geforderte Entkopplung der Hardware in abstrakte Schnittstellen kann sie unabhängig von Typ und Hersteller angeschlossen werden. Sollte dennoch Software (z.B. Treiber) notwendig werden, ist die Entwicklung und Bereitstellung Aufgabe des Softwareentwicklers. Dennoch benötigt die Konfiguration vom Hardwareentwickler eine semantische Darstellung der Hardware, die gewissen Regeln entspricht. Nur so kann sie in der Plattform konfiguriert und serviceübergreifend genutzt werden.

Hierfür wurden für den Hardwarehersteller auch Werkzeuge entwickelt und bereitgestellt. Insgesamt profitiert er vom Mehraufwand der semantischen Darstellung seiner Hardware durch vielfältigere Anwendungsmöglichkeiten und den damit verbundenen möglichen höheren Absatz.

AAL-Anbieter: Sie begleiten einen großen Teil des Lifecycle. Aus Sicht der Konfiguration sind sie Case Manager. Sie übernehmen die fertigen AAL-Services von den Entwicklern, um damit in der Planung der AAL-Umgebung die Bedürfnisse des Endanwenders und im Besonderen des Bewohners zu befriedigen. In der Planungsphase nutzen die AAL-Anbieter die semantische Konfiguration, um geeignete Lösungen für die geplante AAL-Umgebung zu erstellen. Der AAL-Anbieter wird in diesem Schritt durch geeignete Werkzeuge unterstützt. Nach Abschluss der Planungsphase übergibt er an den Hausmeister oder wenn nötig an weitere Handwerker. Er selbst nimmt nun

bis zum Ende des Lifecycles eine koordinierende Rolle zwischen den Endanwendern und den Deployern ein.

Deployer: Sie sind mit zwei Rollen am Konfigurationslifecycle beteiligt und integrieren die AAL-Services in die AAL-Umgebung. Aus der Planungsphase erfahren sie über den AAL-Anbieter, welche Services wie installiert werden sollen. Nach der erfolgreichen Installation werden sie den Service entsprechend der in der Planung erfassten Anforderungen personalisieren. Hierbei können sie auch auf die Gruppe der Betreuer zurückgreifen. Sollte der AAL-Anbieter in seiner koordinierenden Funktion nicht als Vertrauensperson mit vor Ort sein, sind bei der Installation Betreuer anwesend. Auch in der Wartung können, je nach Anforderungen, die Dienste des Hausmeisters oder Handwerkers nötig werden. So ist die Funktion des Systems zu überwachen und gegebenenfalls auf Unregelmäßigkeiten geeignet zu reagieren. Für jeden der drei Schritte, an denen der Deployer beteiligt ist, müssen geeignete Werkzeuge zur Verfügung stehen.

Hausmeister: Er hat ein solides technisches Grundwissen und ist die erste Instanz, die bei technischen Problemen von den verschiedenen anderen Rollen zu Hilfe gerufen werden kann. Er ist kurzfristig erreichbar und kann viele einfache Probleme lösen. Bei Bedarf wird er in Abstimmung mit dem AAL-Anbieter einen passenden Handwerker hinzurufen.

Hardwareentwickler: Er ist ein Fachmann, der entsprechend seinem Fachgebiet bestimmte technische Aufgaben in der AAL-Umgebung übernimmt.

Betreuer: Er nimmt im Konfigurationslifecycle eher eine untergeordnete Rolle ein. Dennoch hat er die Möglichkeit, einfache Personalisierungsarbeiten vorzunehmen. Dies ist der Fall, wenn der AAL-Service bereits in der Umgebung installiert und personalisiert ist und nachfolgend kleinere Anpassungen notwendig werden. Nicht immer muss hierbei der Hausmeister hinzugezogen werden, wenn es sich um einfache Anpassungen (z.B. neue Handynummer oder medizinische Profile aktualisieren) handelt. Auch einfache Wartungsarbeiten (z.B. Batterie wechseln) können vom Betreuer in seiner täglichen Arbeit mit erledigt werden. Die Unterteilung in formelle und informelle Betreuer spielt für die damit verbundenen Aufgaben im Konfigurationslifecycle bis auf rechtliche Aspekte keine Rolle.

Falls ein Bewohner aktiv am Konfigurationslifecycle teilnimmt, handelt er als Akteur einer anderen Gruppe. Sollte er beispielsweise Wartungsaufgaben übernehmen, so nimmt er die Rolle des Hausmeisters ein.

4.3 Konfigurierbare AAL-Services

Dieses Kapitel beschäftigt sich mit den notwendigen Anpassungen des AAL-Services, um ein flexibles Management zu erlauben. Die Definition eines AAL-Services (vgl. Abbildung 1.1) beinhaltet hierbei drei Teile, die berücksichtigt werden müssen, um den gesamten Service konfigurierbar zu machen. Nur so kann sichergestellt werden, dass in allen Phasen des Konfigurationslifecycles (siehe Kapitel 4.1) die nötige Flexibilität zur Verfügung steht.

Es ist wichtig, die verschiedenen am Lifecycle beteiligten Akteure konsequent zu berücksichtigen. In den ersten beiden Phasen des Lifecycles muss der AAL-Service konfigurierbar gemacht werden. Durch den Entwickler werden *Konfigurationsdefinitionen* passend zum AAL-Service implementiert und durch den AAL-Anbieter werden Metainformationen erfasst. Sie beinhalten *Konfigurationsoptionen* und *Konfigurationsregeln*. Die mit dem Service ausgelieferte Konfigurationsdefinition wird vom Deployer verwendet, um passende *Konfigurationsinstanzen* anzulegen.

Damit Anwendungsfälle sich flexibel in die AAL-Plattform integrieren lassen, müssen sie vom Entwickler entsprechend angepasst werden. Hierzu erstellt der Entwickler eine Konfigurationsdefinition, die beschreibt, welche Komponenten des AAL-Services wie konfigurierbar sind. Diese Konfigurationsdefinition beinhaltet je nach Komplexität des AAL-Services verschiedene Konfigurationsoptionen und -regeln. Die Konfigurationsoptionen beschreiben die einzustellenden Parameter des AAL-Services. Die Regeln definieren das Verhalten der Konfigurationsoptionen zueinander. Dies unterstützt den Deployer bei der Personalisierung des AAL-Service. Das Ergebnis der Personalisierung auf Basis der Konfigurationsdefinition sind Konfigurationsinstanzen.

Aufbauend aus den im Analysekapitel erfassten Anforderungen der verschiedenen AAL-Services lässt sich der Konfigurationsbedarf in Komplexitätsklassen (siehe Kapitel 4.3.1) einteilen. Die Referenzanwendungsfälle lassen sich diesen Klassen zuordnen. Die in Kapitel 3 erarbeiteten technischen Anforderungen an die Konfigurationsparameter sind von drei Seiten zu bearbeiten, um sie weiter zu verfeinern und zu formalisieren. Diese Seiten sind, wie in der Analyse dargestellt, die Akteure, die AAL-Plattform und die AAL-Services. Es werden die verschiedenen Arten von Konfigurationsoptionen (siehe Kapitel 4.3.2) und die dazugehörigen Regeln (siehe Kapitel 4.3.3) vorgestellt. Beide zusammen erlauben die Abbildung aller Komplexitätsklassen. Zur Nutzung dieser Optionen und Regeln im Lifecycle bedarf es aber der Möglichkeit, sie entsprechend als Konfigurationsinstanzen zu speichern. Dies ist im Kapitel 4.3.4 beschrieben. Als Austauschformat zwischen den verschiedenen Konfigurationen zwischen den Lifecyclephasen dient die Konfigurationsdefinition (siehe Kapitel 4.3.5).

4.3.1 Konfigurationskomplexität

Die untersuchten Anwendungsfälle zeigen unterschiedliche Arten von Komplexitäten und verschiedene Möglichkeiten sie flexibel an die AAL-Umgebung anzupassen. In der Analyse zum Konfigurationsframework wurden zehn ausgewählte Referenzanwendungsfälle genauer analysiert. Jeder dieser Anwendungsfälle hat seine eigenen Parameter. Dennoch konnte in der Konzeption zusammen mit den Entwicklern eine erste Unterteilung in Konfigurationsparameter und Konfigurationswissensobjekte vorgenommen werden. In den Tabellen 3.10-3.19 sind diese Informationen auf der linken Seite der Doppellinie dargestellt. Während der Konzeption wurde der rechte Teil der Tabelle befüllt und daraus die verschiedenen Komplexitätsklassen abgeleitet.

Im Folgenden werden die Komplexitätsklassen K_1, \dots, K_4 (vgl. Abbildung 4.9) der Konfigurationsparameter vorgestellt und anhand der dazugehörigen Referenzanwendungsfälle beispielhaft diskutiert. Es ist zu beachten, dass gilt:

$$K_1 \subset K_2 \subset K_3 \subset K_4$$

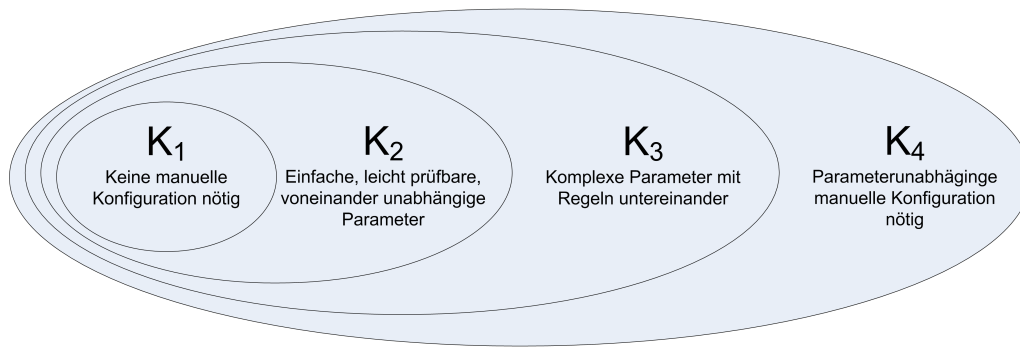


Abbildung 4.9: Übersicht zu den Komplexitätsklassen von AAL-Anwendungsfällen

AALS	Name	Konfigurationskomplexität	Anzahl der Ontologien
I	Health Management	2	5
II	Nutritional Advisor	3	1
III	Agenda and Reminders	4	2
IV	Safety and Security at home	3	1
V	Help when outdoors	2	0
VI	Long Term behavior Analyzer	2	2
VII	AALfficiency	2	3
VIII	Food and Shopping	1	1
IX	Personal Safety	3	0
X	Medication Manager	2	1

Tabelle 4.1: Zuordnung der Referenzanwendungsfälle zu den Komplexitätsklassen und die Anzahl ihrer Ontologien.

Hierbei ist nicht jeder Anwendungsfall eindeutig zuzuordnen, wie wir in den Beispielen zu den einzelnen Kategorien sehen werden. Dennoch lässt sich die Spannweite möglicher Komplexitätsgrade an den Referenzanwendungsfällen gut aufzeigen. Tabelle 4.1 stellt die Zuordnung kumuliert dar. Es sind die Referenzanwendungsfälle in ihrer finalen Version jeweils einer Komplexitätskategorie zugeordnet und auch die Anzahl der von ihnen mitgelieferten Wissensobjekte in Form von Ontologien dargestellt.

Klasse K_1 : Diese Kategorie beinhaltet Anwendungsfälle ohne weiteren Konfigurationsbedarf. Sie bringen im Wesentlichen ihre Standardinformationen mit oder können sie direkt aus der AAL-Umgebung beziehen. Die Konfiguration eines solchen AAL-Services geschieht weitgehend automatisch und wird vom Deployer nur überwacht. Beispiele von Daten, die in der automatischen Konfiguration genutzt werden, sind: Icon, Name, Beschreibung, Autor oder Versionsnummer. Diese Daten sowie weitere Informationen zur Automatisierung sind durch den Entwickler bereitzustellen.

Ein Vertreter dieser Kategorie ist AALS VIII. Dieser AAL-Service benötigt neben Informationen zum Gesundheitsprofil des Bewohners nur Daten (Vor- und Nachname sowie die Adresse des Bewohners), die in der AAL-Umgebung

schon vorhanden sind. Da diese Daten normalerweise beim Installieren der AAL-Plattform mit eingegeben werden, kann der AAL-Service automatisch in Betrieb genommen werden. Natürlich ist es schwer, eine Grenze zu ziehen, welche Daten wirklich in der AAL-Umgebung als bekannt angenommen werden können und welche nicht.

Klasse K_2 : Diese Kategorie umschließt alle Anwendungsfälle, die einfache Konfigurationsmöglichkeiten bieten. Dies sind meist atomare Parameter, die keine komplexe Überprüfung benötigen. Weiterhin gibt es auch keine Regeln zwischen den Parametern, die automatisch aufgelöst werden oder durch den Deployer zu beachten sind.

Viele Anwendungsfälle (AALS I, V, VI, VII, X) fallen in diese Kategorie. Besonders hervorzuheben ist an dieser Stelle Referenzanwendungsfall VI und VII, da sie auf weitere Dateien zugreifen wollen, um den Service betriebsbereit zu machen. Hierfür könnte man sich komplexere Regeln vorstellen. Diese wurden von den Entwicklern nicht bei der Konfiguration gesehen, sondern die Eingabedateien wurden serviceintern geprüft.

Klasse K_3 : Diese Kategorie beinhaltet AAL-Services mit komplexen Konfigurationen. Es existieren Abhängigkeiten zwischen einzelnen Parametern. Weiterhin können komplexere, regelbasierte Einschränkungen zu einzelnen Parametern oder zwischen ihnen existieren. Auch die direkte Beeinflussung des Konfigurationsablaufs ist möglich sowie das Einbinden und Editieren von externen Ressourcen (z.B. Ontologieprofilen) zu Konfigurationszwecken.

Beispiele für diese Art von Anwendungsfällen sind AALS II, IV, IX. Sie beinhalten komplexe Objekte, für die Parameter zu setzen und zu prüfen sind. Ein Beispiel hierfür ist der Parameter „Behave_rules“, der in einer eigenen Beschreibungssprache bestimmte Verhaltensmuster darstellt. Weiterhin beeinflussen sich bestimmte Parametereinstellungen gegenseitig, wie beispielsweise die „Public IP address“ für den Webzugriff weiterhin die Instanziierung des „Web user“ verlangt.

Klasse K_4 : In dieser Menge finden sich Anwendungsfälle, die eine umfassendere Konfiguration benötigen, welche per Hand durchgeführt werden muss. Oft bedeutet diese Kategorie eine Aufteilung in Konfigurationsparameter der Kategorien 1-3 und einzelner Aspekte, die einer speziellen Nachbearbeitung durch den Deployer bedürfen.

Ein Beispiel hierfür ist der Anwendungsfall III, der spezielle Operationen auf dem Dateisystem vom Deployer fordert. Der Parameter „Confadmin“ verlangt hierbei in das richtige Ausführungsverzeichnis kopiert zu werden. Hierbei ist sicherzustellen, dass es sich um das richtige Zielverzeichnis handelt und Menüdateien dort vorliegen.

Orthogonal zu den Komplexitätsklassen werden in den nächsten Kapiteln die Konzepte vorgestellt, um die vier benötigten Arten der Komplexität zu bedienen.

4.3.2 Konfigurationsoptionen

Die Parameter, die es zu konfigurieren gilt, stellen sich in einer großen Breite dar. Es fängt an bei einfachen Zahlenwerten oder Zeichenfolgen bis hin zu Personen

oder Hardware. Zur Abdeckung aller verschiedenen Parameterformen müssen unterschiedliche Konfigurationsoptionen definiert werden. Wie man aus den Referenzanwendungsfällen ableiten kann, sollten die einfachen Typen wie Boolean, Integer, Double und String beherrscht werden. Um auf komplexere Parameter eingehen zu können, sollte es die Auswahl aus einer Liste geben. Für feste Werte bietet sich die Darstellung vom Datentyp Map an. Für eher dynamische Inhalte der Liste sollte eine Abfragesprache unterstützt werden, um verschiedene externe Datenbanken in der Konfiguration dynamisch zu nutzen. Zur Abfrage der Wissensobjekte aus der Ontologie im Zusammenhang mit der Wiederverwendung von Konfigurationsparametern ist eine graphbasierte Abfragesprache notwendig. Hierfür soll eine SQL- und SPARQL¹-Konfigurationsoption vorhanden sein über die Datenbanken und Ontologien direkt in der Konfiguration genutzt werden können.

4.3.3 Konfigurationsregeln

Aus den Referenzanwendungsfällen lassen sich Regeltypen zwischen den Konfigurationsoptionen ableiten, die für die Komplexitätsklassen K_3 und K_4 notwendig sind.

R1 Aktivieren von Konfigurationsoptionen: Es muss möglich sein, eine Konfigurationsoption oder Konfigurationsoptionsgruppe zu aktivieren oder zu deaktivieren. Der Auslöser hierfür ist freigestellt und kann automatisch vom System, von einer anderen Konfigurationsoption oder vom Deployer selbst vorgenommen werden.

So können beispielhaft bei AALS II die Parameter (Server IP, username und password) für den Remotezugang zum Nutritional Web Server erst in der Konfiguration angefordert werden, wenn der Fernzugriff eingeschaltet wird:

$$\text{RemoteZugriff} = \text{true} \Rightarrow \text{aktiviert}(\text{ServerEinstellungen})$$

R2 Limitieren von Konfigurationsoptionen: Damit nur gültige Parameter eingegeben werden können, muss es eine Möglichkeit der Limitierung der Eingabe geben. Weiterhin benötigt eine Fehleingabe eine geeignete Form des Feedbacks an den Deployer. Dies sollte für jeden Typ von Konfigurationsoptionen möglich sein (z.B. Ober- und Untergrenzen). Bei komplexeren Konfigurationsoptionen sind andere geeignete Limitierungen durch den Entwickler zu setzen oder entsprechend aus einfachen Limitierungen zusammenzubauen.

Ein Beispiel ist das Überprüfen der Eingabe der Web Server IP des AALS II Nutritional Advisors gegen einen regulären Ausdruck:

$$\begin{aligned} & \backslash b(25[0 - 5]|2[0 - 4][0 - 9][01]?[0 - 9][0 - 9]?)\backslash. \\ & (25[0 - 5]|2[0 - 4][0 - 9][01]?[0 - 9][0 - 9]?)\backslash. \\ & (25[0 - 5]|2[0 - 4][0 - 9][01]?[0 - 9][0 - 9]?)\backslash. \\ & (25[0 - 5]|2[0 - 4][0 - 9][01]?[0 - 9][0 - 9]?)\backslash b \end{aligned}$$

¹SPARQL 1.1 Overview W3C Recommendation 21 March 2013 <http://www.w3.org/TR/sparql11-overview/> (Zugriff am 21.11.2014)

R3 Konfigurationsübergreifende Regel: Diese Regel soll es ermöglichen, abhängig von der Eingabe für eine Konfigurationsoption, eine andere Konfigurationsoption zu beeinflussen.

Auf das Beispiel des AALS II Nutritional Advisors übertragen heißt das: Wenn im medizinischen Profil eine Krankheit angegeben ist, muss die Ernährung und damit auch der Blutzuckerwert angepasst werden.

$$profile(Diabetes) \Rightarrow Blutzuckerwert < 5,6 \text{ mmol/l}$$

R4 Boolesche Regeln: Diese Art von Regeln erlaubt es, andere Regeln durch boolesche Operatoren zu kombinieren. Die Operatoren werden hierbei auf Konfigurationsoptionen ausgeführt.

So kann das Einschalten des Remotezugriffs des AALS II Nutritional Advisors für die Aktivierung der Felder Server IP und der Zugangsdaten sorgen.

$$RemoteZugriff = true \Rightarrow aktiviert(ServerIP) \& aktiviert(Login)$$

4.3.4 Konfigurationsinstanzen

Damit ein AAL-Service entsprechend der Bedürfnisse des Endanwenders funktioniert, wird er nach der Installation durch den Deployer personalisiert. In der Personalisierungsphase werden hierzu die Konfigurationsoptionen instanziiert. Diese Konfigurationsinstanzen werden in der AAL-Plattform gesichert. Hierfür stehen zwei Möglichkeiten zur Verfügung. Zum einen kann die Speicherung der Konfigurationsinstanzen nur dem dazugehörigen AAL-Service bereitgestellt werden, zum anderen kann sie in der gesamten AAL-Umgebung verfügbar sein. Das Schema für die Konfigurationsinstanzierungen ist in Abbildung 4.10 dargestellt und beinhaltet die notwendige Mächtigkeit für die Abdeckung aller Komplexitätsklassen (siehe Kapitel 4.3.1). Die hervorgehobenen Felder sind hierbei verpflichtend auszufüllen, während alle anderen optional sind. Weiterhin können einige Objekte in einer Instanzbeschreibung nur einmal vorkommen (siehe *hasA*-Beziehung) und andere Objekte hingegen beliebig oft (siehe *has[0..n]*-Beziehung) in einer Konfigurationsinstanz verwendet werden. Die Struktur ist erweiterbar angelegt, um einfach auf neue Anforderungen zu reagieren. In der folgenden Aufzählung sind die einzelnen Objekte der Konfigurationsinstanzen genauer beschrieben.

configurationInstance: Dieses Objekt stellt die Wurzel einer Konfigurationsinstanz (*configurationInstance*) dar und speichert eine Instanziierung der Konfigurationsdefinition eines AAL-Services. Die Konfigurationsinstanz zerfällt hierbei in die folgenden Objekte:

id: Das *id*-Objekt dient zur eindeutigen Identifizierung der Konfigurationsinstanz und ist zwingend erforderlich. Nur so kann sichergestellt werden, dass die passende Personalisierung abfragbar ist. Sollen die Instanzen in der AAL-Umgebung bekannt sein, ist die *id* als URI der entsprechenden Ontologie abzulegen.

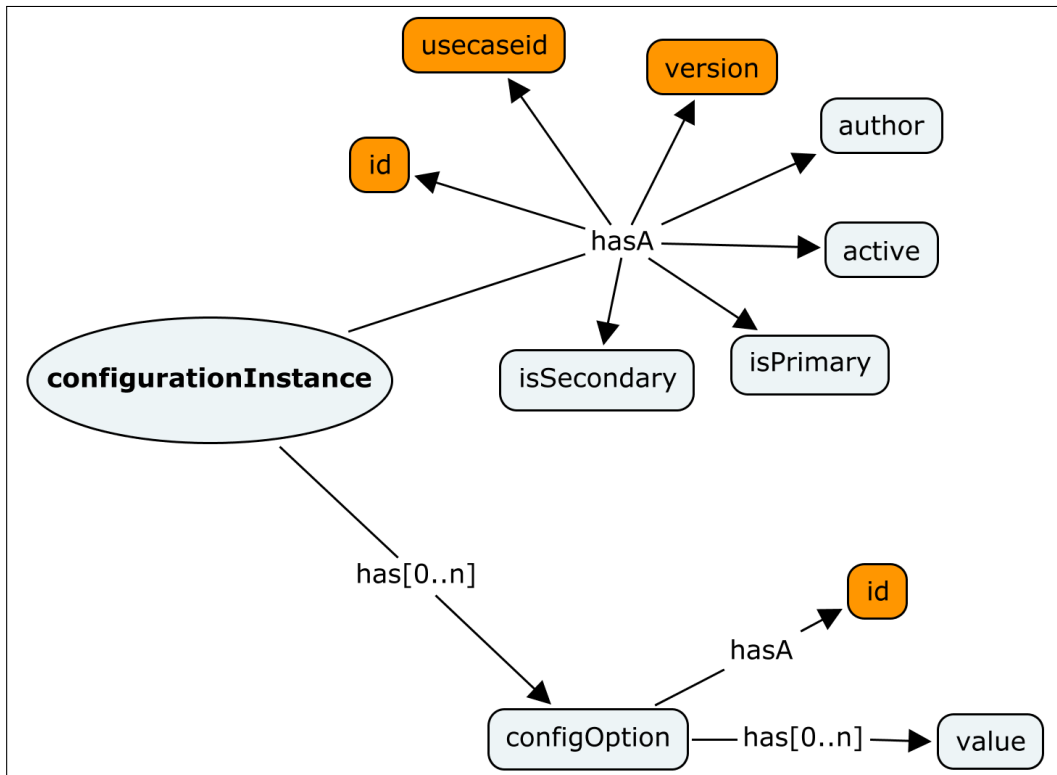


Abbildung 4.10: Überblick über die Konfigurationsinstanzen

usecaseid: Die ID des AAL-Services (*usecaseid*) ist der eindeutige Name des Services, zu dem die Konfigurationsinstanz gehört. Diese ID ist verpflichtend auszufüllen. Es empfiehlt sich, einen eindeutigen Bezeichner zu wählen. Zusammen mit der *id* ergibt sich eine eindeutige Zuordnung der Instanz zu einer AAL-Service-Version. Die *usecaseid* ist besonders wichtig, wenn die Personalisierung nur für den zugehörigen AAL-Service bereitgestellt werden soll.

version: Die Version (*version*) beschreibt eine vom Deployer gesetzte Versionsnummer für die entsprechende Konfigurationsinstanz des AAL-Service. Diese Angabe ist obligatorisch. Sie wird zur Verwaltung verschiedener Konfigurationsinstanzen zu einem AAL-Service verwendet.

author: Der Deployer füllt während der Personalisierungsphase das Objekt des Autors (*author*) aus. Dieses sollte den Namen des Ansprechpartners für diese Konfiguration beinhalten.

active: Das Aktiv-Objekt (*active*) identifiziert bei mehreren Konfigurationsinstanzen die für einen AAL-Service zu nutzende Instanz. Es ist zu beachten, dass immer nur eine Instanz gleichzeitig aktiv sein kann.

isPrimary: Es setzt eine Instanz als primäre Personalisierung (*isPrimary*) für einen AAL-Service. Es kann immer nur eine primäre Konfigurationsinstanz geben.

isSecondary: Es setzt eine Instanz als sekundäre Personalisierung (*isSecondary*) für einen AAL-Service. Diese kann nur einmal existieren und dient als Backup-Personalisierung, falls die primäre Konfigurationsinstanz nicht mehr genutzt werden kann (z.B. Sensorausfall).

configOption: Eine Konfigurationsinstanz kann beliebig viele Konfigurationsoptionen (*configOption*) haben. Diese Konfigurationsoptionen sind der Speicher für die Instanzen der Konfigurationsdefinition.

id: Diese ID dient der eindeutigen Bestimmung der Konfigurationsoption. Sollte die Konfigurationsinstanz nicht nur im zugehörigen AAL-Service genutzt werden, ist die Speicherung einer URI (z.B. Property) erforderlich.

value: Jede Konfigurationsoption kann beliebig viele Werte (*value*) speichern.

In Anlehnung an den in Kapitel 1 eingeführte Beispielanwendungsfall für Menschen mit Demenz sollen einige Konfigurationsinstanzen beschrieben werden. Der AAL-Anwendungsfall unterstützt bei der medizinisch empfohlenen Ernährung. Hierzu hat er unter anderem eine Erinnerungsfunktion, die konfiguriert werden kann. Es kann ein Timeout für die Erinnerung gesetzt und eine zu informierende Person gewählt werden. Hierzu werden zwei Konfigurationsoptionen entsprechend der Konfigurationsdefinition mit Instanzen belegt.

Timeout: Der Timeout wird als *configOption* gespeichert. Hierzu erhält er eine *id* (z.B. *timeout*) die zusammen mit der *usecaseid* eindeutig ist. Außerdem enthält die *configOption* den *value* (z.B. 30) der die Timeout-Zeit speichert.

Person: Die Person wird als *configOption* gespeichert. Sie erhält eine zusammen mit der *usecaseid* eindeutige *id* (z.B. <http://ontology.universAAL.org/Profile.owl#AssistedPerson>). In dem *value* der *configOption* wird diesmal die zu informierende Person abgelegt (z.B. Opa Peter).

4.3.5 Konfigurationsdefinitionen

Die Konfigurationsdefinition wird in der Entwicklungsphase 4.1.1 vom Softwareentwickler erstellt und ist das Austauschformat, um konfigurationsrelevante Informationen weiterzugeben. So können in der Planungsphase alle konfigurierbaren Parameter eines AAL-Services berücksichtigt werden, die vom Entwickler festgelegt wurden. Weiterhin wird die Konfigurationsdefinition verwendet, wenn nach der Installation die Personalisierung des AAL-Services ansteht. Es wird die Konfigurationsdefinition eingelesen und entsprechend zur Instanzbelegung der Konfigurationsoptionen dem Deployer präsentiert. Weiterhin beinhaltet die Konfigurationsdefinition auch Metainformationen, die den Personalisierungsprozess geeignet unterstützen. In den Abbildungen 4.11 und 4.12 ist ein Überblick zu allen Objekten der Konfigurationsdefinition gegeben. Die orange hervorgehobenen Objekte sind hierbei verpflichtend auszufüllen, während alle anderen optional sind. Weiterhin können einige Objekte in einer Instanzbeschreibung nur einmal vorkommen (siehe *hasA*-Beziehung) und andere Objekte beliebig oft (siehe *has[0..n]*-Beziehung) in einer Konfigurationsdefinition verwendet werden. Vier verschiedene Objekte in Abbildung 4.11 erben vom gleichen Objekt „*configItem*“ (siehe Abbildung 4.12) und erweitern es geeignet. Die Struktur ist bei neuen Anforderungen leicht zu erweitern. In der folgenden Aufzählung sind die einzelnen Objekte der Instanziierung genauer beschrieben.

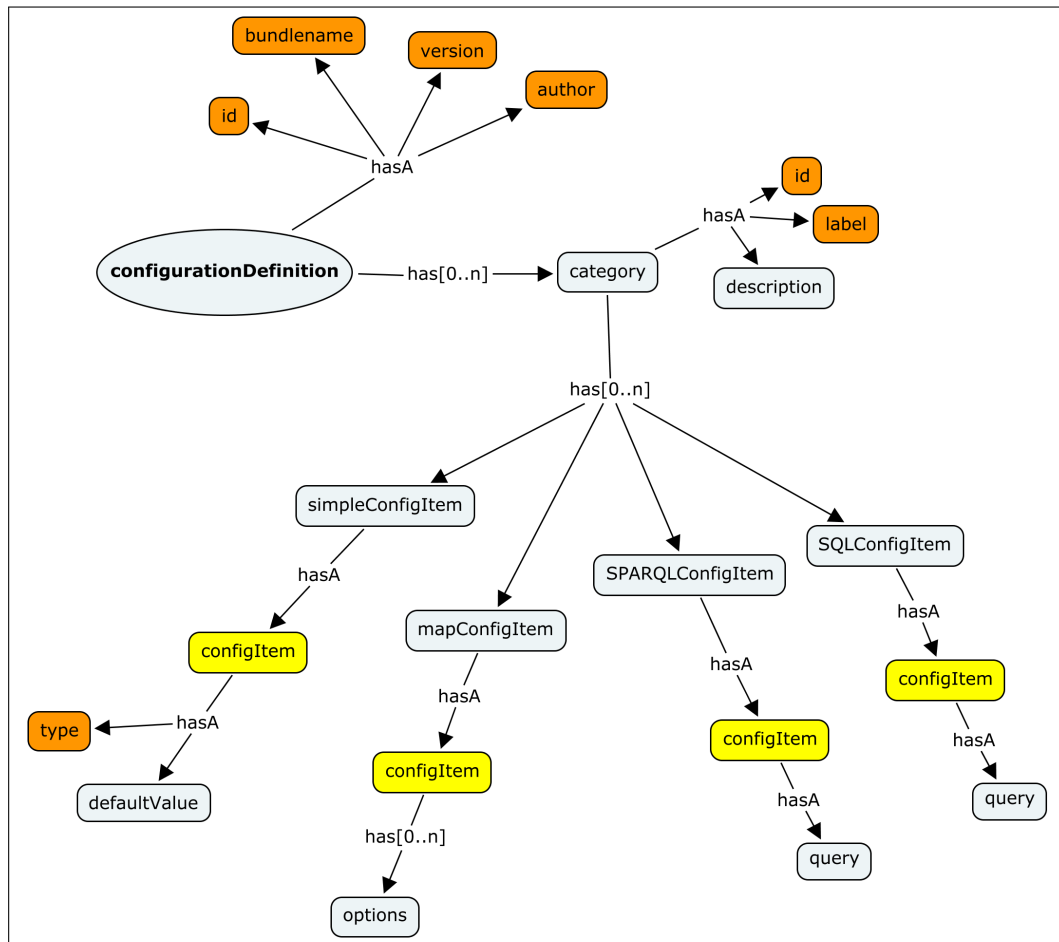


Abbildung 4.11: Überblick über die Konfigurationsdefinition

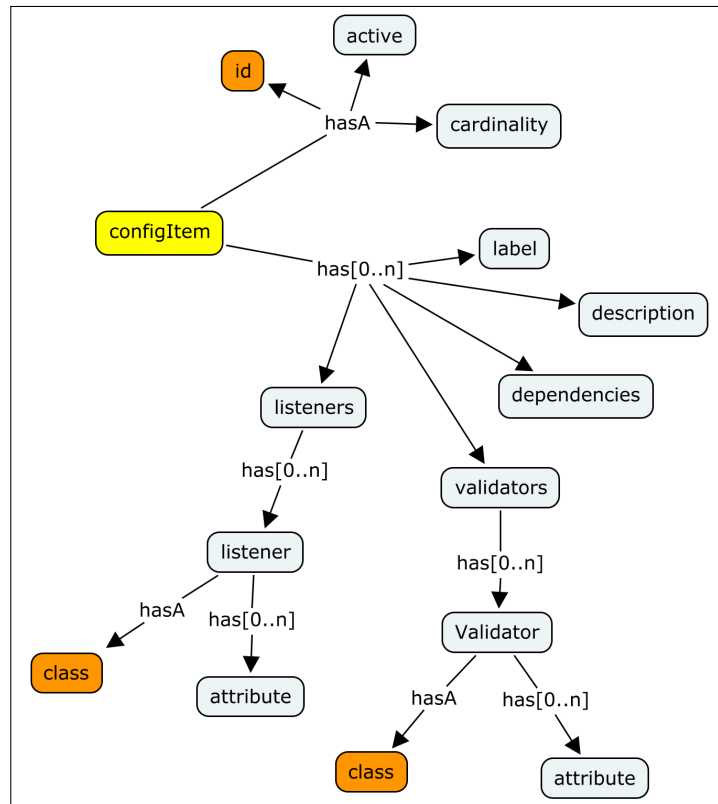


Abbildung 4.12: Überblick über das configItem

configurationDefinition: Es ist das oberste Objekt der Konfigurationsdefinition (*configurationDefinition*) eines AAL-Services. Es dient zur Repräsentation aller konfigurierbaren Objekte. Die Konfigurationsdefinition beinhaltet die folgenden Elemente:

id: Dieses Objekt erlaubt die eindeutige Identifizierung einer Konfigurationsdefinition. Es ist zwingend erforderlich, damit eine eindeutige Verarbeitung durch alle späteren Phasen im Konfigurationslifecycle möglich ist. Bei einer späteren Verwendung dieses Objektes in der gesamten AAL-Umgebung empfiehlt es sich, hier eine URI zu wählen. Dieses Objekt kann oft als *id* in die Konfigurationsinstanz übernommen werden.

bundleName: Der Name (*bundleName*) dient der Zuordnung der Konfigurationsdefinition zu einem AAL-Service. Es sollte der Projektname des Services eingetragen werden. Diese ID ist verpflichtend auszufüllen. Das Objekt kann oft als *usecaseid* in die Konfigurationsinstanz übernommen werden.

version: Die Version (*version*) ist eine vom Entwickler gesetzte Versionsnummer für die Konfigurationsdefinition. Zusammen mit dem *bundleName* stellt sie als verpflichtendes Objekt eine eindeutige Identifizierung des AAL-Services. Die Version sollte der Versionsnummer des Bundles entsprechen. Dieses Objekt kann oft als *version* in die Konfigurationsinstanz übernommen werden.

author: In den meisten Fällen wird hier der Name des Entwicklers eingetragen. Das Feld dient der Identifikation des Autors (*author*) der Konfigura-

tionsdefinition. Vor allem bei größeren Softwareprojekten mit mehreren Entwicklern kann hierüber die Verantwortlichkeit sichergestellt werden.

category: Neben den Standardinformationen einer Konfigurationsdefinition werden alle weiteren Objekte in Kategorien (*category*) zusammengefasst. Kategorien erleichtern die spätere Auffindung einzelner Konfigurationsoptionen sowie ihre Repräsentation gegenüber dem Benutzer. Eine Kategorie beinhaltet hierbei folgende Objekte:

id: Diese ID identifiziert die Kategorie eindeutig und ist somit zwingend erforderlich. Es kann für eine plattformweite Nutzung der Kategorie sinnvoll sein, hier eine URI zu verwenden. Zusammen mit der *Configuration-id* kann so die Kategorie AAL-serviceübergreifend in der AAL-Umgebung eindeutig beschrieben werden.

label: Dieses Objekt persistiert den Beschreibungstext (*label*) der Kategorie. Da eine Kategorie mehrere Konfigurationsoptionen zusammenfassen kann, wird hier eine sie beschreibende Überschrift eingetragen. Dieses Objekt ist verpflichtend auszufüllen.

description: Das Beschreibungsobjekt (*description*) erlaubt weitere Ergänzungen zu der Kategorie. Dies ist für das Verstehen der Konfigurationsdefinition und -optionen während der Nutzung durch den Deployer in späteren Phasen (z.B. Personalisierung) notwendig.

simpleConfigItem: Dieses Objekt (*simpleConfigItem*) ist die einfachste Form, einen Parameter konfigurierbar zu machen. Es können sich beliebig viele *simpleConfigItem* in einer Kategorie befinden. Objekte, die mehrfach genutzt werden, sind in der Abbildung 4.11 gelb markiert und in Abbildung 4.12 erweitert dargestellt. Diese Kategorie enthält die folgenden Objekte zur näheren Beschreibung:

id: Die ID identifiziert das Konfigurationselement und ist im Falle einer Nutzung außerhalb des dazugehörigen AAL-Services mit einer URI zu versehen. Diese ID sichert eine eindeutige Identifizierung innerhalb des AAL-Services und in Kombination mit den IDs der darüberliegenden Objekte auch eine plattformweite Identifizierung. Darum ist dieses Objekt verpflichtend.

active: Jedes Konfigurationselement kann aktiv (*active*) oder inaktiv sein. Je nachdem, wie die Zielumgebung und die damit verbundenen Anforderungen aussehen, werden nur die Parameter für die notwendigen aktiven Konfigurationselemente abgefragt.

cardinality: Die Kardinalität (*cardinality*) eines Konfigurationselements beschreibt, wie oft dieses Element mit einem Konfigurationsparameter instanziiert werden kann. Die Kardinalität kann hierbei von 0 (entspricht einer optionalen Belegung) bis zu n Instanzen reichen.

label: Der Name (*label*) des Konfigurationselements dient als Beschreibung desselben. Diese ist wichtig zum Verständnis, was zu konfigurieren bzw. welcher Parameter in der Personalisierungsphase einzutragen ist.

description: Das Beschreibungsobjekt (*description*) soll zusammen mit dem *label* eine aussagekräftige Darstellung für die Instanziierung der Konfigurationsoption bieten.

dependencies: Mit Hilfe dieses Objekts können Abhängigkeiten (*dependencies*) zwischen verschiedenen Konfigurationsoptionen hergestellt werden. Durch Abhängigkeiten und die Möglichkeit einzelne Konfigurationselemente aktiv zu schalten, kann die gesamte Konfiguration gegenüber dem Deployer dynamisch gestaltet werden.

validators: Zu den dynamischen Komponenten gehören auch die Validatoren (*validators*), welche erlauben die Eingabe für ein Konfigurationselement zu überprüfen. Es können hierbei beliebig viele Validatoren auf einer Konfigurationsoption ausgeführt werden. Verschiedene einfache Validatoren sollten ohne Aufwand für den Entwickler verfügbar sein. Weiterhin sollte es für den Entwickler auch die Möglichkeit geben, eigene Validatoren in der Konfigurationsdefinition mitzuliefern. Hier sind einfache Prüfungen mit einem regulären Ausdruck oder ob ein Parameter in einem vorgegebenen Zahlenbereich liegt denkbar. Für Validatoren ist die Java-Klasse (*class*), über die sie sich ausführen lassen, verpflichtend anzugeben. Außerdem gibt es die Möglichkeit, in der Konfigurationsdefinition Attribute (*attribute*) für den Validator zu übergeben.

listeners: Eine weitere Komponente zur Unterstützung der Dynamik während der Konfiguration und um den Anforderungen der Regeln gerecht zu werden, ist der Listener. Er überwacht eine Konfigurationsoption und kann auf ein bestimmtes Ereignis während seiner Überwachung reagieren und etwas anderes auslösen. Einfache Listener sollten, wie bei den Validatoren, gegeben sein. Weiterhin müssen sich die Konzepte durch neue Listener leicht erweitern lassen. Ein Listener, der auf Änderungen der Konfigurationsoption eine Aktion ausführt, wäre ein guter Standardkandidat. Zu einer Konfigurationsoption können beliebig viele Listener zugeordnet werden. Hierzu wird jeweils zwingend die Java-Klasse (*class*) zu deren Ausführung angegeben. Weiterhin können auch Listenern Attribute (*attribute*) übergeben werden.

type: Neben den bisher aufgeführten Objekten besitzt das *SimpleConfigItem* noch den Objekt Typ (*type*). Es dient dazu näher zu spezifizieren, welche Art (z.B. String, INT, DOUBLE) von Parameter sich hinter diesem Konfigurationselement verbirgt. Die Angabe des Typs ist obligatorisch.

defaultValue: Dieses Objekt dient zur Angabe eines Standardwerts für den zu konfigurierenden Parameter. Dies ist wichtig, um einen AAL-Service auch ohne Personalisierung rudimentär lauffähig zu halten. Weiterhin kann dieser Wert genutzt werden, falls Probleme in der AAL-Umgebung auftreten, die sich nicht über die Nutzung der Sekundärkonfigurationsinstanz lösen lassen.

mapConfigItem: Dieses Objekt (*MapConfigItem*) dient dazu, aus einer statischen Liste von Elementen den passenden Parameter zu wählen. Von diesem Objekt können sich beliebig viele in einer Kategorie befinden. Neben dem folgenden Objekt *options* besitzt das *MapConfigItem* alle Objekte des *configItems* siehe Abbildung 4.11 und 4.12.

options: Über dieses Objekt werden die Elemente, die zur Auswahl stehen, definiert. Es kann beliebig viele Optionen (*options*) geben.

SPARQLConfigItem: Das SPARQL-Konfigurationselement (*SPARQLConfigItem*) dient als Eingabe für eine dynamische Liste von Elementen zur Auswahl von einem Parameter. Hiermit kann auf die Wissensbasis der gesamten AAL-Umgebung zugegriffen werden. Neben dem Objekt *query* besitzt das *SPARQLConfigItem* alle Objekte des *configItems*.

query: In diesem Objekt wird die SPARQL-Abfrage (*query*) abgelegt.

SQLConfigItem: Dieses Konfigurationselement (*SQLConfigItem*) ist ähnlich aufgebaut wie das vorherige. Auch das SQL-Konfigurationselement dient der dynamischen Bereitstellung von Elementen zur Auswahl von einem Parameter für die Konfigurationsinstanz. Der Unterschied zum *SPARQLConfigItem* besteht lediglich in der Abfrageart der Elemente. Neben dem Objekt *query* besitzt das *SQLConfigItem* alle Objekte des *configItems*.

query: In diesem Objekt wird die SQL-Abfrage (*query*) abgelegt.

Die formale Beschreibung der Konfigurationsdefinitionen wurde als XSD Schema² abgebildet. Für das folgende Beispiel³ werden alle Elemente bis zur *category* vorausgesetzt und definieren nur die Rahmenbedingungen für die *configOptions* Timeout und Person. Für Timeout benötigen wir ein *simpleConfigItem*, das in Listing 4.1 beispielhaft dargestellt ist. Es fordert in der Konfiguration die Eingabe eines Integers, der durch einen Validator vom Typ Regulärer Ausdruck überprüft wird. Wird keine Eingabe vorgenommen ist der Standardwert 10.

```

1 <universaal:SimpleConfigItem cardinality="1..1" id="timeout" type="int">
2   <universaal:label>Timeout for the Reminder</universaal:label>
3   <universaal:description>Please insert the seconds for the timeout.
4   </universaal:description>
5   <universaal:validators>
6     <universaal:validator class=" de.fzi.aal.configuration.model.validators.
7       RegExpValidator">
8       <universaal:attribute>[0-9]*</universaal:attribute>
9     </universaal:validator>
10  </universaal:validators>
11  <universaal:defaultValue>10</universaal:defaultValue>
12 </universaal:SimpleConfigItem>

```

Listing 4.1: Beispiel einer möglichen Timeout-Konfigurationsdefinition in XML

Für Person benötigen wir ein *SPARQLConfigItem*, das in Listing 4.2 beispielhaft dargestellt ist. In der Abfrage werden 10 Personen der Gruppe *AssistedPerson* mit ihrem Namen ausgegeben. Während der Konfiguration wird eine von ihnen ausgewählt und als Konfigurationsinstanz gespeichert.

²<http://forge.universaal.org/svn/uaaltools/trunk/uCC/ucc.configuration.model/xmlFiles> (Zugriff am 21.11.2014)

³vgl. Beispielanwendungsfall Kapitel 1


```

1 <universaal:SPARQLConfigItem cardinality="1..1" id="person">
2   <universaal:label>Choose a Contact Person</universaal:label>
3   <universaal:description>Please choose a contact person from the
4     ontology instances.</universaal:description>
5   <universaal:query>
6     PREFIX uAAL:<http://ontology.universaal.org/Profile.owl#>
7     Select ?name
8     WHERE{
9       ?object uAAL:Profile "AssistedPersonProfile" .
10      ?object uAAL:name ?name .
11    }LIMIT 10
12  </universaal:query>
13 </universaal:SPARQLConfigItem>

```

Listing 4.2: Beispiel einer möglichen Konfigurationsdefinition für die Auswahl der zu benachrichtigenden Person in XML

4.3.6 Manifest

Der AAL-Service besteht aus mehreren Teilen, die durch ein Manifest beschrieben werden. So stehen alle wichtigen Informationen für die Phasen im Konfigurationsliifecycle bereit. Es beinhaltet die folgenden Informationsgruppen.

- Allgemeine Informationen zum AAL-Anwendungsfall
- Ansprechpartner mit Kontaktinformationen
- Service-Level-Agreement und Lizenzen
- Informationen zur Installation
- Abhängigkeiten zu anderen Anwendungsfällen
- Beschreibung einzelner Komponenten des Anwendungsfalls

Das Listing 4.3 zeigt einen Ausschnitt der XSD-Schema-Datei⁴ des Manifests zum Thema allgemeine Informationen.

```

1 <xs:element name="name" type="xs:string">
2   <xs:annotation>
3     <xs:documentation>friendly name</xs:documentation>
4   </xs:annotation>
5 </xs:element>
6 <xs:element name="version" type="uapp:versionType">
7   <xs:annotation>
8     <xs:documentation>version of the service</xs:documentation>
9   </xs:annotation>
10 </xs:element>
11 <xs:element name="serviceId" type="xs:string">
12   <xs:annotation>
13     <xs:documentation>unique name e.g. package name</xs:documentation>
14   </xs:annotation>
15 </xs:element>
16 <xs:element name="description" type="xs:string">
17   <xs:annotation>
18     <xs:documentation>free text description about the service</xs:documentation>
19   </xs:annotation>
20 </xs:element>

```

Listing 4.3: Auszug aus dem XSD-Schema der Manifestdatei

⁴<http://forge.universaal.org/svn/uaaltools> (Zugriff am 21.11.2014)

4.4 Ontologien

Die Ontologien bilden einen wichtigen Teil der zu konfigurierenden Parameter. Gerade durch sie wird es möglich, flexibel über mehrere AAL-Anwendungsfälle hinweg zu konfigurieren. Das Projekt universAAL stellt bereits die technische Grundstruktur zur Nutzung von Ontologien. Das Ontologie-Management von universAAL unterstützt beim Integrieren neuer Ontologien auf OWL⁵-Basis oder deren Instanziierung. Weiterhin wird ein automatisches Ontologie-Alignment und -Mapping unterstützt. Der Austausch der Ontologieobjekte zwischen den verschiedenen Komponenten der AAL-Plattform und zwischen AAL-Services erfolgt durch serialisiertes RDF⁶.

Für die Konfiguration muss definiert sein, welche Ontologien wo und wie im Lifecycle zu erstellen, zu instanzieren und zu nutzen sind. So ist es möglich, dass die Informationen in der Ontologie helfen, AAL-Anwendungsfälle semiautomatisch zu konfigurieren und bei Bedarf passende Instanzen anzulegen oder auszuwählen.

Die Erinnerungsfunktion im Anwendungsfall „Nutritional Advicer“ ist hierfür ein gutes Beispiel. Sie versucht den Bewohner in der Wohnung ausfindig zu machen, um ihn an das Trinken zu erinnern. Sollte das Auffinden über längere Zeit nicht möglich sein, kann der Anwendungsfall eine weitere Person informieren. Der Entwickler des Anwendungsfalls beschreibt semantisch, welche Art von Sensorik und Ansprechpartner hierfür benötigt werden. In der anschließenden Planungsphase wird diese Beschreibung zusammen mit der in der Space-Ontologie verfügbaren Sensorik ausgewertet. Wenn die Sensorik in der AAL-Umgebung die Anforderungen des Anwendungsfalls nicht erfüllt, kann aus einem Hardware-Ontologie-Repository ein passender Kandidat gewählt werden. Der Deployer kann diese Auswahl durch Einschränkungen (z.B. Preis, Farbe) auf der Hardware-Ontologie beeinflussen. Nach der Installation des AAL-Anwendungsfalls und seiner Hardware kann entsprechend der Ergebnisse der Planungsphase die Instanziierung der Sensorik in der AAL-Umgebung automatisch geschehen. Jetzt ist noch der Ansprechpartner zu wählen, der im Falle der Nichterreichbarkeit zu kontaktieren ist. Der Entwickler kann definieren, welche Eigenschaften diese Person aufweisen soll. Nur der Bewohner wird einen für sich passenden Kontakt auswählen können. So werden in der Personalisierung entsprechend den Anforderungen des Anwendungsfalls die möglichen Kontakte zur Auswahl gestellt. Der Deployer kann in der Personalisierungsphase bei Bedarf auch neue Kontakte in der Ontologie ablegen.

Zur Nutzung der Ontologien im Konfigurationslifecycle wurde das Konzept der Profile in den Ontologien eingeführt. Sie ermöglichen Personen, AAL-Anwendungsfälle und Hardware für die AAL-Umgebung mit konfigurationsrelevanten Informationen zu versehen. In Abbildung 4.13 ist ein Ausschnitt⁷ des dazugehörigen entwickelten Design Patterns zu sehen. Auf höchster Ebene verwendet die Profile-Ontologie die folgenden drei Konzepte:

ProfilableEntity: Jede Komponente eines AAL-Anwendungsfalls kann von dieser Klasse Sub-Konzepte bilden, die sie für seine Funktionen benötigt. Diese

⁵Web Ontology Language - OWL 2 Document Overview – W3C-Empfehlung vom 27. Oktober 2009 <http://www.w3.org/TR/owl2-overview/> (Zugriff am 21.11.2014)

⁶Resource Description Framework <http://www.w3.org/RDF/> (Zugriff am 21.11.2014)

⁷Die gesamte Profilontologie ist verfügbar unter: <http://ontology.universaal.org/Profile.owl> und <http://ontology.universaal.org/Profile.ttl> (Zugriff am 21.11.2014)

Klassen werden teilweise von anderen AAL-Ontologien bereitgestellt und sind Sub-Konzepte von den universAAL Superklassen `ManagedIndividual` oder `PhysicalThing`.

Profile: Jede `ProfileableEntity` (Domain) kann über `hasProfile` (Property) mit `Profile` (Range) verbunden werden. `Profile` beinhalten allgemeine Informationen entsprechend der jeweiligen Software, Hardware oder menschlichen Dienstleistung.

SubProfile: Jedes `Profile` (Domain) kann beliebige `SubProfiles` (Range) haben. Ihre Instanzen sind über `hasSubProfile` (Property) verbunden. In ihr werden speziellere domänenspezifische Informationen abgelegt.

Von diesen drei Konzepten ausgehend erben verschiedene Konzepte, die konfigurationsrelevante Informationen tragen (siehe Abbildung 4.13). Sie lassen sich in die aus dem Lifecycle bekannten Informationsgruppen AAL-Umgebung, AAL-Anwendungsfall, Akteure und Hardware teilen.

Die AAL-Umgebungsinformationen werden im `Space` und seinen `SpaceProfiles` abgebildet. In ihnen werden alle Komponenten der Umgebung genau spezifiziert. Dies ermöglicht den AAL-Anwendungsfällen sich über die Komponenten der Umgebung auszutauschen und sie gemeinsam zu nutzen. Jedem `Space` kann so beispielsweise Hardware (`Device`) zugeordnet werden.

Informationen zu AAL-Anwendungsfällen werden durch das Konzept `AALService` abgebildet. Durch seine `Profile` und `SubProfile` beschreibt der Service sich selbst. Über weitere `SubProfile` beschreibt der Service seine Schnittstellen (`AALAppSubProfile`), Hardware (`HardwareSubProfile`) und Abhängigkeiten (`Requirement`). `SubProfiles` der Personen werden über die Property (`humanResourceSubProfile`) an den AAL-Service gebunden. Die Schnittstellen beschreiben, welche Informationen zwischen Anwendungsfällen ausgetauscht werden können und welche Informationen der Service von anderen Anwendungen benötigt. Diese anderen Anwendungen können ihrerseits wieder Abhängigkeiten (`Requirement`) zu Hardware und Software haben. Das `HardwareSubProfile` beschreibt, welche Sensorik und Aktorik bzw. damit verbundene Informationen benötigt werden. Über `humanResourceSubProfile` spezifiziert der Service seine Anforderungen an bestimmte Personen.

Die Informationen zu Akteuren in der AAL-Umgebung werden in den Konzepten des `Users` und seinen Derivaten dargestellt. Auch sie haben verschiedene `Profiles` und `SubProfiles` zur umfassenden Beschreibung der Personen.

Die Hardwareinformationen werden von universAAL in einer `Device` Ontologie⁸ realisiert. Diese ist für die Konfigurationszwecke erweitert worden. Sie beschränkte sich vorerst nur auf die für die Funktion in der AAL-Umgebung notwendigen Informationen. Während des Konfigurationslifecycles werden aber weitere Informationen benötigt, die durch Erweiterungen der universAAL-Ontologie realisiert sind. Für die Sensorik wurden Teile der Semantik Sensor Network Ontology des W3C (SSN) [LeHT14] übernommen und für die Aktorik und Metainformationen eigene Erweiterungen integriert. Neben der Nutzung der `Device` Ontologie im Lifecycle ist mit ihr

⁸<http://ontology.universaal.org/Device.owl> (Zugriff am 21.11.2014)

auch das Hardware-Ontologie-Repository umgesetzt, das alle verfügbare Hardware für AAL-Umgebungen abbilden kann.

Abschließend soll ergänzend zu Abbildung 4.13 entsprechend des Beispielanwendungsfalls (vgl. Kapitel 1) die Ontologie für die „Assisted Person“ vorgestellt werden. In Listing 4.4 ist ein Ausschnitt dargestellt, der die „Assisted Person“ sowie ihr Profil beschreibt und den Zugriff auf alle damit verbundenen Informationen innerhalb der AAL-Umgebung erlaubt. Sie ermöglicht zur Konfigurationszeit entsprechend der Konfigurationsdefinition alle „Assisted Persons“ der AAL-Umgebung abzufragen und eine zu wählen, um sie als Konfigurationsinstanz zu speichern.

```

2 <Class rdf:about="http://ontology.universAAL.org/Profile.owl#AssistedPerson">
3   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
4     Assisted Person
5   </rdfs:label>
6   <rdfs:subClassOf rdf:resource="http://ontology.universAAL.org/
7 Profile.owl#User"/>
8   <rdfs:subClassOf>
9     <Restriction>
10      <onProperty rdf:resource="http://ontology.universAAL.org/
11 Profile.owl#hasProfile"/>
12      <maxCardinality rdf:datatype="http://www.w3.org/2001/
13 XMLSchema#nonNegativeInteger">
14        1
15      </maxCardinality>
16    </Restriction>
17 </rdfs:subClassOf>
18 <rdfs:subClassOf>
19   <Restriction>
20     <onProperty rdf:resource="http://ontology.universAAL.org/
21 Profile.owl#hasProfile"/>
22     <allValuesFrom rdf:resource="http://ontology.universAAL.org/
23 Profile.owl#AssistedPersonProfile"/>
24   </Restriction>
25 </rdfs:subClassOf>
26 <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
27   The assisted person that is the end user of the system
28 </rdfs:comment>
29 </Class>

31 <Class rdf:about="http://ontology.universAAL.org/
32 Profile.owl#AssistedPersonProfile">
33   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
34     Assisted Person Profile
35   </rdfs:label>
36   <rdfs:subClassOf rdf:resource="http://ontology.universAAL.org/
37 Profile.owl#UserProfile"/>
38   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
39     Profile of an assisted person
40   </rdfs:comment>
41 </Class>

```

Listing 4.4: Beispiel der AssistedPerson- und Profile-Ontology

4.5 Zusammenfassung

Dieses Kapitel beschreibt die Konzeption des Konfigurationsframeworks im Sinne der DSR-Methode. Hierbei wurde ein Lifecycle erarbeitet, der über fünf Phasen AAL-Anwendungsfälle durch die Konfiguration begleitet. Den einzelnen Phasen der Entwicklung, Planung, Installation, Personalisierung und Wartung konnten hierbei verschiedene Akteure entsprechend der Aufgaben zugeordnet werden. Wichtig ist vor allem die konzeptionelle Entwicklung der verschiedenen Konfigurationskomplexitäten sowie die daraus entstandenen Regeln, Konfigurationsparameter und Ontologien.

Die sieben kumulierten technischen Anforderungen konnten somit durch die Konzepte entsprechend bedient werden.

R 1: Das Framework soll Dienste und Werkzeuge zur Installation neuer AAL-Anwendungsfälle in der AAL-Umgebung bereitstellen.

Diese Forderung wird von der Installationsphase des entwickelten Lifecycles abgedeckt. Sowohl die Konzepte (siehe Kapitel 4.1.3) als auch die beteiligten Personen (siehe Kapitel 4.2) und notwendigen Informationen (siehe Kapitel 4.3.6) konnten definiert werden.

R 2: Das Framework soll Dienste und Werkzeuge zur Personalisierung neuer AAL-Anwendungsfälle in der AAL-Umgebung bereitstellen. Diese Anpassungen sollen nach der Initialisierung des AAL-Anwendungsfalls zu jedem Zeitpunkt möglich sein.

Die Forderung der zeitpunktungebundenen Personalisierung wurde in den entsprechenden Phasen (siehe Kapitel 4.1.4 und 4.1.5) eingebaut und durch Änderungen an den AAL-Anwendungsfällen (siehe Kapitel 4.3) erreicht.

R 3: Dem Framework soll eine einheitliche Konfigurationssprache zu Grunde liegen, die alle beteiligten Komponenten und Akteure des Frameworks nutzen.

Die entwickelte Konfigurationssprache (siehe Kapitel 4.3.3, 4.3.5 und 4.4) ist nach Komplexitätsklassen (siehe Kapitel 4.3.1) eingeteilt und über den gesamten Lifecycle der Konfiguration für alle Akteure entsprechend ihrer Aufgaben verfügbar.

R 4: Das Framework soll den Austausch von Verbrauchsgütern oder defekten Komponenten in der AAL-Umgebung unterstützen.

Diese Forderung ist Teil des flexiblen Umgangs mit dem AAL-Anwendungsfällen im Framework und ihrer Aufteilung in Komponenten (siehe Kapitel 4.3), die untereinander mit semantischen Schnittstellen (siehe Kapitel 4.4) arbeiten.

R 5: Das Framework soll das Updaten und Deinstallieren von AAL-Anwendungsfällen unterstützen.

Die Wartungsphase (siehe Kapitel 4.1.5) unterstützt das Updaten und Deinstallieren von AAL-Anwendungsfällen. Hierbei unterstützen neben dem Manifest verschiedenen Konfigurationsparameter (z.B. ID und Version; vgl. Kapitel 4.3.4), so dass eine eindeutige Zuordnung im Falle eines Updates oder einer Deinstallation möglich ist.

R 6: Das Framework soll mit der Middleware kommunizieren, um Informationen in der AAL-Umgebung bereitzustellen oder aus ihr zu beziehen.

Diese Forderung wurde über Ontologien realisiert (siehe Kapitel 4.4).

R 7: Das Framework soll verschiedene Akteure mit verschiedenen Rechten unterstützen.

Diese Forderung ist durch die Aufteilung in verschiedene Rollen und ihre Beteiligung an den verschiedenen Lifecyclephasen erfüllt (siehe Kapitel 4.2).

5. Implementierung des Konfigurationsframeworks

Entsprechend der DSR-Methode werden in diesem Kapitel die verschiedenen Konzepte als Artefakte implementiert, um sie anschließend evaluieren zu können. Sie werden entsprechend der Konzeption in einem Framework zusammengefasst und entlang des Lifecycles genutzt. Dieses Kapitel stellt zu den verschiedenen Phasen des Konfigurationslifecycles die dazugehörigen Werkzeuge vor. Zum besseren Verständnis werden die Werkzeuge mit dem Beispiel des Erinnerungsanwendungsfalls verknüpft.

Das gesamte Konfigurationsframework ist als Open Source¹ unter der Apache Lizenz² verfügbar. Es wird von der AALOA Community³ gepflegt und in verschiedenen Projekten (z.B. EU-Projekt ReAAL⁴) verwendet.

Im Kapitel 5.1 werden die Werkzeuge für den Entwickler vorgestellt, die ihm helfen seine AAL-Anwendungsfälle konfigurierbar zu machen. Anschließend zeigt Kapitel 5.2 wie die Planungswerkzeuge implementiert wurden und wie sie die Einrichtung einer AAL-Umgebung unterstützen. Das Kapitel 5.3 bespricht die Werkzeuge zur Installation, Personalisierung und Wartung der AAL-Anwendungsfälle und AAL-Umgebung. Abschließend werden in Kapitel 5.4 die verschiedenen Dokumentationen und Trainingsmaterialien für die verschiedenen Akteure und ihre Werkzeuge vorgestellt.

5.1 Werkzeuge für die Entwicklung

Für die Phase der Entwicklung im Lifecycle wurden zwei verschiedene Werkzeuge zur Unterstützung des Entwicklers implementiert. Diese sind der „Configuration Editor“ (CE) und der „File Packager“ (FP). Der CE unterstützt den Entwickler

¹<http://forge.universaal.org/svn/uaaltools> (Zugriff am 21.11.2014)

²<http://www.apache.org/licenses/LICENSE-2.0> (Zugriff am 21.11.2014)

³<http://www.aalooa.org> (Zugriff am 21.11.2014)

⁴<http://www.cip-reaal.eu/> (Zugriff am 21.11.2014)

dabei, AAL-Anwendungsfälle konfigurierbar zu machen und der FP setzt die verschiedenen Komponenten des Anwendungsfalls mit den notwendigen Metadaten zu einem Paket zusammen. Dieses wird in die weiteren Phasen des Lifecycles übergeben. Der CE sowie der FP sind als Eclipse Plugin⁵ realisiert, so dass sie direkt in die Entwicklungsumgebung eingebunden sind.

Der CE wird vom Entwickler genutzt, um eine Konfiguration zu der Soft- und Hardware seines AAL-Service zu erzeugen oder zu editieren. Hierbei unterstützt er alle in Kapitel 4.3.5 vorgestellten Konfigurationsdefinitionen. Dies bedeutet, dass neben den verschiedenen Konfigurationsoptionen auch die Nutzung der Validatoren, Listener und Regeln möglich ist.

Der CE überprüft die Eingaben des Entwicklers gegen das XSD-Schema und lässt nur gültige Konfigurationen zu. Hierzu unterstützt er eine einfache Editoransicht (siehe Abbildung 5.1) und eine XML-Quellcodeansicht. Zwischen beiden Sichten kann über Karteireiter unten im Bild jederzeit umgeschaltet werden. Im Kasten „Konfiguration“ werden die Basisinformationen notiert. Darunter befindet sich auf der linken Seite ein Überblick zu allen erstellten Konfigurationsdefinitionen (z.B. UpdateTime, ImageWidth und addMailPlugin), die nach ihren Kategorien sortiert sind (im Beispiel MainParameters und mailPlugin). Die rechte Seite bietet die Möglichkeit zur ausgewählten Konfigurationsdefinition die jeweiligen Parameter zu setzen. Das Ergebnis ist eine XML-Datei, die alle für den Anwendungsfall notwendigen Konfigurationsdefinitionen beinhaltet und in das dazugehörige Java Package gelegt wird.

Der FP ist als Wizard aufgebaut, der die verschiedenen Softwarekomponenten des Anwendungsfalls in einer Datei zusammenfasst. Weiterhin werden Informationen gesammelt und ein Manifest (siehe Kapitel 4.3.6) erstellt, das die weitere Verarbeitung dieser Datei im Lifecycle erlaubt.

5.2 Werkzeuge für die Planung

Das zentrale Werkzeug dieser Phase ist der „Design Assistant“ (DA). Er ist als Android-App (DA-Client) programmiert, um sich möglichst flexibel in den AAL-Umgebungen einsetzen zu lassen. Eine Serverkomponente (DA-Server) zur Persistierung der Daten kommuniziert mit dem DA-Client über das Internet. Auf Basis der in der AAL-Umgebung bereits vorhandenen Informationen aus der Space-Ontologie und der AAL-Service-Ontologie sammelt der DA-Client durch Nutzereingaben weitere zur Installation des Anwendungsfalls notwendige Informationen. Diese werden dann als Ontologieinstanzen auf dem DA-Server gespeichert. In Abbildung 5.2 ist die Übersicht zu einer mit dem DA-Client erstellten Wohnung zu sehen. Die Sichten im DA-Client dienen zur Informationserhebung und -anzeige. Der AAL-Service fordert hierbei neben den Eckdaten der Wohnung auch eine Priorisierung der Zimmer. Sie werden auf Basis der Ontologien dynamisch erzeugt. Der Server übernimmt die weiteren Auswertungen der Clienteingaben. So können mehrere AAL-Umgebungen gleichzeitig mit den neuesten Hardware und Anwendungsfallontologien vom Server versorgt werden. Am Ende der Planung generiert der DA-Server einen PDF-Report, der weitere Anweisungen für die Akteure der Installationsphase enthält. Weiterhin werden die gesammelten Informationen an die Ziel-AAL-Umgebung übertragen.

⁵Plug-in Development Environment (PDE) <http://www.eclipse.org/pde/> (Zugriff am 21.11.2014)

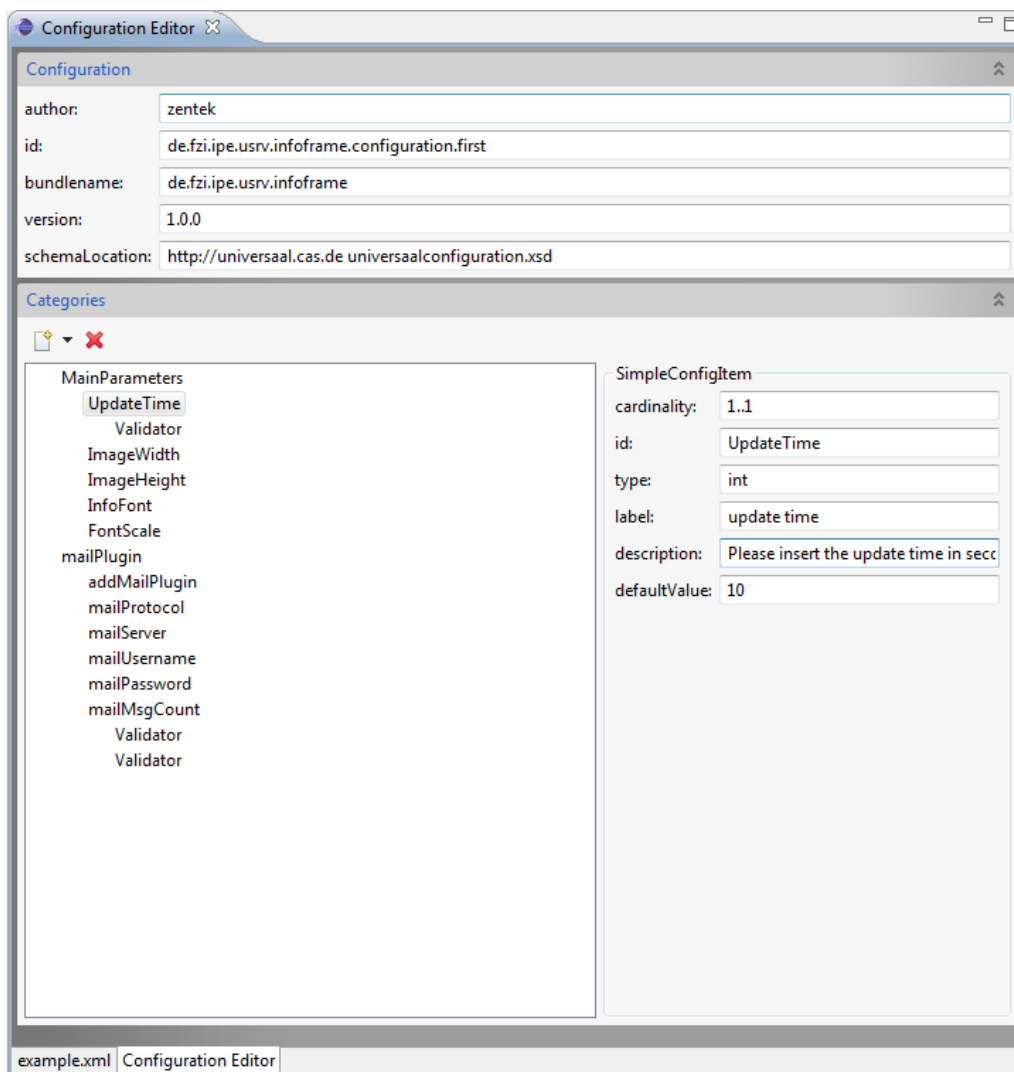
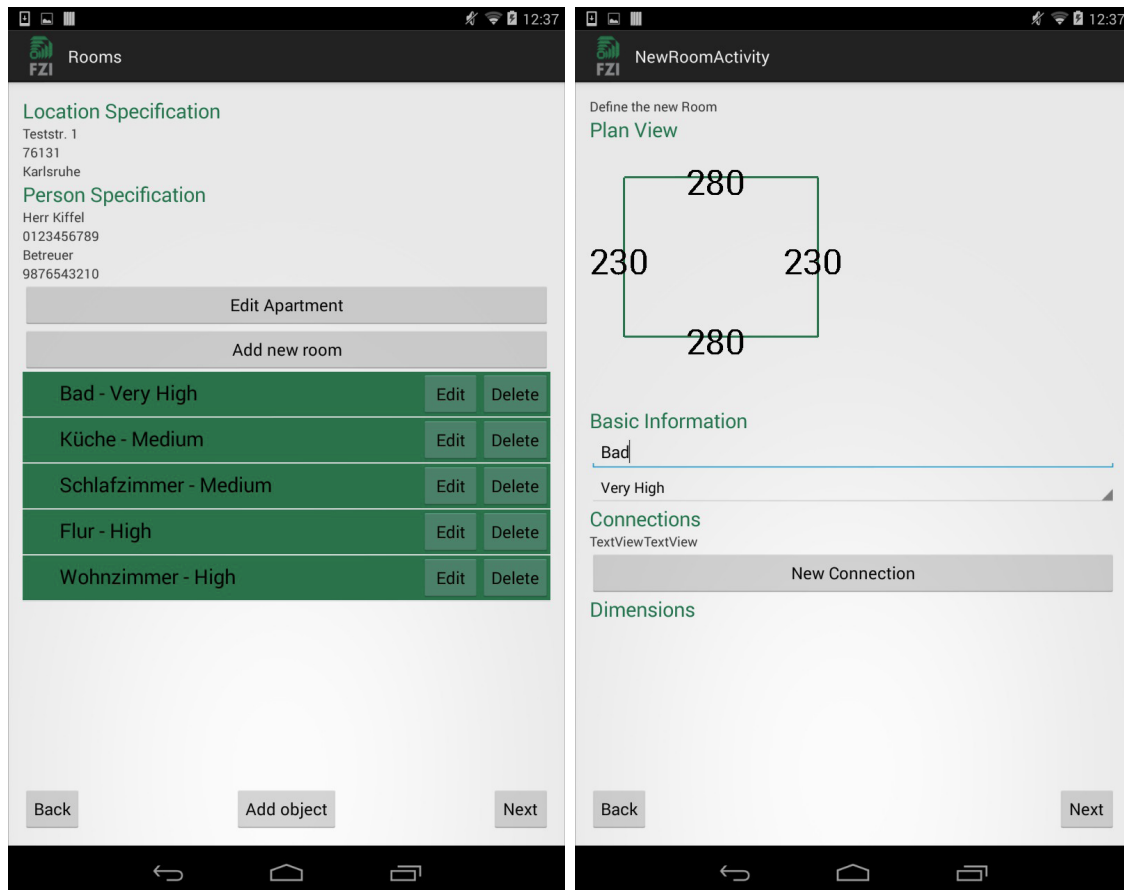


Abbildung 5.1: Darstellung des Configuration Editors in der einfachen Editoransicht



(a) Übersicht zu den Räumen einer Wohnung (b) Einstellungen zu einem Raum der Wohnung

Abbildung 5.2: Überblick zum DA-Client

Die Klassenbibliothek (siehe Komponentendiagramm in Abbildung 5.3) zur Datenübertragung und -repräsentation zwischen DA-Client und DA-Server zerfällt in die folgenden Komponenten:

Base: Diese Komponente enthält Schnittstellen und Hilfsklassen für die **Container** und die **Message**-Komponente.

Container: Er beinhaltet Klassen für die Datenkapselung der gesammelten Informationen auf DA-Client Seite. DA-Server nutzt die **Container**-Komponente, um die Informationen nach der Übertragung auszulesen und weiterzuverarbeiten.

History: Diese Komponente ist für die Zwischenspeicherung aller gesendeten und empfangenen **Messages** verantwortlich. Sie hilft bei Verbindungsabbrüchen und dient zur Realisierung der Undo- und Redo-Funktion.

Message: Die **Message**-Komponente enthält die Klassen zum Austausch von Nachrichten zwischen DA-Client und DA-Server. Sie sind entsprechend ihrer Funktionen in die Unterkomponenten **Add**, **Delete**, **Get** und **Update** aufgeteilt.

Session: Diese Komponente enthält alle relevanten Informationen zur Verbindung zwischen DA-Client und DA-Server.

Utilities: Diese Komponente enthält alle mathematischen Hilfsklassen.

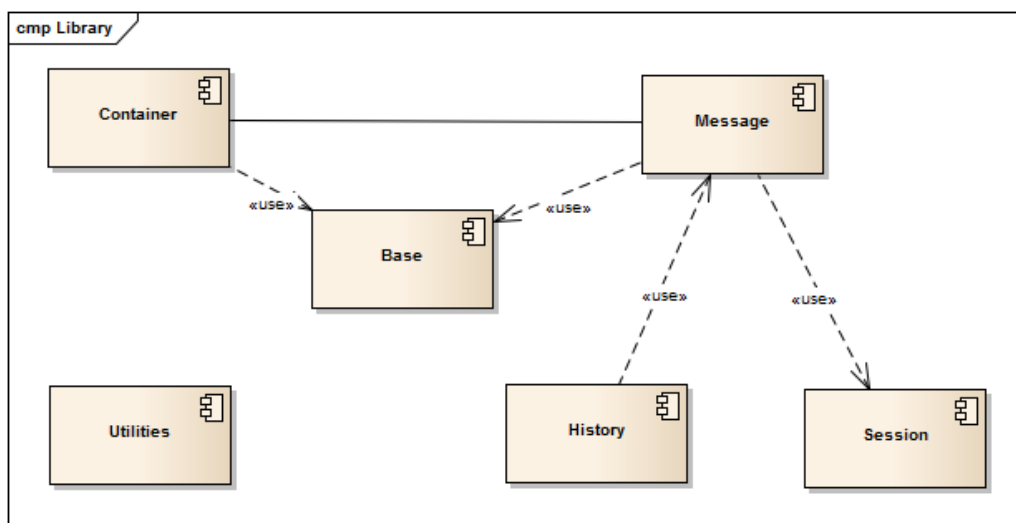


Abbildung 5.3: Komponentendiagramm der gemeinsam genutzten Klassenbibliothek zur Datenübertragung und -repräsentation zwischen DA-Client und DA-Server.

Der DA-Client teilt sich in die Komponenten **Core**, **Data** und **Utilities**, welche androidkonform zusammengeführt sind.

Core: Der **Core** ist die zentrale Komponente des Clients und ist entsprechend seinem Wizardsgedanken als Zustandsautomat realisiert. In ihm sind alle notwendigen **Activities** als Singleton zu finden. Somit ist die Verbindung zum DA-Server von allen **Activities** leicht nutzbar.

Data: Diese Komponente ist für die serielle Kommunikation zum DA-Server verantwortlich.

Utilities: Die Komponente **Utilities** enthält Hilfsklassen und eigene Steuerungselemente, die von der **Core**-Komponente genutzt werden.

Der DA-Server ist in fünf Komponenten realisiert. Hierbei wurde auf starke Parallelisierbarkeit geachtet.

Core: Diese Komponente ist der Startpunkt für den DA-Server und enthält die Handler zum Annehmen und Bearbeiten von DA-Client-Anfragen sowie eine Konsole.

Data: Diese Komponente realisiert die Datenanbindung des Triple Stores über Jena TDB⁶ für die Ontologien.

Handler: In dieser Komponente werden die verschiedenen Anfrageoperationen der DA-Clients realisiert. Die einzelnen Operationen sind in einer Datenbank ausgelagert. So können sie je nach Anwendungsfall erweitert oder angepasst werden.

Printer: Diese Komponente realisiert die Ausgabe der gesammelten Informationen am Ende der Planung als PDF-Datei und in digitaler Form für die weitere Verarbeitung in der AAL-Umgebung.

Report: Die **Report**-Komponente generiert alle Informationen für die Ausgabe durch den **Printer**. Hierbei ist die Ausgabe stark vom zu planenden Anwendungsfall abhängig.

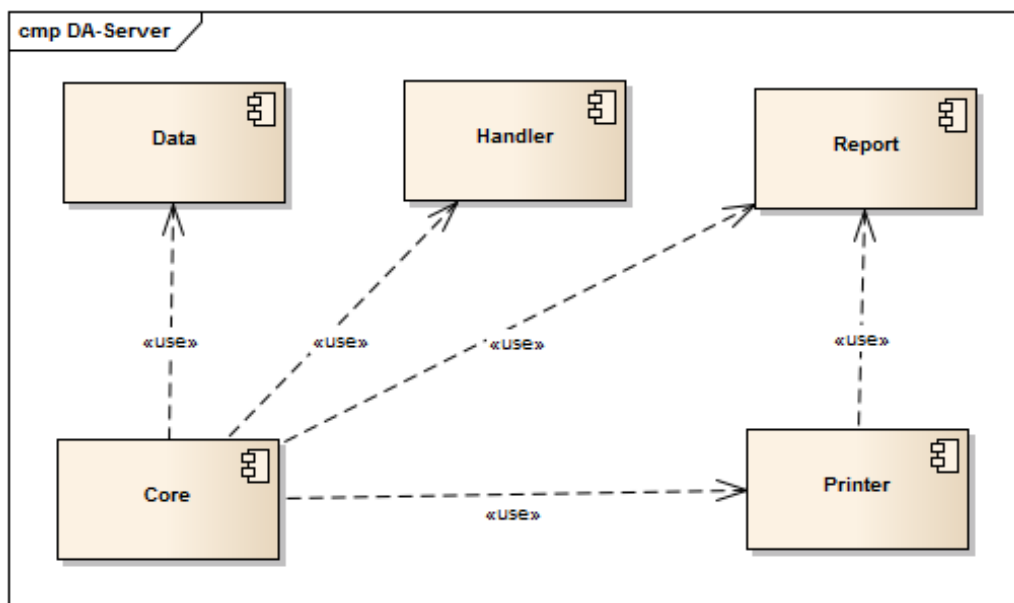


Abbildung 5.4: Komponentendiagramm des DA-Servers

⁶<https://jena.apache.org/documentation/tdb/index.html> (Zugriff am 21.11.2014)

5.3 Werkzeuge für die Installation, die Personalisierung und die Wartung

Die Werkzeuge für die Installation, Personalisierung und Wartung sind im „Control Centre“ (CC) vereint. Der CC ist hierbei Teil der Middleware in der AAL-Umgebung. Er übernimmt entsprechend der Phasen drei Hauptaufgaben. Als erstes installiert er den AAL-Anwendungsfall, indem er die vom FP erstellte Datei entpackt und entsprechend des Manifests (siehe Kapitel 4.3.6) die Installation der Software und Ontologien ausführt. Nach der Installation erfolgt die Personalisierung des Anwendungsfalls durch die zwei CC-Komponenten: *Configurator* und *Ontology Editor*. In der Wartung können diese beiden Komponenten genutzt werden, um Änderungen während der Laufzeit vorzunehmen. Sie beinhaltet auch Funktionen zur Überwachung der Funktionalität der AAL-Umgebung (z.B. AAL-Anwendungsfall deinstallieren, Sensorfunktion oder Batteriestatus überwachen).

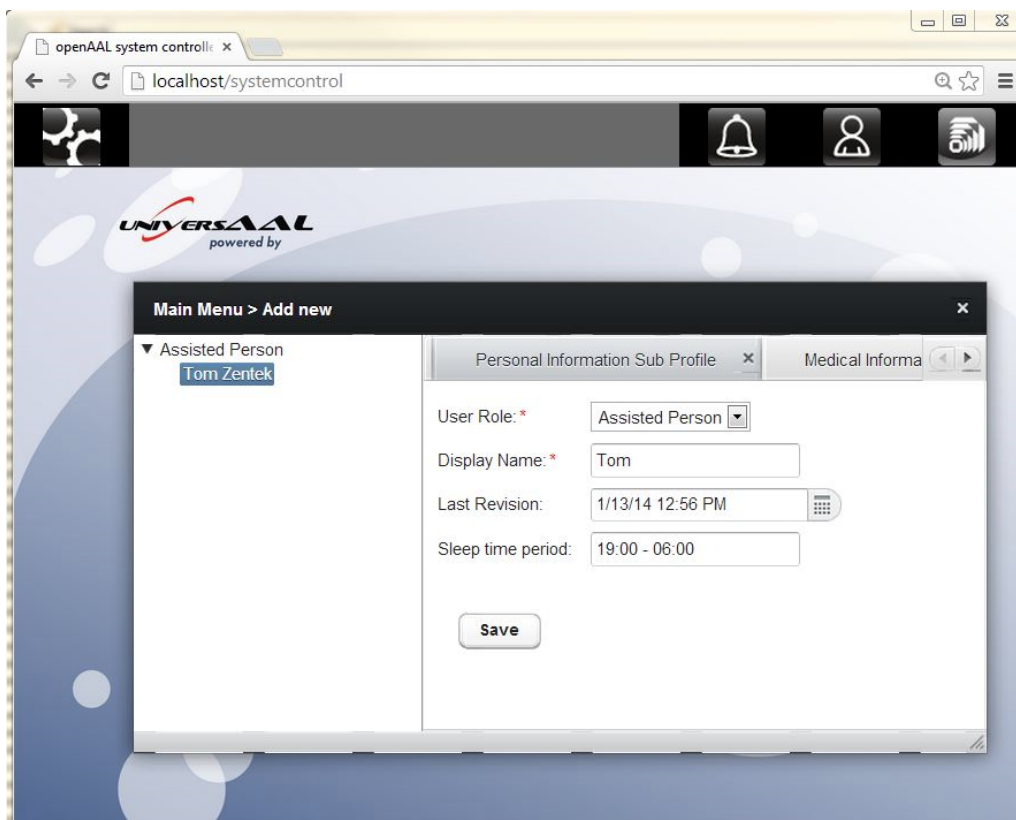


Abbildung 5.5: Darstellung des Control Centres und geöffnetem Ontologie Editor

Auf Abbildung 5.5 ist die mit dem GWT⁷ Derivat Vaadin⁸ entwickelte Oberfläche des CCs dargestellt. Links oben ist über die zwei Zahnräder das CC-Startmenü mit allen Funktionen zur Installation, Personalisierung und Wartung erreichbar. Rechts oben ist das Einstellungsmenü zu erreichen, in dem CC-interne Parameter gesetzt werden können. Weiterhin werden bei der Glocke aktuelle Mitteilungen und Warnungen angezeigt. In der Mitte sieht man den Ontologie Editor beim editieren der Assisted Person. Die einzelnen *SubProfiles* werden hierbei als Kateireiter dargestellt.

⁷Google Web Toolkit: <http://www.gwtproject.org/> (Zugriff am 21.11.2014)

⁸<https://vaadin.com/home> (Zugriff am 21.11.2014)

Die Abbildung 5.6 zeigt das Konfigurationsfenster zu der in der Entwicklung mit dem Configuration Editor (vgl. Abbildung 5.1) erstellten XML-Datei. Entsprechend der Konfigurationsdefinition des Entwicklers kann nun der Deployer die Konfigurationsinstanzen belegen.

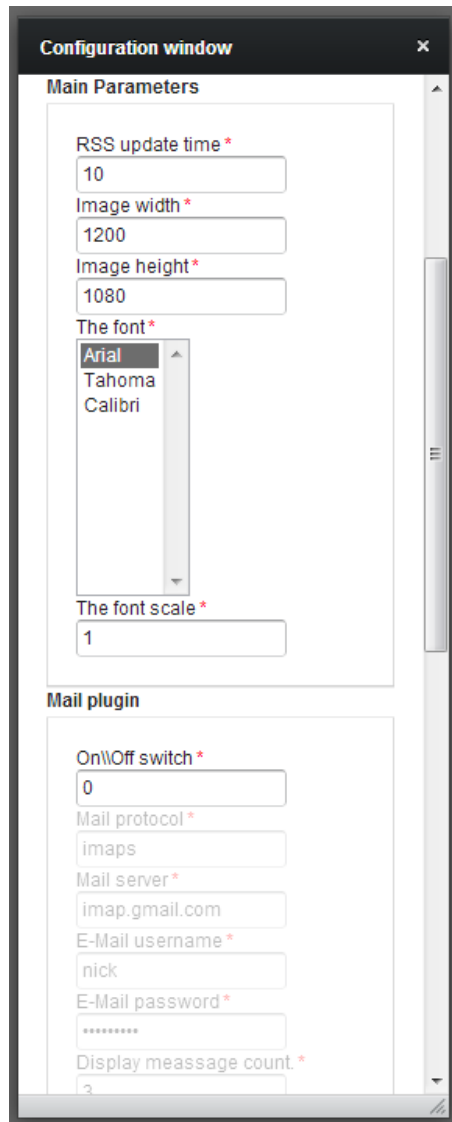


Abbildung 5.6: Darstellung des Konfigurationsfensters eines AAL-Service

Der CC selbst besteht aus den folgenden Komponenten, die sich untereinander über OSGi austauschen.

Configuration: Diese Komponente stellt dem Deployer dynamisch die Abfragen zur Personalisierung des Anwendungsfalls bereit.

Controller: Der **Controller** ist die Hauptschnittstelle zur Middleware für die Integration von neuen AAL-Services über eine API.

Database: Diese Komponente realisiert die Datenanbindung über einen Manager zum middlewareinternen Sesame⁹ Triplestore.

⁹Open Source Framework zum Abfragen und Analysieren von RDF. <http://www.openrdf.org/> (Zugriff am 21.11.2014)

Frontend: Diese Komponente beinhaltet die GUI des CC und ist die Schnittstelle zum Deployer. Sie wird mit dem Vaadin Framework in einem beliebigen Webbrowser ausgeführt.

Model: Die Model-Komponente unterstützt die Prozessierung der AAL-Anwendungsfälle durch die Installation und Personalisierung.

ProfileManager: Diese Komponente arbeitet eng mit der Database zusammen, da sie dynamisch die Sichten zu den Ontologiekonzepten und -instanzen erstellt. Sie nimmt auch neue Instanzen oder Änderungen entgegen, um sie über die Database persistieren zu lassen.

Nach diesem Überblick zum Aufbau des CCs soll die **Configuration**-Komponente als eines der zentralen Elemente genauer betrachtet werden. Wie Abbildung 5.7 zu entnehmen, ist zerfällt die **Configuration** in vier Komponenten. Weiterhin ist die Interaktion mit den Anwendungsfällen (**UseCase**) und dem Speicher (**physical storage**) in der Middleware abgebildet.

ConfigurationModel: Diese Komponente beinhaltet die Grundlagen des Konfigurationsframeworks. Hier sind die Klassen und Funktionen implementiert, die das Verarbeiten der Konfigurationsdefinitionen und -instanzen erlauben. Zur einfachen Verarbeitung und Erweiterbarkeit sind sie als XSD-Schema¹⁰ beschrieben. Weiterhin unterstützt das Modell das dynamische Laden der Listener und Validatoren durch entsprechende Factory-Methoden. Auf diese Weise können die vom Software-Entwickler des AAL-Anwendungsfalls mitgelieferten Funktionen in der Konfiguration ausgeführt werden. Zyklen und damit mögliche Mehrfachveränderungen bei der Listener-Auswertung werden durch die Konfiguration nicht abgefangen.

ConfigurationDefinitionRegistry: Diese Komponente dient zur Verarbeitung der Konfigurationsdefinitionen des AAL-Services (siehe Kapitel 4.3.5). Ein AAL-Service (**UseCase**) registriert hier seine Konfigurationsdefinitionen, der **Configurator** wird über Listener informiert und kann sie einlesen.

ConfigurationInstanceStorage: Diese Komponente realisiert die Speicherung aller Konfigurationsinstanzen, die entsprechend der Konfigurationsdefinitionen vom **Configurator** angelegt wurden. Ein AAL-Anwendungsfall kann direkt über ein Interface dieser Komponente auf seine Konfigurationsinstanzen zugreifen. Im Normalfall erhält er die primären Konfigurationsinstanzen zurück.

Configurator: Diese Komponente ist zentral im Konfigurationsframework. Durch sie kann der Deployer mit einer GUI die AAL-Services konfigurieren. Es kann ein AAL-Service aus der **ConfigurationDefinitionRegistry** gewählt werden und der **Configurator** generiert die passende Sicht und verknüpft die Eingaben soweit nötig mit Validatoren und Listnern. Weiterhin nutzt er den **ConfigurationInstanceStorage**, um die fertige Konfiguration zu speichern.

¹⁰<http://forge.universaal.org/svn/uaaltools/trunk/uCC/ucc.configuration.model/xmlFiles> (Zugriff am 21.11.2014)

Er ist auch für die Konfigurationsinstanz Metadaten verantwortlich und beeinflusst Parameter wie beispielsweise die Primär-, Sekundär und Notfallkonfigurationen. Werden die Konfigurationsdefinitionen erweitert, muss auch der **Configurator** erweitert werden.

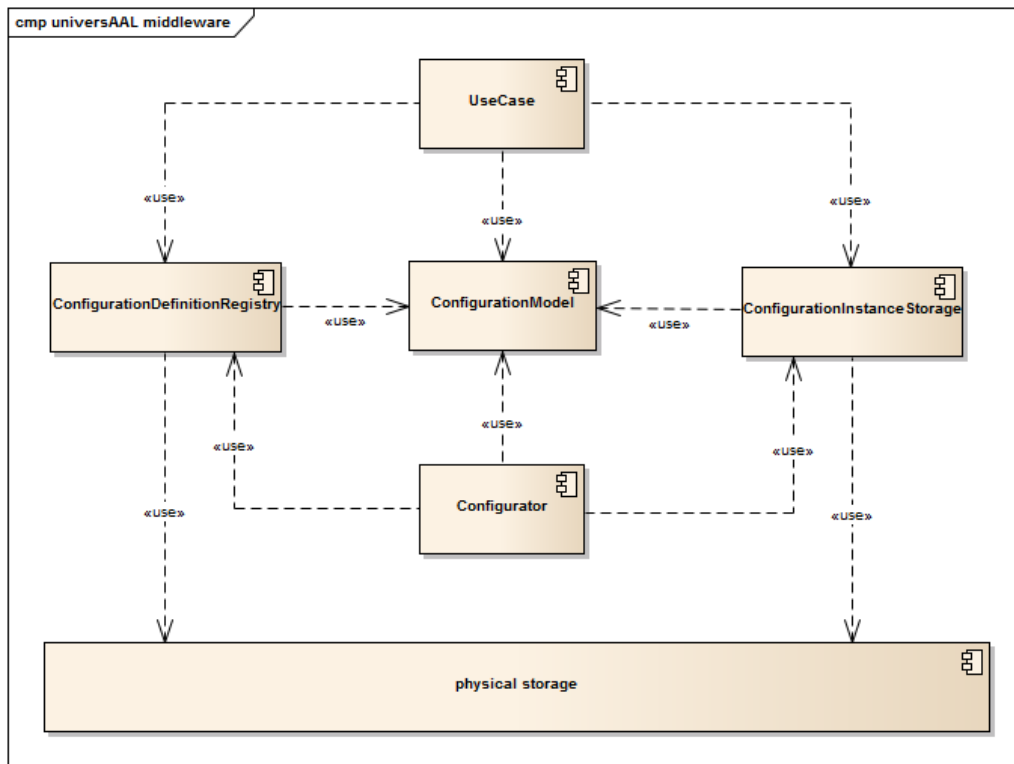


Abbildung 5.7: Komponentendiagramm der Configuration-Komponente

Die Abbildung 5.8 zeigt ein Sequenzdiagramm mit der Kommunikation zwischen den verschiedenen an einer erfolgreichen Personalisierung beteiligten **Configuration**-Komponenten. Die einzelnen Schritte sind hierbei in der Abbildung 5.8 nummeriert. Der **AAL-Service** ist bereits erfolgreich in der AAL-Umgebung installiert und wird nun durch den Deployer gestartet (1). Daraufhin registriert sich der **AAL-Service** und erhält seine default-Konfiguration (1 und 2). Jetzt ist er funktionsfähig und kann weiter konfiguriert werden (3). Hierzu öffnet der Deployer über den CC den **Configurator** (4), welcher seinerseits aus der **ConfigurationRegistry** alle verfügbaren AAL-Services ausliest (4.1 und 4.2). Der Deployer wählt den zu konfigurierenden AAL-Service aus (5) und gibt entsprechend dem Model die Parameter ein (6). Hierbei erhält er erstes Feedback auf seine Eingaben durch die mit den Parametern verbundenen Listener (7). Speichert er die Konfiguration des AAL-Services (8) wird die Validation über alle Parameter ausgeführt (8.1.1) und entsprechendes Feedback geliefert (8.1.2 und 9). Abschließend gibt der Deployer noch die Metadaten (z.B. Autor der Konfiguration, aktiv geschaltet) zur Konfiguration ein (10) und der AAL-Service ist fertig konfiguriert(11). Abschließend wird der AAL-Service über die neuen Konfigurationsparameter informiert (12) und lädt sie zur Laufzeit nach (13).

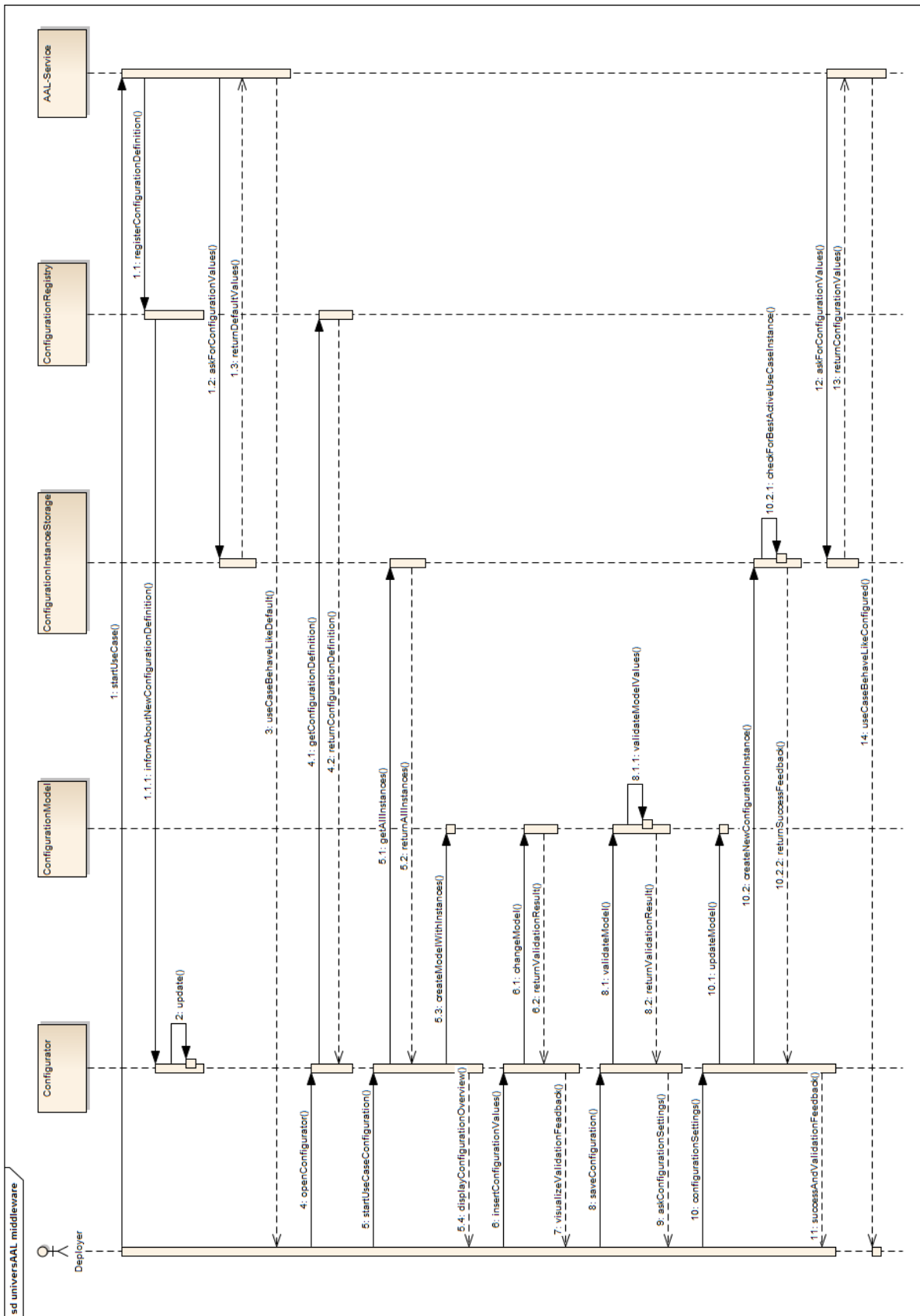


Abbildung 5.8: Sequenzdiagramm der Konfigurationsabläufe in der Personalisierungsphase

5.4 Dokumentation und Schulungsunterlagen

Zur Nutzung der vorgestellten Konzepte und dem dazugehörigen Framework sind Schulungen der verschiedenen beteiligten Rollen notwendig. Die Entwickler müssen lernen, wie sie ihre AAL-Services konfigurierbar machen und welche Werkzeuge ihnen dabei helfen. Die AAL-Anbieter müssen den Konfigurationslifecycle und die Werkzeuge, die sie in der Planung unterstützen können, verstehen. Die Gruppe der Deployer muss lernen, wie ein AAL-Service sich durch den Konfigurationslifecycle bewegt und wie sie ihn installieren, personalisieren und warten. Hierzu müssen sie im Umgang mit verschiedenen Werkzeugen geschult werden.

Es wurden vier verschiedene Schulungsmaterialien erstellt:

Referenzdokumentation: Die Referenzdokumentation beschreibt das Konfigurationsframework. Hierbei wird die Einbindung des Frameworks in die AAL-Umgebung und die AAL-Plattform dargestellt. Weiterhin wird beschrieben, an welchen Stellen sich der Konfigurationslifecycle im universAAL-Lifecycle wiederfindet. Die Kernkomponenten sind der *Configuration Editor*¹¹ für den Entwickler, der *Design Assistant* für den AAL-Anbieter und der *Control Centre*¹² für den Deployer.

Werkzeugdokumentation: Die Werkzeugdokumentation ist anwenderfokussiert und beschreibt primär die Nutzung der Werkzeuge im Framework: der *Configuration Editor*¹³ für das Erstellen der Konfigurationen für den Entwickler, der *Design Assistant* für das Entwerfen von Lösungen in der AAL-Umgebung durch den AAL-Anbieter und das *Control Centre*¹⁴ mit seinen Erweiterungen (*uStore*¹⁵, *Profile Editor*¹⁶) zum Installieren, Personalisieren und Warten der AAL-Services.

Trainingsmaterial: Das Trainingsmaterial ist neben den reinen Dokumentationen für eine interaktivere Art des Erlernens der Konfigurationskonzepte und Werkzeuge durch einen Trainer gedacht. Hierfür wurden verschiedene Materialien¹⁷ (z.B. Folien, Workshops, Videos) erstellt. In 20 verschiedenen Trainingsevents¹⁸ wurden diese Materialien über drei Jahre eingesetzt.

Weiterhin wurden die Lerninhalte zum Selbsttraining in Moodle¹⁹ bereitgestellt.

Open Source: Das gesamte Framework ist Open Source²⁰ unter der Apache Lizenz²¹ verfügbar. Die weitere Pflege wurde von der AALOA Community²²

¹¹http://forge.universaal.org/wiki/support:RD_Configuration (Zugriff am 21.11.2014)

¹²http://forge.universaal.org/wiki/support:RD_uCC (Zugriff am 21.11.2014)

¹³http://forge.universaal.org/wiki/uaaltools:Configuration_Editor (Zugriff am 21.11.2014)

¹⁴http://forge.universaal.org/wiki/uaaltools:UCC_-_configuration_and_personalization (Zugriff am 21.11.2014)

¹⁵http://forge.universaal.org/wiki/uaaltools:UStore_plugin (Zugriff am 21.11.2014)

¹⁶http://forge.universaal.org/wiki/uaaltools:Profile_Editor (Zugriff am 21.11.2014)

¹⁷<http://www.universaal.org/training/index.php/materials/modules> (Zugriff am 21.11.2014)

¹⁸<http://www.universaal.org/training/index.php/events> (Zugriff am 21.11.2014)

¹⁹<http://www.universaal.org/formation/> (Zugriff am 21.11.2014)

²⁰<http://forge.universaal.org/svn/uaaltools> (Zugriff am 21.11.2014)

²¹<http://www.apache.org/licenses/LICENSE-2.0> (Zugriff am 21.11.2014)

²²<http://www.aalooa.org/> (Zugriff am 21.11.2014)

übernommen, so dass es in weiteren Projekten Verwendung findet (z.B. EU-Projekt ReAAL²³).

5.5 Zusammenfassung

Die Implementierung erfolgte wie die Konzeption in zwei Iterationen entlang der DSR-Methode. In diesem Kapitel wurde das finale Ergebnis dargestellt.

Die Implementierung erfüllt im Zuge dieser Arbeit drei Aufgaben. Die erste ist die Umsetzung der erarbeiteten Konzepte in Framework zur flexiblen Konfiguration von AAL-Umgebungen. Die zweite Aufgabe ist, durch die Implementierung der Konzepte eine Basis für die Evaluation zu schaffen. Mit Hilfe des Frameworks und der Living Lab-Methode können die Konzepte und ihre Umsetzung umfassender präsentiert und sogar von den Teilnehmern der Evaluation ausprobiert werden. Die dritte Aufgabe der Implementierung ist es eine Referenzimplementierung für flexibles Management von AAL-Umgebungen zu schaffen.

Alle drei Aufgaben werden von der Implementierung des Konfigurationsframeworks in diesem Kapitel erfüllt. Hierzu wurden entlang der verschiedenen Phasen Werkzeuge zur Unterstützung der jeweils beteiligten Akteure entwickelt.

Der Entwickler hat mit dem „Configuration Editor“ und der „File Packager“ zwei Werkzeuge mit denen er in der Entwicklungsphase mit geringem Mehraufwand die Grundlage für flexibles Management legen kann. Er macht die AAL-Anwendungsfälle konfigurierbar.

In der Planungsphase wird der „Design Assistant“ eingesetzt, um entsprechend der Bedürfnisse des Endanwenders eine AAL-Lösung aus AAL-Anwendungsfällen und ihren Komponenten für die Ziel-AAL-Umgebung zu planen. Hier werden weitere Informationen gesammelt und für die folgenden Phasen bereitgestellt.

Der größte Teil des entwickelten Frameworks befindet sich in der AAL-Umgebung. Als Teil der Middleware unterstützt er den Deployer bei der Installation, Personalisierung und Wartung. Diese Funktionen werden im „Control Centre“ bereitgestellt. Auf Basis der konfigurierbaren AAL-Anwendungsfälle und den in der Planung gesammelten Informationen, kann die Umgebung jetzt flexibel gemanagt werden.

Weiterhin wurde zum Open Source Konfigurationsframework eine umfassende Dokumentation mit Schulungsunterlagen und Beispielen erstellt.

²³<http://www.cip-reaal.eu/home/> (Zugriff am 21.11.2014)

6. Evaluation des Konfigurationsframeworks

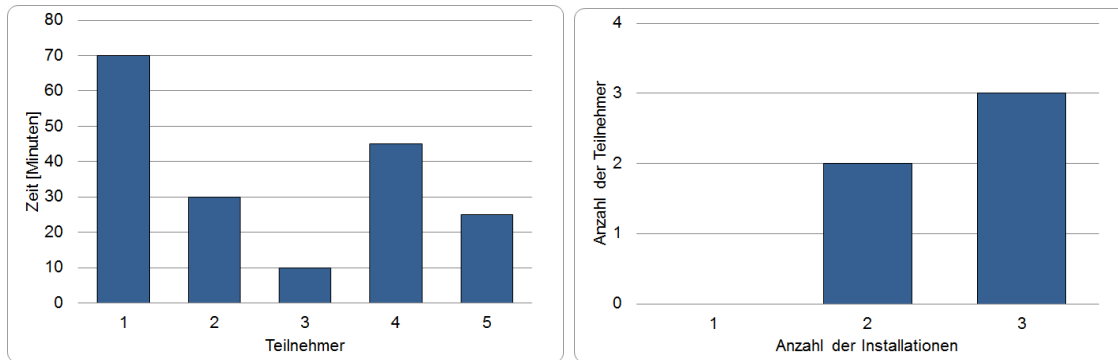
Dieses Kapitel präsentiert die zwei verschiedenen Evaluationen, die im Laufe dieser Arbeit entstanden sind. Die Validierungen im Rahmen der Fokusgruppen 1-4 sind nicht Teil des Evaluationskapitels und wurden bereits in den Kapiteln 3 und 4 vorgestellt. Die Evaluation ist die dritte Phase der DSR-Methode und dient der Prüfung des Konzeptes gegen die Forschungsfragen. Sie wurde in zwei Iterationen bearbeitet und bedient sich der Living Lab-Methode. Hierbei kamen die Fokusgruppen 05 und 07 zum Einsatz. Sie unterstützten bei der Planung und Vorbereitung sowie der Auswertung der Evaluationen.

In Kapitel 6.1 wird zuerst die Evaluation der Anforderungen besprochen, aus der eine Gewichtung für die weitere Entwicklung abgeleitet wurde. Hierzu wurde bereits der Prototyp aus der ersten Iteration des Konfigurationslifecycle eingesetzt. Dieses Kapitel evaluiert, ob die Forschungsfrage 1 nach den Anforderungen an ein flexibles Management ausreichend beantwortet wurde.

Das Kapitel 6.2 beschäftigt sich mit dem finalen Konfigurationslifecycle und seiner Frameworkimplementierung in einem Workshop, indem ihre Qualität evaluiert wurde. Diese Evaluation zielt auf die zweite Forschungsfrage nach den Vor- und Nachteilen einer ontologiebasierten Konfiguration.

6.1 Evaluation der Qualitätsanforderungen an das initiale Konfigurationsframework

Die in Kapitel 3.2 erarbeiteten Anforderungen sollten nach ihrer ersten Validierung in der Fokusgruppe 01 in diesem Schritt nach bestimmten Qualitätskriterien evaluiert werden. Aus dieser Evaluation lassen sich Handlungsanweisungen und die Relevanz einzelner Anforderungen für die Konzeptionsphase ableiten. Im Mittelpunkt dieser Evaluation stehen die Qualitätsanforderungen der Deployer mit dem Ziel, daraus Gewichtungen der Qualitätskriterien und der damit verbundenen Anforderungen abzuleiten. Auf dieser Basis kann dann der Konfigurationslifecycle für eine



(a) Zeitbedarf für die Integration des AAL-Services „Nutritional Advisor“ in den Living Labs (b) Anzahl der bereits durchgeführten Installationen von AAL-Services

Abbildung 6.1: Auswertung der Evaluation

flexible AAL-Umgebung geschaffen werden. Die gewichteten Anforderungen geben Leitlinien für die Konzeption und unterstützen die Entwicklung des Lifecycles.

Die folgenden Fragen werden in dieser Anforderungsevaluation behandelt:

E1.1: Wie wichtig ist den Deployern die Funktionalität, Effizienz, Benutzbarkeit und die Zuverlässigkeit bei der Integration von AAL-Services?

E1.2: Wie sieht ein Ranking bezüglich dieser Qualitätskriterien aus?

Bisher sind für die Konfiguration von AAL-Umgebungen keine passenden Qualitätsmodelle entwickelt worden. Um eine AAL-Umgebung mit Blick auf Flexibilität zu evaluieren, musste das viel genutzte Product Quality Model (PQM) [ISO/10] an die AAL-Qualitätsziele angepasst werden. Darum wurde im Zuge der Evaluation ein eigenes AAL-Qualitätsmodell entwickelt. Auf Basis einer Demonstration des Konfigurationsframeworks und -lifecycles wurde von den Teilnehmern ein Fragebogen für die Qualitätskriterien beantwortet.

Die Anforderungsevaluation präsentiert zuerst die Ergebnisse des Pretests (siehe Kapitel 6.1.1). Anschließend beschreibt Kapitel 6.1.2 die Ergebnisse der Evaluation. Die verschiedenen Meinungen werden dargestellt und zu einem Ranking der Qualitätskriterien verbunden. Abschließend werden die Ergebnisse in Kapitel 6.1.3 eingehend diskutiert und Implikationen des Rankings auf die Anforderungen aufgezeigt.

6.1.1 Ergebnisse des Evaluationstests

Ziel des Evaluationstests war es, Fehler und Verständnisprobleme im Ablauf der Integration eines Anwendungsfalls zu finden. Hierbei wird die Living Lab-Methode mit den Fragebogen kombiniert. Dies erreicht eine detaillierte Auseinandersetzung mit den Konfigurationskonzepten und dem -framework bevor die Bewertung in den Fragebogen festgehalten wird. Mit Hilfe der Konfigurationskonzepte und des -frameworks wurde der „Nutritional Advisor“ entlang des Konfigurationslifecycles in die AAL-Umgebung (Living Lab) integriert.

Laut Frage Q_{pre5} konnte die Integration des Nutritional Advisors von allen Teilnehmern ($m=5$; $w=0$) erfolgreich umgesetzt werden. Der Zeitbedarf der einzelnen Installationen ist in Abbildung 6.1a dargestellt (vgl. Frage Q_{pre7}). Teilweise kam es zu einer langen Integrationszeit ($\varnothing 36 \pm 20, 35min$) und einmal wurde Unterstützung in Form von telefonischem Support benötigt (vgl. Frage Q_{pre9}). Frage Q_{pre3} und Q_{pre4} ist zu entnehmen, dass alle Teilnehmer bereits Erfahrungen mit der Installation von AAL-Services in der Middleware hatten. Die verschiedenen Erfahrungswerte sind in Abbildung 6.1b dargestellt.

Q_{pre6} , Q_{pre10} und Q_{pre11} fragen nach Problemen in der Dokumentation und bei der Integration des AAL-Anwendungsfalls. Die Teilnehmer des Evaluationstests gaben die folgenden Rückmeldungen: Es wurden an ein einigen Stellen Verständnisprobleme und Fehler bei den Abläufen des Lifecycles identifiziert (z.B. Step 2.5 download of the service; Step 3.3 This source folder is not existing!). Weiterhin wurden einzelne Sprachdateien während der Personalisierungsphase vermisst (z.B. Step 3.2 There are only menu files in en, es, hr. My system is German! -> de).

Weiterhin gab es von fast allen Teilnehmern generelle Anmerkungen (Frage Q_{pre12}). Hierdurch wurden kleine Fehler in der Lifecycle-Ablaufdokumentation aufgedeckt. Weiterhin wurde beanstandet, dass der Download des AAL-Services teilweise sehr langsam ist. Zwei Mal wurde auch der Wunsch geäußert den Lifecycle mehr zu automatisieren. Einige weitere Fragen zeigten Klärungsbedarf bei den Konfigurationskonzepten.

Die Fragen Q_{pre0} nach dem Namen diente für eventuelle Rückfragen. Q_{pre1} und Q_{pre2} wurden wegen mangelnder Relevanz nicht ausgewertet.

Diskussion:

Für die Evaluation der Qualitätsanforderungen an das Konfigurationsframework konnte durch den Pretest wichtiges Feedback gesammelt werden. In der Fokusgruppe 05 wurden die einzelnen Aspekte diskutiert und anschließend die Hauptevaluation, wo nötig, angepasst.

Ein wichtiges Thema war die Stabilität der Präsentation im Living Lab. Während des Pretests hatten einige Teilnehmer mit dem Herunterladen des Anwendungsfalls, durch langsame Verbindungen oder teilweise Ausfall der Webseite Probleme.

Weiterhin konnte aufgrund der Kommentare zu den Fragen $Q_{pre10-12}$ Probleme in der Verständlichkeit der Lifecycle Prozesse identifiziert und behoben werden. Aus diesen Kommentaren leitet sich der Bedarf ab, die Integrationsprozesse weiter zu verschlanken und zu automatisieren.

Insgesamt gab es keine Kritik am methodischen Aufbau der Evaluation. Dies wurde in den Einzelgesprächen bestätigt. Auf Basis der Ergebnisse wurde die Präsentation der Prozesse und Werkzeuge nochmals angepasst. Auch der Ablauf der Demonstration wurde um drei Schritte (Entwicklerunterstützung, Wartungsmöglichkeiten, Deinstallation) erweitert und verständlicher gestaltet.

6.1.2 Ergebnisse der Anforderungsevaluation

Nach den Anpassungen der Anforderungsevaluation auf Basis der Erfahrungen aus dem Testlauf wurde mit der Evaluation begonnen. Die Ergebnisse der Evaluation beruhen auf Fragebogen (siehe Kapitel 2.3.1), die von 27 der 32 Teilnehmer ausgefüllt

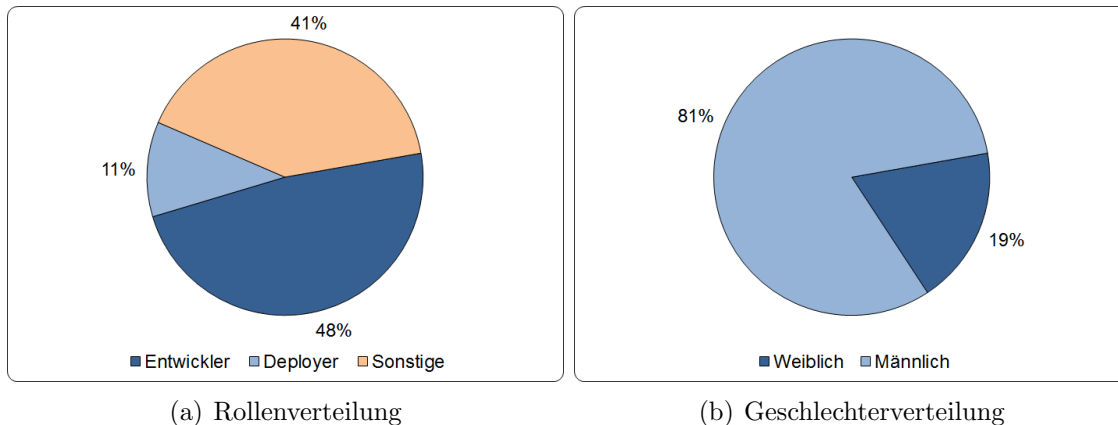


Abbildung 6.2: Überblick zu den Teilnehmern (I)

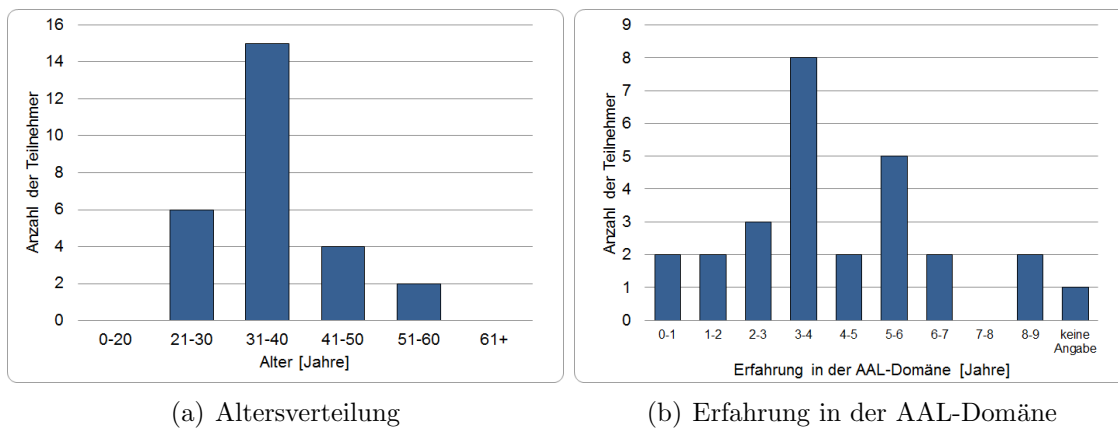


Abbildung 6.3: Überblick zu den Teilnehmern (II)

wurden. Hierbei wurden 23 in Papierform ausgefüllt und 4 Teilnehmer entschieden sich für den elektronischen Weg. Somit ist nach Nielsen eine statistische Belastbarkeit der abgeleiteten Erkenntnisse möglich. Die Teilnehmer schätzen sich hierbei von ihrem beruflichen Hintergrund hauptsächlich als Developer (48%) ein, gefolgt von Deployern mit 11% (vgl. Abbildung 6.2a). Aus den Daten ergibt sich ein Durchschnittsalter von 36 Jahren (vgl. Abbildung 6.3a) bei fast 20% Frauenanteil (vgl. Abbildung 6.2b). Bis auf vier Teilnehmer erfüllen alle die Mindestanforderung zur Anforderungsevaluation aus Deployersicht. Die durchschnittlich Erfahrung in der AAL-Domäne betrug 3,6 Jahre (vergl. Abbildung 6.3b).

Kernergebnis dieser Evaluation ist die Vorgabe der Qualitätskriterien für die Anforderungen. Für die in Kapitel 2.3.1 beschriebene AHP-Methode konnten hierbei nach der Konsolidierung 23 der 27 Datensätze genutzt werden. Zwei mussten aufgrund von Unvollständigkeit und zwei wegen zu geringer Konsistenz verworfen werden.

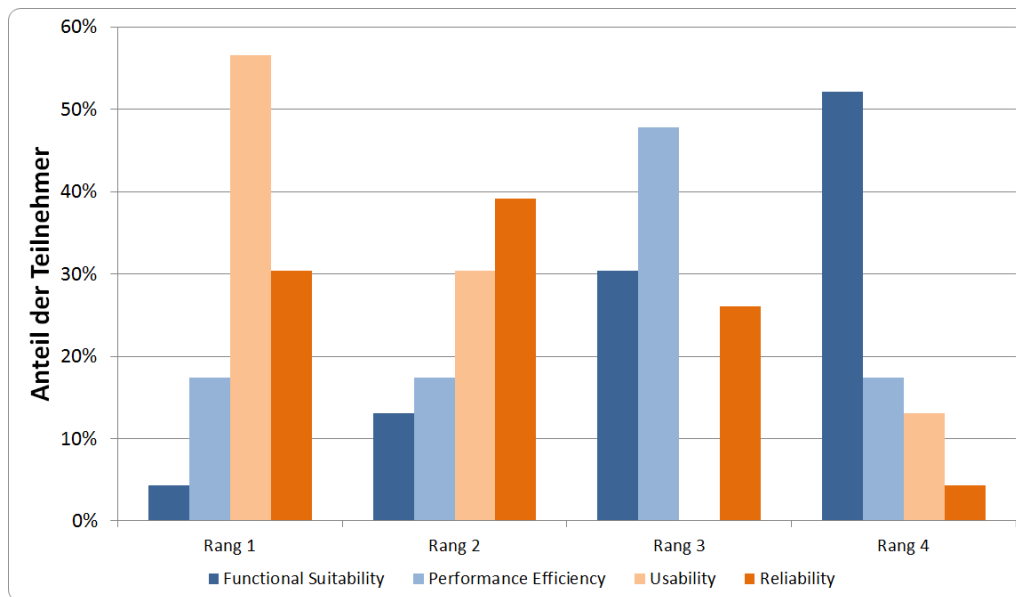


Abbildung 6.4: Prioritätsverteilung der Einzelwertungen der Teilnehmer. Rang 1 stellt die wichtigste Anforderungsrichtung dar und Rang 4 die unwichtigste.

Für verbleibende 23 Datensätze ergibt sich die folgende Matrix. Die ausführlichen Berechnungen zur AHP finden sich in Anhang A.4.

$$AHP = \begin{pmatrix} 1 & 1 & 2,5 & 1 \\ 1 & 1 & 2 & 2 \\ 0,4 & 0,5 & 1 & 1 \\ 1 & 0,5 & 1 & 1 \end{pmatrix} \quad (6.1)$$

Daraus ergeben sich die folgenden geschätzten Eigenwerte:

$$\lambda_1 = 1,21; \lambda_2 = 1,34; \lambda_3 = 0,64; \lambda_4 = 0,81 \quad (6.2)$$

Der dazugehörige Prioritätsvektor ist:

$$P = \begin{pmatrix} 0,303 \\ 0,334 \\ 0,160 \\ 0,204 \end{pmatrix} \quad (6.3)$$

Für die Konsistenzprüfung ergeben sich die folgenden Werte:

$$\lambda_{max} = 4,0868; CI = 0,0289; CR = 0,03213 \quad (6.4)$$

Wie in Abbildung 6.4 zu sehen, bildet sich das folgende Ranking:

1. Benutzbarkeit (Usability)
2. Zuverlässigkeit (Reliability)

3. Effizienz (Performance Efficiency)

4. Funktionalität (Functional Suitability)

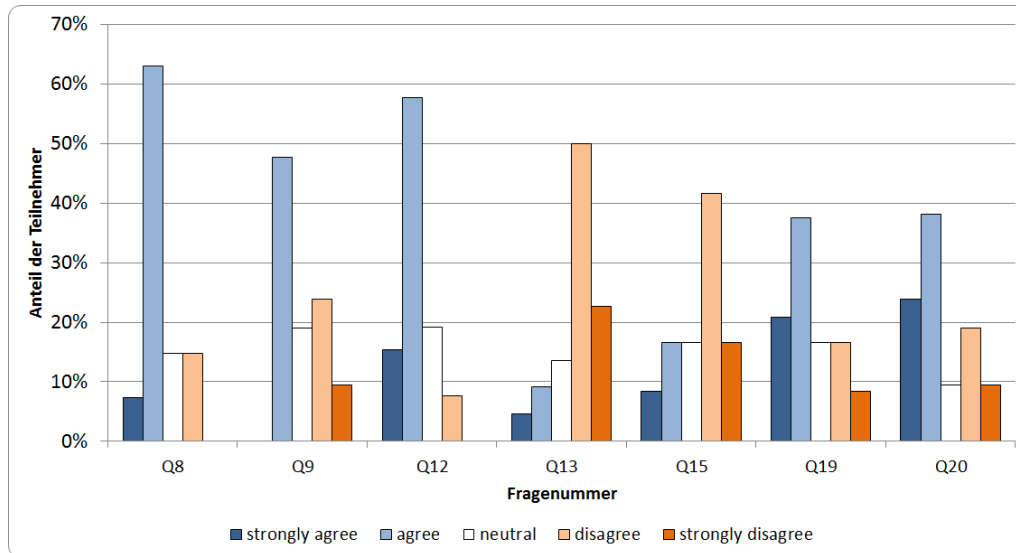


Abbildung 6.5: Überblick zu allen Fragen der Gruppe 1, die hauptsächlich für die AHP-Auswertung genutzt wurden.

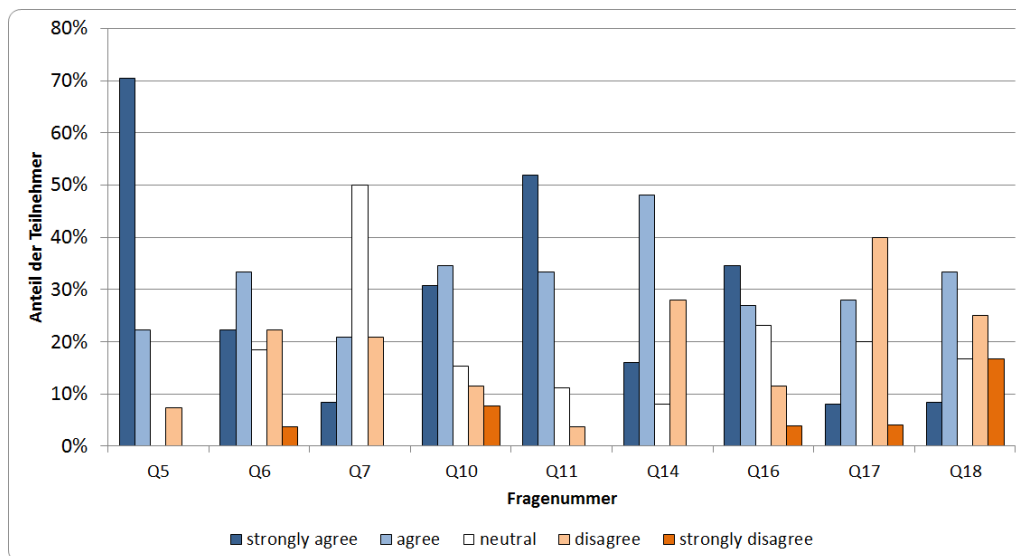


Abbildung 6.6: Überblick zu allen Fragen der Gruppe 2, die zur Kontrolle der Gewichtung aus der AHP-Auswertung genutzt wurden.

Einzelne Antworten der Teilnehmer auf Fragen der Gruppe 2 (Kontrollfragen der AHP, vgl. Kapitel 2.3.1) bestätigen die Ergebnisse der AHP und boten weitere wichtige qualitative Einsichten in die Anforderungen des Deployers. Einige Fragen, die aus den Befragungsergebnissen besonders herausstechen, sollen im Folgenden näher betrachtet werden (vgl. Abbildung 6.5 und 6.6):

- Q5: Die Deployer wollen eine „One-Click-Installation“. Dies bedeutet, dass ein AAL-Anwendungsfall aus dem online store (uStore) mit einem einzigen Mausklick in die AAL-Umgebung integriert werden kann. 93% der Teilnehmer stimmen dieser Aussage zu. Der Fokus liegt auf der Benutzbarkeit und Effizienz während Funktionalität in den Hintergrund tritt.
- Q6: Diese Frage wurde von den Teilnehmern mit keiner klaren Tendenz beantwortet. Es ist eine schwache Zustimmung aus den Antworten abzulesen.
- Q7: 50% wählten hierbei eine neutrale Antwort. Die Antwort wurde teilweise so kommentiert, dass anzunehmen ist, dass die Frage, auch nach der Demonstration, nicht für alle Teilnehmer verständlich war.
- Q8: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung zeigt diese Frage mit 70% den starken Wunsch der Teilnehmer nach Flexibilität.
- Q9: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung befürworteten 48% der Teilnehmer diese Art von Effizienz.
- Q10: Die Developer fordern, dass der Konfigurationslifecycle geografisch unabhängig und auf mobilen Endgeräten mit den dazugehörigen Werkzeugen möglich ist. 65% der Teilnehmer stimmen dieser Aussage zu.
- Q11: Unterstützung durch mitgelieferte Standardkonfigurationen zur Unterstützung des Deployers ist die zweitstärkste Forderung mit 85%. Hierbei zeigt sich erneut die Wichtigkeit des Benutzbarkeitsaspektes und auch der Zuverlässigkeit.
- Q12: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung stimmen 73% der Aussage zu. Die Teilnehmer wünschen sich einen guten Fehlerschutz, was für Zuverlässigkeit spricht.
- Q13: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung wünschen sich 73% der Teilnehmer eine Validierung der Konfigurationseingaben, um Probleme in der Ausführung der AAL-Anwendungsfälle zu vermeiden.
- Q14: 64% der Teilnehmer finden eine Undo Funktionalität im Konfigurationsframework wichtig. 28% finden diese Funktion nicht wichtig.
- Q15: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung lehnen 58% Komplexität zugunsten der Benutzbarkeit ab.
- Q16: 62% der Teilnehmer wollten, dass der Konfigurationslifecycle sich auch auf angrenzende Wartungs- und Verwaltungsaufgaben bezieht. Nur 15% sind gegen diesen erweiterten Aufgabenraum und somit mehr Flexibilität.
- Q17: Über die Frage der Systemanforderungen eines Konfigurationsframeworks konnten die Befragten kein eindeutiges Stimmungsbild abgeben. 36% erwarteten ein ressourcenschonendes Framework und 44% nehmen einen höheren Ressourcenverbrauch für diese Unterstützung in Kauf. Die restlichen 20% antworteten neutral.

- Q18: Auf die Frage nach Schulungen zur Konfiguratornutzung (uCC) zeichnet sich kein klares Bild. Sowohl Ablehnung wie Befürwortung erreichen 42%. Die vorgeführte Demonstration des Konfigurationslifecycles wirkte also auf die Teilnehmer unterschiedlich schwer.
- Q19: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung wird diese Frage mit 58% Zustimmung eher bestätigt. Die Benutzbarkeit ist wichtiger für die meisten Teilnehmer als die Effizienz.
- Q20: Diese Frage wurde im Zuge der AHP-Auswertung genutzt. Unabhängig von der AHP-Auswertung wird mit 62% die Verwaltung von Hardware und menschlichen Resource gefordert. Hierbei wird sogar akzeptiert, wenn es zu einem Verlust an Bedienkomfort kommt.

Weitere Kommentare im Freitext (Q21) lassen sich wie folgt zusammenfassen. Bezugnehmend auf die Demonstration den Konfigurationslifecycles wurde der Wunsch nach weiteren „helping and simplifying tool[s]“ geäußert. Diese helfenden und vereinfachenden Werkzeuge für die verschiedenen beteiligten Akteure bezogen sich auf die Hardware und das damit verbundenen Management. Die Unterstützung der Softwarekomponenten scheint ausreichend. Ein weiterer Punkt der mehrfach aufgegriffen wurde, war die Forderung nach einer Multiplattformunterstützung der Werkzeuge besonders in der Planungsphase des Lifecycles. Weiterhin wurde angesprochen, dass ein Fernzugriff für verschiedene Akteure und deren Arbeiten im Lifecycle sinnvoll wären.

6.1.3 Diskussion der Ergebnisse

Es hat sich gezeigt, dass die angepassten ISO Qualitätsmerkmale sehr gut im AAL-Bereich zu verwenden sind. Sie decken nach Einschätzung der Teilnehmer die AAL-relevanten Merkmale ab. Die ausführlichen Pretests in den fünf Living Labs und die die Unterstützung während der Evaluationskonzeption durch Fokusgruppe 05 haben eine gute Evaluation gesichert. Durch ihre Vorarbeit konnte eine ansprechende, verständliche und umfassende Demonstration sichergestellt werden. Nur vereinzelt kommt es zu Verständnisproblemen (Q7), wie einem Kommentar zu entnehmen ist.

Die Durchführung und anschließende Auswertung der Evaluation liefert erfolgreich eine Gewichtung der Anforderungen. Für die weitere Entwicklung des Konfigurationslifecycles gibt es auf Basis der AHP-Ergebnisse klare Vorgaben, die durch die Fragen der Gruppe 2 bestätigt wurden. Die Benutzbarkeit (Usability) und Zuverlässigkeit (Reliability) sind für den Deployer am Wichtigsten. In der weiteren Entwicklung des Frameworks sind somit Anforderungen die Benutzbarkeit und Zuverlässigkeit steigern deutlich in den Vordergrund zu stellen. Von eher geringeren Priorität sind Effizienz (Performance Efficiency) und Funktionalität (Functional Suitability) für die zweite Iteration des Konfigurationslifecycles. Sie werden wie Kompatibilität (Compatibility), Sicherheit (Security), Wartbarkeit (Maintainability) und Übertragbarkeit (Portability) im Lifecycle und dem Framework sekundär eingebaut.

Während dieser Evaluation konnte aber auch festgestellt werden, dass eine noch intensivere Auseinandersetzung der Teilnehmer mit den Prozessen des Konfigurationslifecycles wünschenswert wäre. Es kam vereinzelt zu Verständnisproblemen, die

nur teilweise während der Demonstration und im Anschluss vor der Beantwortung der Fragen geklärt werden konnten. Allerdings ist es schwierig, eine Gruppe von über 30 Teilnehmern besser zu schulen. Hierfür war die vorherige Validierung in den Fokusgruppen 1-4 mit intensiveren Schulungen und der Möglichkeit, selbst Prozesse und Werkzeuge des Lifecycles auszuprobieren, eine gute Ergänzung. Falls notwendig, wurden diese Kleingruppenbewertungen mit verschiedenen Teilnehmern wiederholt, um belastbare Werte zu erzeugen.

Durch diese erfolgreiche Evaluation konnten gewichtete Anforderungen und Leitlinien für die Weiterentwicklung des Konfigurationslifecycles in der zweiten Iteration gefunden werden.

Die Frage E1.1 nach einzelnen Qualitätsanforderungen und ihrer Wichtigkeit im Konfigurationsframework konnte durch den Fragebogen von allen Seite beleuchtet werden.

Die Frage E1.2 nach den gewichteten Qualitätsanforderungen an das Konfigurationsframework konnte durch die AHP-Analyse beantwortet werden. Das Ergebnis ist: Benutzbarkeit > Zuverlässigkeit > Effizienz > Funktionalität.

6.2 Evaluation des finalen Konfigurationsframeworks

Ziel dieser Evaluation ist es, den fertigen Konfigurationslifecycle sowie seine Implementierung als Konfigurationsframework zu bewerten. Die Grundlage bildet der in Kapitel 4.1 dargestellte Lifecycle in seiner finalen Version und der finale Prototyp des Frameworks, genannt „Vaadin“ (vgl. Kapitel 5). Die finale Evaluation erfolgte in einem halbtägigen Workshop im Begleitprogramm des 6. AAL-Kongresses in Berlin. Zur Bewertung der Qualität des Lifecycles und des Frameworks wurde die Quality-in-Use-Metrik ausgewählt. Sie lässt die notwendigen Freiheiten, um auf alle Besonderheiten der AAL-Domäne und ihrer Nutzer einzugehen und stellt trotzdem ein Rahmengerüst, um alle wichtigen Qualitätsparameter zu betrachten.

Die zentrale Frage der Frameworkevaluation ist:

E2.0: Wie bewerten Entwickler und Deployer die Qualität der wissensbasierte Konfiguration und ihre Umsetzung als Framework und Lifecycle?

Die Quality-in-Use-Methode erlaubt die Konfiguration in den Dimensionen Effektivität, Effizienz, Zufriedenheit und Vollständigkeit zu untersuchen. Hierbei werden nach einer Schulung und Demonstration des Frameworks und des Lifecycles die Teilnehmer selbst die Konzepte und Prozesse erfahren, indem sie in der Rolle des Entwicklers und des Deployers selbst zwei Aufgaben bearbeiten. Abschließend wird in einer Gruppendiskussion und einem Fragebogen das Feedback eingeholt.

6.2.1 Ergebnisse der finalen Evaluation

Die Ergebnisse des Evaluationsworkshops werden in diesem Kapitel zusammengefasst. Insgesamt nahmen 10 Probanden [$w=2$; $m=8$] (siehe Abbildung 6.7a) an dem Workshop teil. Eine Person hat sich jedoch nicht aktiv an der Hands-On-Phase und

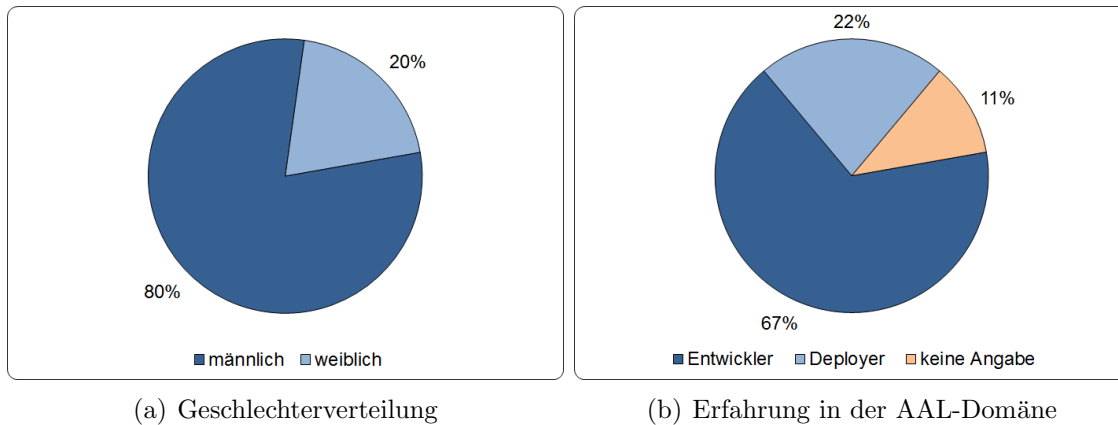


Abbildung 6.7: Überblick zu den Teilnehmern (I)

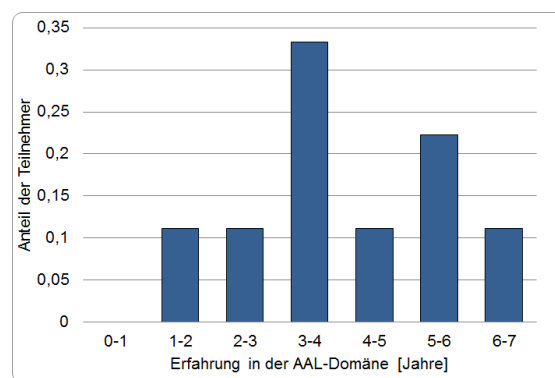


Abbildung 6.8: Überblick zu den Teilnehmern (II) - AAL-Erfahrung in Jahren

dem Fragebogen beteiligt und konnte somit in diesen Teilen der Auswertung nicht berücksichtigt werden.

Die Teilnehmer bestanden zum Großteil aus Personen, die primär AAL-Anwendungsfälle entwickeln (67%; vgl. Abbildung 6.7b). Der entwicklungslastige Teil des Konfigurationslifecycle ist also durch ihre Kompetenzen abgedeckt. 22% der Teilnehmer gaben an, Deployer zu sein. Da auch Entwickler die Deployer-Werkzeuge nutzen, um ihre Software zu debuggen oder zu testen, ist auch der Deployeranteil abgedeckt.

Die Erfahrung der Teilnehmer im AAL-Umfeld in Jahren kann in Abbildung 6.8 entnommen werden. Hierbei ist hervorzuheben, dass kein Teilnehmer weniger als 1,5 Jahre in AAL tätig war und der Durchschnitt bei 3,61 Jahren lag. Die Erfahrung der Domänenexperten kann somit als fortgeschritten beschrieben werden und erlaubt die Annahme, dass sich alle Workshopteilnehmer sowohl in die Entwickler- als auch Deployerrolle hineinendenken können.

Die Ergebnisse der Evaluation werden im Folgenden gemäß ihrer Qualitätskriterien im Sinne der Quality-in-Use präsentiert.

6.2.1.1 Effektivität

Die Erfolgsrate stellt den zentralen Parameter zur Messung einer richtig gelösten Aufgabe (vgl. Kapitel 2.3.2 Frage E2.1.1). Hierbei wurde definiert, dass jede Teil-

aufgabe korrekt gelöst ist, sobald alle in ihr beinhalteten Elementaraufgaben ohne Fehler durchgeführt wurden. Diese Frage wird für die Prozesse der Entwickler und Deployer einzeln untersucht.

In Abbildung 6.9 sind die verschiedenen Erfolgsraten in ihrer Zuordnung zu den einzelnen Teilaufgaben für die Entwickler (Step1) und Deployer (Step2) dargestellt. Es gab zwei Aufgaben zu bearbeiten, die jeweils in drei Teilaufgaben zerfallen. Diese Teile wiederholen sich in Aufgabe 1 und 2. Insgesamt steigt die Komplexität der einzelnen Teilaufgaben langsam an. In Abbildung 6.9 ist zu sehen, dass die Nutzung von Validatoren Probleme bereitet. Der Umgang mit normalen Konfigurationselementen und Listnern gestaltet sich dagegen einfacher. Mit steigendem Schwierigkeitsgrad der einzelnen Aufgaben scheitern mehr Teilnehmer an den Aufgaben. Die schwerste Teilaufgabe (vgl. Abb. 6.9 Aufgabe 2: Listener implementieren) wurde entweder erfolgreich gelöst oder gar nicht erst bearbeitet. Es ist anzunehmen, dass dies auf die Programmierfähigkeiten der Teilnehmer zurückzuführen ist.

Nach den einzelnen Programmieraufgaben wurde jede erzeugte Änderung des „Inframes“ mit Hilfe des Konfigurators aus dem Framework installiert und konfiguriert. Der Konfigurationserfolg aus Deployer-Sicht ist jeweils in Abbildung 6.9 Step 2 dargestellt. Hierbei ist zu beachten, dass nur nach erfolgreicher Entwicklung auch eine Integration des AAL-Services in die Plattform vorgenommen werden konnte.

Die Leistungsfähigkeit der Konzepte wird durch den Fragebogen (vgl. Kapitel 2.3.2 Frage E2.1.2) mit der Frage Q5 geprüft. Neben zwei Enthaltungen bewerten die Domänenexperten die Leistungsfähigkeit der Konzepte als eher hoch (vgl. Abbildung 6.10).

Die Evaluation der Ergebnisse erfolgte über die Anzahl und Art der Probleme (vgl. Kapitel 2.3.2, Frage E2.1.3), die während der Hands-On-Phase aufgetreten sind. Sie wurden mit Hilfe des Videomaterials genau evaluiert und die häufigsten Probleme in Abbildung 6.11 dargestellt. Es ist zu sehen, dass 27% der Probleme beim Entwickeln der konfigurierbaren Anwendungsfälle aufgetreten sind und sich 73% der Probleme auf Bedienfehler des Konfigurators im Framework beziehen. Die Analyse des Videomaterials ergab weiterhin, dass 82% der Teilnehmer die meisten dieser Probleme selber beheben konnten. Die restlichen Fehler konnten mit Hilfe der Betreuer gelöst werden. Es gab also keine unlösbaren Probleme während des Konfigurationslifecycle.

Zuerst sollen die Probleme im Deployteil (Step 2) der einzelnen Teilaufgaben genauer beleuchtet werden:

Keine Schrift ausgewählt: Am meisten Probleme kamen bei der Auswahl der Schriftart während der Konfiguration des „Inframes“ vor. Innerhalb der vorbereiteten Konfigurationsdatei gab es ein MapConfigItem, welches zur Bestimmung der Schriftart dient. Wertet der Konfigurator diese Datei aus, muss vom Teilnehmer neben anderen Parametern explizit die Schriftart gesetzt werden. Wird die Schriftart nicht gewählt, erkennt dies ein Validator und gibt eine Information an den Teilnehmer aus. Auf diese Weise waren die Teilnehmer immer in der Lage, den Fehler durch Auswahl der Schriftart nachträglich zu beheben.

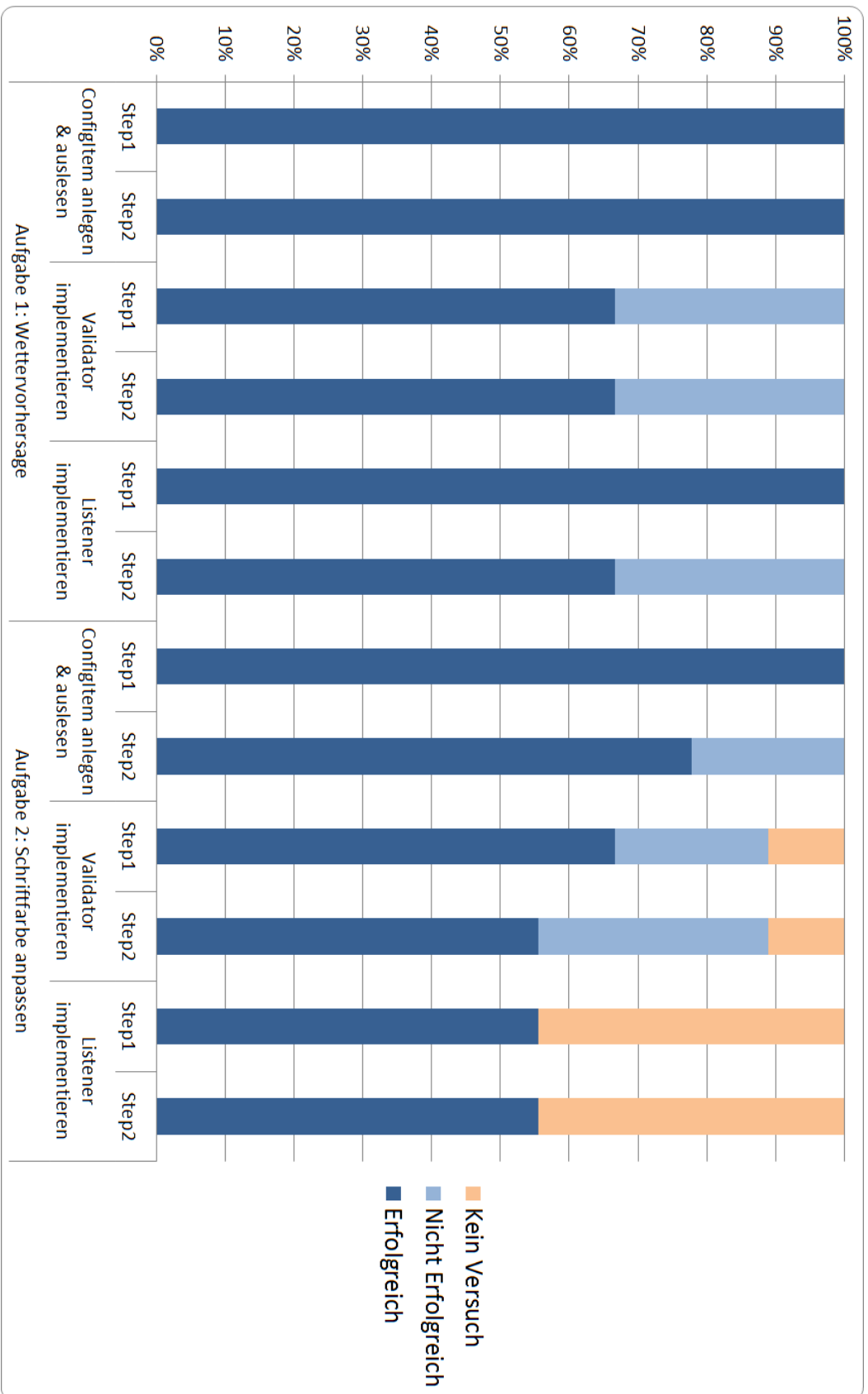


Abbildung 6.9: Auswertung zur erfolgreichen Lösung der Ausgaben 1 und 2 der Hands-On-Phase (n=9)

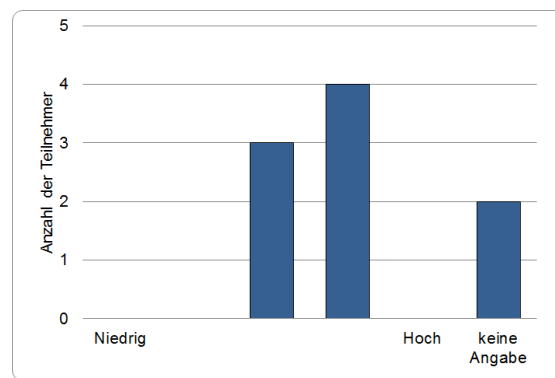


Abbildung 6.10: Q5 Leistungsfähigkeit: Können Sie Ihre AAL-Services mit den vorgestellten Konfigurationskonzepten umsetzen?

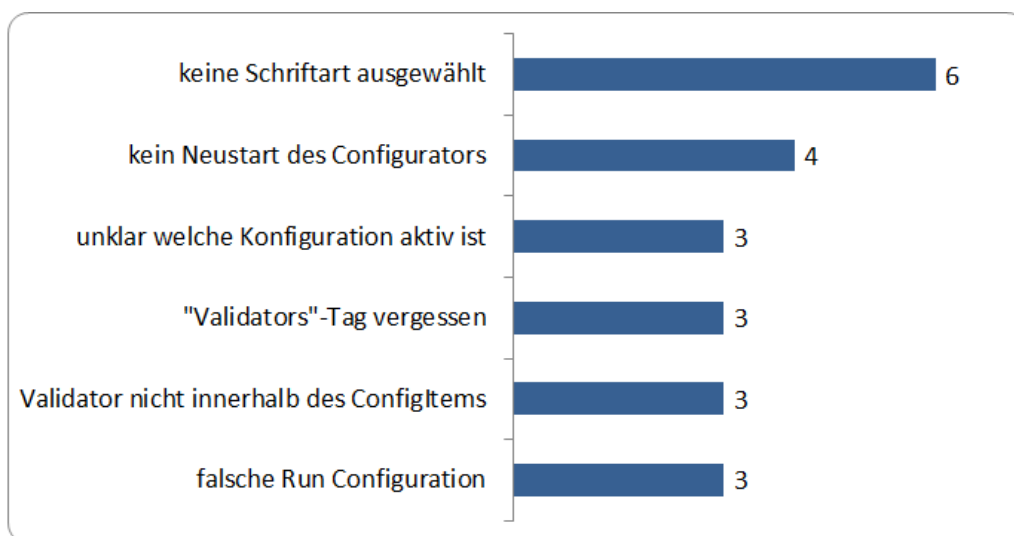


Abbildung 6.11: Erfasste Bedienfehler auf Basis der Videoanalyse (n=22)

Kein Neustart des Configurators: Bei einigen Aufgaben musste der Konfigurator neu gestartet werden, um die Konfigurationsänderungen des „Infoframes“ neu einzulesen. Besonders, wenn nur kleinere Änderungen an XML-Dateien vorgenommen wurden, war dies nicht immer für alle Teilnehmer ersichtlich. Trat dieses Problem auf, konnte es meist selbst gelöst werden.

Unklar, welche Konfiguration aktiv ist: Der Konfigurator ist in der Lage, mehrere Konfigurationen zu einem Anwendungsfall zu speichern. Für die Teilnehmer war es nicht immer ersichtlich, welche die aktive Konfiguration ist bzw. welche zu parametrisieren ist. Brachte eine Instanziierung nicht den gewünschten Effekt, wählten die Probanden danach meist die richtige Konfiguration aus.

Falsche Run Configuration: Damit der konfigurierbare AAL-Service „Infoframe“ und der Konfigurator geeignet miteinander ausgeführt werden, muss die richtige Run Configuration in der Entwicklungsumgebung gestartet werden. Hierbei kam es teilweise zu Verwirrung mit dem Ergebnis, dass die Testumgebung nicht startete. Auch dieser Fehler konnte nach der Fehlermeldung von der Entwicklungsumgebung selbstständig behoben werden. Dieses Problem ist dem Ver-

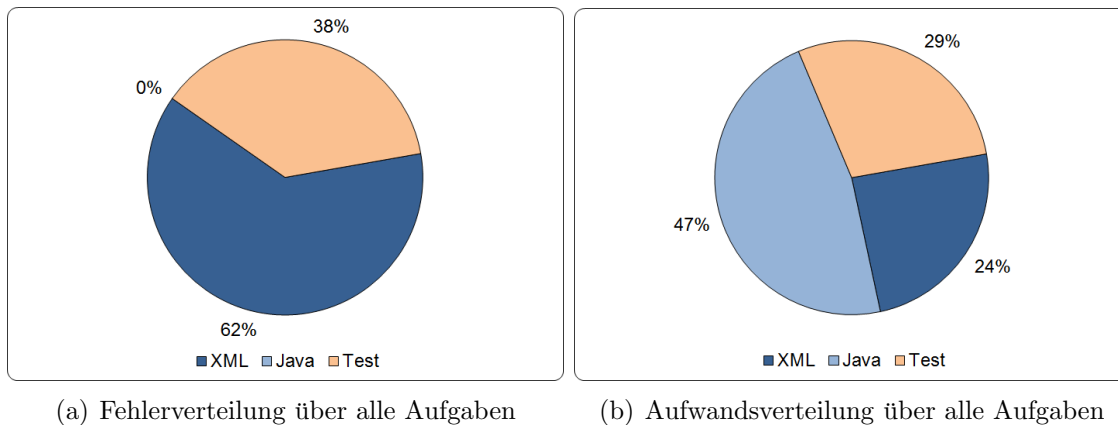


Abbildung 6.12: Auswertung der Aufgabenbearbeitung

suchsaufbau des Workshops geschuldet und würde in einer echten Installation nicht auftreten.

Der zweite Teil der Probleme bezieht sich auf die Anpassungen der AAL-Servicekonfiguration in Java und XML. Hierbei führten Probleme häufig zu einem Fehlschlag der gesamten Teilaufgabe. Dieser Zusammenhang ist gut in Abbildung 6.9 zu erkennen.

Validators Tag vergessen: Validatoren werden zu einem `ConfigurationsItem` hinzugefügt, um es bei seiner Instanziierung zu validieren. Sollen mehrere Validationen ausgeführt werden, müssen die dazugehörigen Validatoren in der XML mit dem `Validators` Tag zusammengefasst werden.

Validator nicht innerhalb des `ConfigurationItem`: Da ein Validator bei der Konfigurationsinstanziierung zur Prüfung der Eingabe eingesetzt wird, muss er zwingend einem `ConfigurationItem` zugeordnet werden.

Weitere Fehler, die in Abbildung 6.11 nicht dargestellt sind, beziehen sich auf Implementierungsfehler während der Quellcodebearbeitung in der Entwicklungsumgebung und hängen nicht mit Konfigurationsaspekten zusammen. In diese Kategorie fallen beispielsweise Tippfehler oder fehlende Importe. Diese Fehler konnten, durch die Mechanismen zur Fehlerkorrektur der Entwicklungsumgebung, von den Teilnehmern schnell und selbstständig behoben werden.

Im Zusammenhang mit den Fehlern, die im Laufe des Praxisteils des Workshops entstanden sind, lässt sich ein weiterer interessanter Zusammenhang auswerten. Betrachtet wurde die Fehlerverteilung in Abhängigkeit zum Aufwand der jeweiligen Aufgabe. Dies ergibt einen wichtigen Hinweis auf Verbesserungspotenzial der dazugehörigen Komponente aus dem Lifecycle oder Framework. In Abbildung 6.12a sieht man die Verteilung dieser schwerwiegenden Fehler, die zu einem Fehlschlag der Aufgabe führen. Blau ist der XML-Anteil der Entwickler und orange der Anteil der Deployer. In Abbildung 6.12b dagegen ist die Zeit angetragen, die im Durchschnitt zur Bearbeitung der einzelnen Aufgabenteile benötigt wurde. Man sieht, dass die

Teilnehmer die Hälfte der Zeit (47%) mit Java zugebracht haben. 29% der Zeit entfällt auf Testen und mit 24% der kleinste Teil auf die Bearbeitung der XML. Hierbei ist zu erkennen, dass der arbeitsintensive Teil Java keine Abbruchfehler erzeugte. Dagegen wurde im kleinsten Teil (XML 24%) der größte Abbruchanteil erzeugt mit 62%.

Zusammenfassung Effektivität

Die Effektivität ist insgesamt als gut zu bezeichnen. Die verschiedenen Aufgaben konnten nach der Schulungseinheit bearbeitet und gelöst werden. Auch die Leistungsfähigkeit der Konzepte in Bezug auf die eigenen Erfahrungen mit AAL-Services wurde eher hoch bewertet. Probleme traten nur im XML-Teil der Konfiguration auf, was kombiniert mit der Aufgabenverteilung zeigt, dass 3/4 der Bearbeitung problemlos erfolgte. Mit steigender Komplexität der Aufgaben nahm die Anzahl der Bearbeitungsversuche aber ab.

Zur Verbesserung der Effektivität sollte die Bearbeitung der XML-Datei benutzerfreundlicher und robuster gestaltet werden. Hierzu kam in der Fokusgruppe 07 die Idee auf, ein Werkzeug in Form eines Plug-Ins für die Entwicklungsumgebung zu erstellen, das die Konsistenz der XML-Daten prüft und bei ihrer Erstellung unterstützt. Der Wunsch nach einer komfortableren XML-Bearbeitung wurde auch in der Diskussion mit den Teilnehmern identifiziert.

6.2.1.2 Effizienz

Zur Bewertung der Effizienz wurde der gemessene und wahrgenommene Aufwand näher analysiert (vgl. Kapitel 2.3.2 Frage E2.2.1). Abbildung 6.13 stellt den Aufwand basierend auf den Ausführungszeiten aus der Videoanalyse dar. Hierbei wird in die beiden verschiedenen Aufgaben (Wettervorhersage und Textfarbe, vgl. Kapitel 5) unterteilt. Weiterhin teilt sich jede Aufgabe in einen Teil der Entwicklung, in der der Teilnehmer die Rolle des Entwicklers einnimmt und einen darauf aufbauenden Teil des Integrierens in die Middleware, in der der Teilnehmer die Rolle des Deployers einnimmt. Bezüglich der gemessenen Ausführungszeiten für Entwicklung und Integration in die Plattform ergibt sich ein doppelt so langer (2,08facher) zeitlicher Aufwand im Vergleich von Aufgabe 1 zu 2. Vergleicht man dagegen die reine Entwicklungszeit, vervierfacht (4,23facht) sich der Aufwand für die gleichen Aufgaben. Bei genauerer Betrachtung wird ersichtlich, dass über die Hälfte der Zeit (57,8%) in Aufgabe 1 für das Testen verwendet wird und nur 14,5% in Aufgabe 2. Dies erlaubt folgende Schlussfolgerung: Das Testen der Implementierungen als Deployer dauert während der Einlernphase verhältnismäßig lange. Danach finden sich die Deployer bei der zweiten Verwendung des Frameworks wesentlich besser zurecht.

Nachdem der Sachverhalt sich für den Deployer recht deutlich abzeichnet, steht die Frage im Raum, wie sich der Aufwand beim Entwickler verhält. Dies ist nicht so einfach abzulesen, da die Komplexität der Entwicklungsaufgabe zwischen Aufgabe 1 und 2 ansteigt. Die Baseline für eine Vergleichsgrundlage stellt hierbei das Keystroke Level Model (KLM, siehe Kapitel 2.3.2). Mit dem KLM können die Ausführungszeiten von erfahrenen Benutzern für eine Aufgabe abgeschätzt werden. Da die Workshopteilnehmer das System zum ersten Mal verwenden, lassen sich die KLM-Abschätzungen nicht direkt als Vergleichswert heranziehen. Deshalb wird nicht der

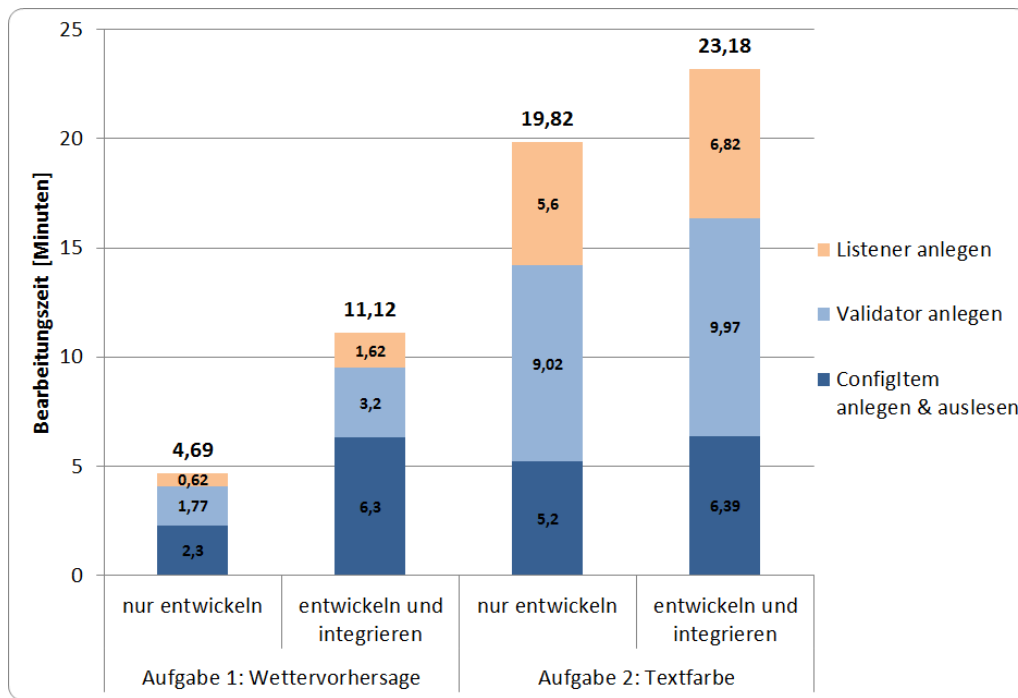


Abbildung 6.13: Durchschnittlicher Aufwand zum Entwickeln und Integrieren (n=9)

	Theoretische Ausführungszeiten			Gemessene Ausführungszeiten			Differenz
	Aufg. 1	Aufg. 2	Faktor	Aufg. 1	Aufg. 2	Faktor	
Anlegen							
ConfigItem	0,57	1,56	2,75	2,3	5,2	2,26	0,49
Validator	0,64	2,29	3,59	1,77	9,02	5,1	-1,51
Listener	0,16	1,94	12,11	0,62	5,6	9,03	3,08

Tabelle 6.1: Darstellung des Mehraufwands zwischen Ausgabe 1 und 2

absolute zeitliche Aufwand zwischen den Teilnehmern und dem KLM verglichen, sondern die Aufwandsdifferenz. Hierzu werden die theoretische und tatsächliche Ausführungszeit zwischen der Aufgabe 1 und 2 in Relation zueinander gesetzt. Die Aufgaben wurden extra so konzipiert, dass sich bei ähnlichem Inhalt nur die Komplexität ändert. Jeder der drei Hauptkonfigurationsaspekte kann einzeln auf eine Verbesserung oder Verschlechterung bezüglich seines zeitlichen Aufwands gegenüber dem theoretischen Aufwand des KLMs bewertet werden.

Die Berechnung des theoretischen Aufwands folgt den in Kapitel 2.3.2 beschriebenen Schritten des KLMs. Es ergeben sich somit für die erste Aufgabe 99 und für die zweite Aufgabe 217 Elementaroperationen. Abbildung 6.14 stellt die kumulierten Elementaroperationen dar. Aus diesen Daten kann ein Faktor für den Mehraufwand zwischen Aufgabe 1 und 2 gewonnen werden. Dieser wurde sowohl für den theoretischen als auch für den tatsächlichen Mehraufwand berechnet. Zusammenfassend sind die resultierenden Faktoren in Tabelle 6.1 dargestellt.

Die Tabelle ist wie folgt zu lesen: Der theoretische Aufwand für das Anlegen eines ConfigItem steigt zwischen Aufgabe 1 und 2 um Faktor 2,75. Aufbauend auf diesen Faktoren für den theoretischen und tatsächlichen Aufwand kann auf eine Verbesse-

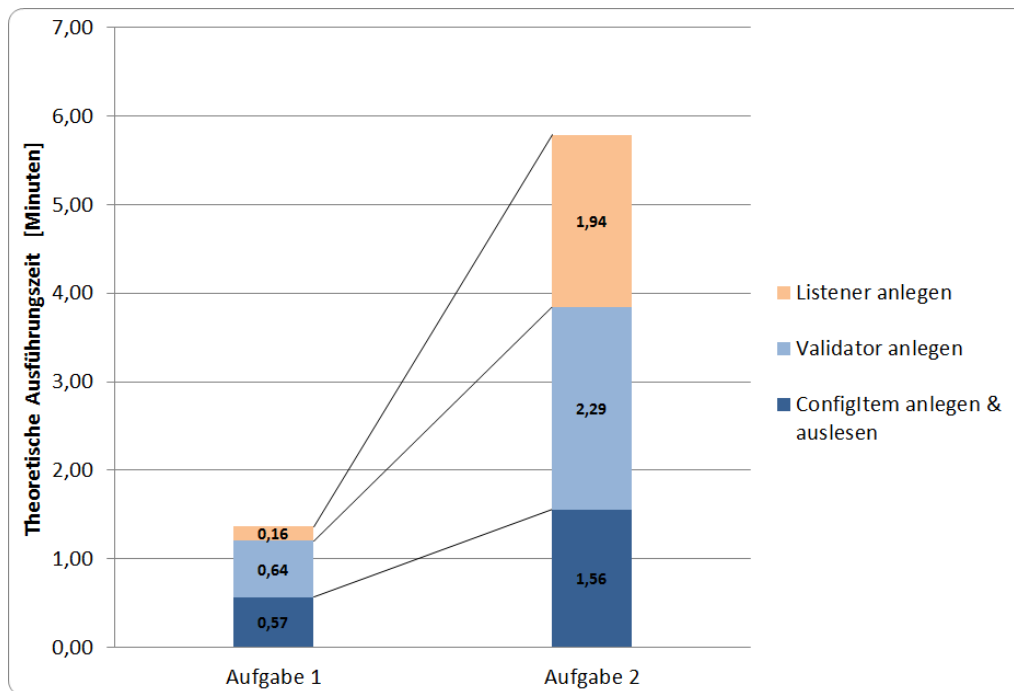


Abbildung 6.14: Ausführungszeiten für Aufgabe 1 und 2 ohne Testen

rung oder Verschlechterung des Lösungsverhaltens des Teilnehmers und des damit verbundenen Aufwandes bzw. Erlernen der Nutzung für ihn geschlossen werden. So zeigt sich beim Anlegen des `ConfigItem` eine leichte Verbesserung von 0,49 (1,12 min) und beim Anlegen eines `Listeners` sogar eine Verbesserung von 3,08 (1,91 min) (vgl. Abbildung 6.15). Beim Anlegen der `Validator` zeigt sich dagegen eine Verschlechterung um 1,51 (2,66 min), die auf die im vorherigen Kapitel beschriebenen Probleme zurückzuführen ist.

Der zweite Teil der einführenden Frage zielte auf den wahrgenommenen Aufwand ab. Dieser Teil wurde über den Fragebogen erhoben (vgl. Abbildung 6.16). Die Teilnehmer beurteilten hierbei die Umsetzung der Konfigurationskonzepte des Livecycles als eher einfach. Bei den tendenziell negativeren Bewertungen dieser Frage befinden sich im Kommentarfeld des Fragebogens Anmerkungen zu fehlenden Werkzeugen zur komfortablen Anpassung der Konfigurationsdefinition.

Für die Effizienz spielt neben dem Aufwand auch die Komplexität eine wichtige Rolle. Sie wurde im Fragebogen mit Frage E2.2.2 (siehe Kapitel 2.3.2) abgefragt und bewertet. Die Vielfalt des vorgestellten Konfigurationslifecycles wird als hoch bewertet (vgl. Abbildung 6.17). Ergänzend wurde in der Diskussion angemerkt, dass die Vielfalt bezogen auf die Konfigurationsmerkmale bzw. deren Komplexität positiv wahrgenommen wird.

Zusammenfassung Effizienz

In Bezug auf die Effizienz ließen sich in der Analyse der Aufwände und der Komplexität wichtige Ergebnisse und Verbesserungspotenziale identifizieren. Die Auswertung zeigt deutlich die Fehleranfälligkeit der XML-Konfiguration für den Entwickler. Sie könnte durch geeignete Werkzeugunterstützung leicht behoben werden. Dennoch erlernten die Entwickler die Anwendung des Konfigurationsframeworks schnell und

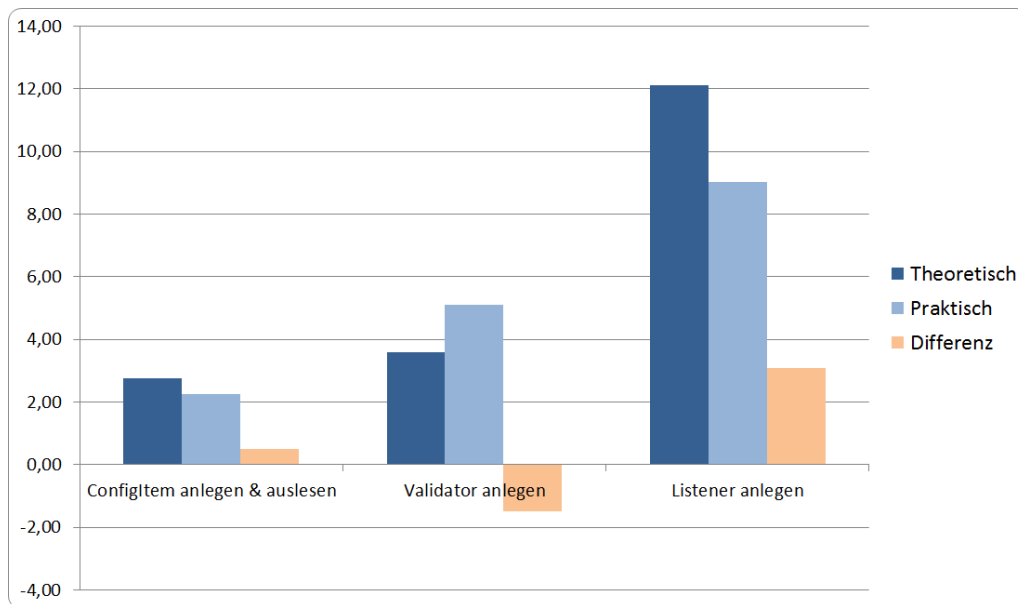


Abbildung 6.15: Gemessener Speed-Up

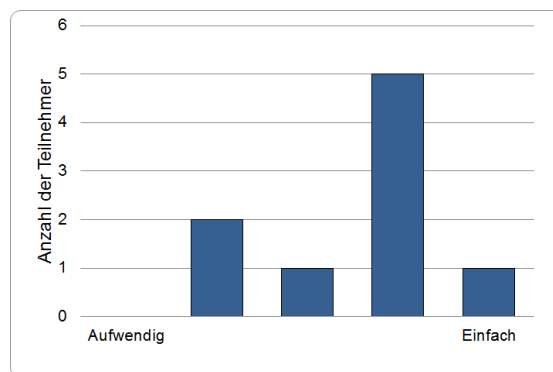


Abbildung 6.16: Q3 Umsetzung: Wie finden Sie die technische Umsetzung der Konfigurationskonzepte (z.B. ConfigItem, Validator und Listener)?

setzten die gestellten Aufgaben entlang des Lifecycles um. Dies ist deutlich durch die entsprechenden Ausführungszeiten belegt, die trotz steigender Komplexität der Aufgaben abnehmen. Die Werkzeuge für den Deployer werden nach einer kurzen Lernphase gut angenommen und die Daten zeigen auch hierbei einen deutlichen Effizienzvorteil.

Weiterhin spricht für eine gelungene Effizienz des Konfigurationsansatzes, dass der wahrgenommene Aufwand bei der Umsetzung der Konfigurationskonzepte als gering beziffert ist, dagegen aber die Vielfalt der Konzepte als eher hoch angesehen wird. Insgesamt werden die Konzepte als stimmig und ausgewogen betrachtet, auch wenn es Verbesserungsbedarf bei den einzelnen Werkzeugen gibt.

6.2.1.3 Zufriedenheit

Die Zufriedenheit mit der vorgestellten und getesteten Lösung wurde durch den Fragebogen (vgl. Kapitel 2.3.2 Fragen E2.3.1 & E2.3.2) und in der Diskussion abgefragt. Im Fragebogen wurde die Zufriedenheit aus der Verständlichkeit und dem

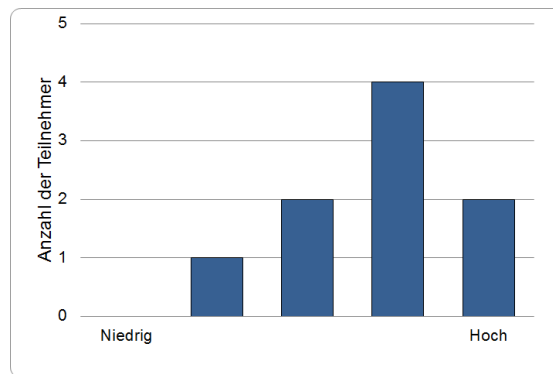
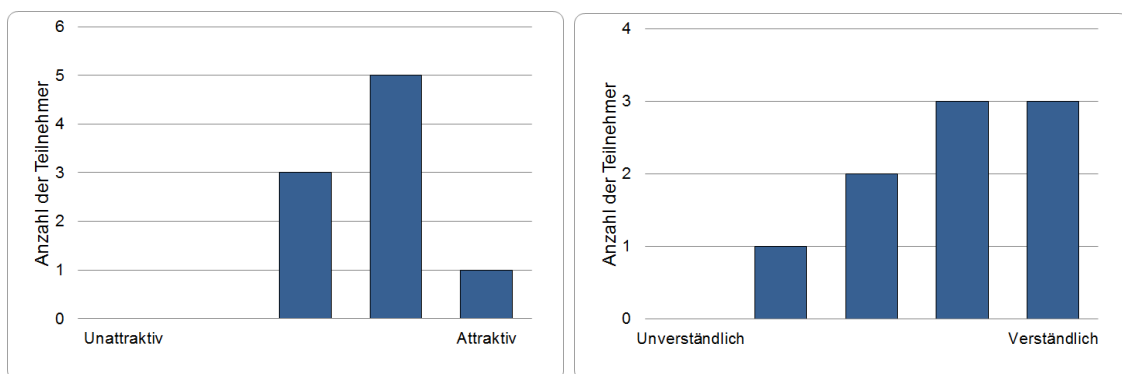


Abbildung 6.17: Q7 Komplexität: Wie schätzen Sie die Vielfalt der vorgestellten Konzepte ein?



(a) Q8 Gesamteindruck: Wie finden Sie die vorgestellten Konzepte und Umsetzungen insgesamt? (b) Q6 Nachvollziehbarkeit: Wie verständlich sind die vorgestellten Konzepte?

Abbildung 6.18: Auswertung der Zufriedenheit der Teilnehmer mit dem Konfigurationsframework

Gesamteindruck abgeleitet. Der Gesamteindruck (siehe Abbildung 6.18a) der Lösung ist für die Teilnehmer eher attraktiv. Die Nachvollziehbarkeit (siehe Abbildung 6.18b) der Lösung stellt sich als verständlich dar. Somit zeigen beide Aspekte ein positives Bild der Zufriedenheit.

Dieses ließ sich in der anschließenden Diskussion weiter hinterfragen und ergab nach Aussage eines Teilnehmers „eine gute Basis zum Entwickeln und Integrieren von eigenen Anwendungen“. Die Integration in vorhandene Entwicklungsumgebungen sowie die Konzepte des Lifecycles sind „für das Programmieren einfach und verständlich“. Weiterhin äußerte ein Teilnehmer, die Werkzeuge hätten eine ansprechende GUI und zeigten eine gute Benutzbarkeit. Neben diesen positiven Aspekten waren die Teilnehmer mit der Nutzung des Lifecycles und des Frameworks für die Gruppe der Endanwender eher unzufrieden. Aber es war nie das Ziel, mit dem Framework die Endanwender zu unterstützen. Im Lifecycle partizipieren sie absichtlich nicht aktiv (vgl. Kapitel 4.2). Auch wenn die Endanwender nicht das Ziel dieser Werkzeuge sind, kann diese Anregung auch für den Deployer eine Verbesserung bedeuten. So wurde beispielsweise vorgeschlagen, den Konfigurator eher als Wizard aufzubauen oder um diese Funktion zu ergänzen, denn es ist „aus Sicht des Anwenders kryptisch und die

Komplexität hoch“. Hier sollte auch die bereits umgesetzte Möglichkeit, vorkonfigurierte Standardparameter zu laden, wieder aufgegriffen werden, „um den manuellen Aufwand zu minimieren“.

Zusammenfassend kann man sagen, dass die vorgestellte Lösung zufriedenstellend ist, aber nach Wunsch einzelner Teilnehmer auf Endanwender ausgedehnt werden sollte.

6.2.1.4 Vollständigkeit

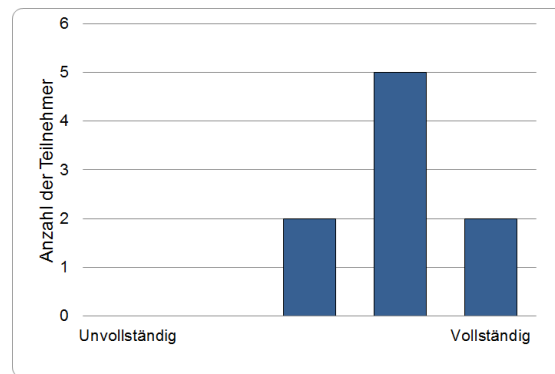


Abbildung 6.19: Funktionalität Q4: Wie schätzen Sie den Funktionsumfang der vorgestellten Konzepte ein?

Die Vollständigkeit der Konfigurationskonzepte wurde im Fragebogen (vgl. Kapitel 2.3.2 Frage E2.4.1) und durch die Diskussion evaluiert (vgl. Abbildung 6.19). Im Kommentarfeld des Fragebogens wird der mangelnde Bezug zu Sensoren und Aktoren angemerkt. In der Diskussion konnte geklärt werden, dass dieser Bezug in der Schulung ausreichend dargestellt worden ist, leider aber in der Hands-on-Phase nicht umgesetzt wurde. Dies hatte hauptsächlich organisatorische Gründe. Es wäre zum einen zu aufwendig gewesen, neben den vorbereiteten Arbeitsplätzen noch Sensoren und Aktoren zur Verfügung zu stellen. Zum anderen wurde sich aus zeitlichen Gründen bewusst dagegen entschieden und die Sensoren über das Internet in den Inframe vorab eingebunden.

In der Diskussion wurde darüber gesprochen, wieviel zur Konvention in die Konfiguration einfließen darf und wie sich das auf die Vollständigkeit der Konzepte auswirkt. Insgesamt herrschte der Konsens, dass viele Regeln die Vollständigkeit der Konfiguration einschränken würden. Aber auch die entgegengesetzte Meinung wurde von einem Teilnehmer zusammengefasst: „convention over configuration“. Dies sei für ihn immer das Mittel der Wahl.

Ein weiterer angesprochener Punkt war die teilweise prototypische Benennung einzelner Parameter in den Werkzeugen des Konfigurationsframeworks. Aus der Gruppendiskussion ergab sich, dass mit der vorgestellten Schulung die einzelnen Frameworkwerkzeuge gut zu nutzen waren, aber eine ausführlichere Beschreibung und Dokumentation der einzelnen Parameter für eine „intuitivere Nutzbarkeit“ notwendig sind.

Insgesamt ist somit eine gute Vollständigkeit des Lifecycles und des Frameworks erreicht, die durch einzelne Optimierungen weitergetrieben werden kann. Prinzipielle Vorbehalte gegen eine Konfiguration versus Konvention konnten natürlich nicht

ausgeräumt werden, aber dennoch wurde auch bei diesem konfigurationskritischen Teilnehmer die vorgestellte Lösung positiv wahrgenommen.

6.2.2 Diskussion der Ergebnisse

Ziel war es, den finalen Prototyp des Konfigurationslifecycle zu evaluieren. Die Bewertung erfolgte gestützt von der Quality-in-Use. Nur so war es möglich, eine umfassende Analyse aller wichtigen Qualitätsmerkmale vorzunehmen und sicherzustellen, dass alle AAL-relevanten Aspekte abgedeckt werden. Hierzu wurde eine Kombination aus Schulung, Praxis und Diskussion gewählt. Das Ergebnis zur Einleitenden Frage E2.0 teilt sich in die vier Qualitätsmerkmale der Quality in Use auf.

Die Effektivität des Systems wurde insgesamt als gut betrachtet. 80,5% der bearbeiteten Aufgaben wurden durch die Teilnehmer erfolgreich gelöst. Auch die Leistungsfähigkeit der Konzepte wird als angemessen betrachtet und aus den aufgetretenen Problemen und Fehlern lässt sich eine eindeutige Maßnahme zur Lösung derselben ableiten: Zur Verbesserung der Effektivität sollte die Bearbeitung der XML-Datei benutzerfreundlicher und robuster gestaltet werden.

Die Effizienz wurde über den gemessenen und wahrgenommenen Aufwand sowie die Komplexität für den Entwickler und Deployer evaluiert. Insgesamt fanden sich sowohl Entwickler als auch Deployer schnell im Lifecycle zurecht und konnten das Framework gut benutzen. Dies spiegelt sich im tatsächlichen und wahrgenommenen Aufwand sowie in der Lernkurve der Teilnehmer wider. Bei der Untersuchung der Effizienz zeigte sich, dass geeignete Werkzeuge (z.B. ein XML-Editor zum Bearbeiten der Konfigurationselemente) dieselbe weiter steigern können.

Die Zufriedenheit mit den Konzepten und deren Umsetzung im Lifecycle war bei den Teilnehmern vorhanden und wurde in der abschließenden Diskussion bestätigt.

Die Vollständigkeit des Konfigurationslifecycle konnte positiv evaluiert werden. Die Teilnehmer waren sich einig, dass sie ihren AAL-Service damit umsetzen könnten und auch andere denkbare AAL-Services ausreichend umsetzbar sind. Nur durch einen Teilnehmer wurde darauf aufmerksam gemacht, Softwaredesign-Paradigma „convention over configuration“ (Konvention vor Konfiguration) nicht zu vergessen.

Insgesamt zeigte diese Evaluation, dass der Konfigurationslifecycle allen Ansprüchen genügt und als Referenz zur flexiblen Einbindung von AAL-Services dient. In der Implementierung des Frameworks können noch einzelne Aspekte verbessert werden und so eine höhere Benutzbarkeit erreicht werden.

6.3 Zusammenfassung

Die Ergebnisse der beiden Evaluationen zeigen, dass flexibles Management von AAL-Umgebungen möglich ist und welche Vor- und Nachteile damit verbunden sind. Der erste Teil der Evaluation beantwortet, auf Grundlage der Analysen aus Kapitel 3, die erste Forschungsfrage. Weiterhin stellt er die Basis für das entwickelte Konfigurationsframework aus Kapitel 4 und deren Implementierung in Kapitel 5 für die Evaluation der zweiten Forschungsfrage.

Forschungsfrage 1: *Was sind die Anforderungen an ein flexibles Management von AAL-Umgebungen und die damit verbundene Konfiguration?*

Die Beantwortung dieser Frage zerfällt in drei Teile, die jeweils in den Fokusgruppen validiert und anschließend zusammen evaluiert wurden. Es müssen die Anforderungen der Akteure, der technischen Infrastruktur und der darauf laufenden AAL-Anwendungsfälle untersucht werden. Als erstes wurden die 11 verschiedenen Akteure der AAL-Umgebungen, die am flexiblen Management partizipieren identifiziert. Sie haben jeder verschiedene Bedürfnisse, die sich zu 63 Anforderungen in fünf Kategorien zusammenfassen lassen. Unter ihnen sind 10 Pflichtanforderungen, ohne die ein flexibles Management nicht möglich ist. Im zweiten Teil wurden Plattformen genauer analysiert. Hierbei ergeben sich weitere fünf technische Anforderungen. Abschließend wurden AAL-Anwendungsfälle auf ihre Anforderungen bezüglich der Flexibilität untersucht. Die Flexibilität wurde hierbei in Komplexitätsgruppen der Konfigurierbarkeit zusammengefasst.

Insgesamt lassen sich die Anforderungen auf vier Forderungen aggregieren, um flexibles Management zu garantieren: (1) Es muss einfach möglich sein neue Funktionalität in die AAL-Umgebung zu integrieren. (2) Das Wissen über die AAL-Umgebung muss in strukturierter Form gesammelt und wieder bereitgestellt werden. (3) Die AAL-Umgebung muss sich warten lassen, um auf unvorhersehbare Veränderungen geeignet zu reagieren. (4) Die AAL-Umgebung muss sich auf die speziellen Bedürfnisse des Bewohner personalisieren lassen.

Nachdem alle drei Teile von den jeweiligen Fokusgruppen validiert waren, begann die Evaluation der Anforderungen. Hierzu wurde der erste Prototyp des Konfigurationsframeworks genutzt. Mit der Living Lab-Methode konnten die verschiedenen Anforderungen an ein flexibles Management direkt am Konfigurationslifecycle vorgestellt werden und von den Teilnehmern der Evaluation bewertet werden. Zusätzlich wurde aus den Antworten abgeleitet, welche Qualitätsmerkmale und damit verbundene Anforderungen als wichtig betrachtet werden.

Die Gewichtung der Anforderungen zeige die folgende Reihenfolge: Die Benutzbarkeit- (Usability) und Zuverlässigkeitsanforderungen (Reliability) sind für den Deployer am Wichtigsten. Auch wichtig aber geringer bewertet sind die Anforderungen mit Bezug zu Effizienz (Performance Efficiency) und Funktionalität (Funktional Suitability). Danach folgen Anforderungen der Kategorien Kompatibilität (Compatibility), Sicherheit (Security), Wartbarkeit (Maintainability) und Übertragbarkeit (Portability).

Im zweiten Teil der Evaluation wurden die einzelnen Vor- und Nachteile des semantischen Konfigurationsframeworks untersucht. Somit konnte auch die zweite Forschungsfrage beantwortet werden.

Forschungsfrage 2: *Welche Vor- und Nachteile bietet eine ontologiebasierte Konfiguration von AAL-Anwendungsfällen für ein flexibles AAL-Umgebungsmanagement?*

Die Vor- und Nachteile konnten sowohl in den die Arbeit begleitenden Fokusgruppen sowie in der finalen Frameworkevaluation erfasst werden. Um sicher zu stellen, dass alle relevanten Bereiche evaluiert werden, wurde das Framework entlang von vier Qualitätsmerkmalen untersucht. Die Effektivität, Effizienz, Zufriedenheit und Vollständigkeit der Konfiguration konnten positiv bewertet werden.

Die Effektivität werden als gut bewertet. Nach der Schulung sind alle Teilnehmer in der Lage das Konfigurationsframework zu benutzen und an sie gestellte Auf-

gaben effektiv zu lösen. Die Leistungsfähigkeit der Konfigurationskonzepte ist von den Teilnehmern als hoch bewertet. Eigene AAL-Anwendungsfälle aus ihrem eigenen Alltag können sie mit dem Framework nutzen. Nur teilweise werden Aufgaben bei steigender Komplexität nicht mehr richtig gelöst. Dies könnten durch geeignete Werkzeugunterstützung behoben werden. Die Effektivität dieses Ansatzes ist als Vorteil zusehen, der durch geeignete Werkzeugunterstützung noch gesteigert werden kann.

Die Effizienz des Frameworks wurde durch eine Messung der Aufwände und der Komplexität bei verschiedenen Konfigurationsaufgaben bewertet. Insgesamt lernen die Teilnehmer schnell und erledigen die gestellten Aufgaben effizient, was durch sinkende Ausführzeiten bei steigender Komplexität zu sehen ist. Nur der Modellierungsschritt (XML Dateien und Ontologien) bereitet teilweise Probleme. Dieser Nachteil ist durch geeignete Modellierungswerkzeuge zu beheben. Insgesamt bewerten die Teilnehmer den wahrgenommenen Aufwand als gering und gleichzeitig die Mächtigkeit des Konfigurationsframeworks als hoch. Trotz der teilweisen Modellierungsprobleme wird insgesamt die technische Umsetzung der Konfigurationskonzepte im Framework als einfach nutzbar angesehen. Die Effizienz des Konfigurationsframeworks wird somit als Vorteil gesehen, der durch weitere Schulungsmaterialien zur Modellierung der Konfigurationsinformationen und Werkzeuge weiter gesteigert werden kann.

Die einzelnen Konzepte des Frameworks sind verständlich nachzuvollziehen und der Funktionsumfang wird als vollständig angesehen. Die Vollständigkeit des Konfigurationsframeworks ist somit als Vorteil zu sehen.

Insgesamt überwiegen die Vorteile einer ontologiebasierten Konfiguration für das flexible Management nach Meinung der Evaluationsteilnehmer, die das Gesamtkonzept attraktiv einschätzen. Die einzelnen Nachteile in der Werkzeugunterstützung betritt nicht die konzeptionelle Ebene und zeigt, dass das Qualitätskriterium Benutzbarkeit aus der ersten Evaluation noch verbessert werden kann.

7. Zusammenfassung und Ausblick

In dem relativ jungen Forschungsfeld AAL stellen heterogene Komponenten und sich schnell ändernde Kontexte eine große Herausforderung dar. Mit Hilfe eines flexiblen Managements soll diesen begegnet werden. Hierzu bedarf es einer Analyse der Anforderungen (Forschungsfrage 1) und ein Framework zur Gestaltung der Flexibilität (Forschungsfrage 2).

Mit dieser Arbeit liegt die Entwicklung und Evaluation eines Konfigurationsframeworks und seiner Referenzimplementierung zum flexiblen Management von AAL-Umgebungen vor. Das Konfigurationsframework basiert auf einer umfassenden Analyse der Akteure und Anforderungen an ein flexibles Management und endet in einem Konfigurationsframework. Dieses Framework besteht aus mehreren Methoden und Prozessen für die Konfiguration von AAL-Umgebungen. Ein für AAL-Umgebungen entwickelter Lifecycle strukturiert den Einsatz dieser Methoden und Prozesse. Die in dieser Arbeit umgesetzte Referenzimplementierung diente der Evaluation. Die Evaluation dieser Arbeit bewertete die Anforderungen an das flexibles Management sowie das entwickelte Konfigurationsframework.

7.1 Zusammenfassung

Die Kommunikation von Alltagsgegenständen miteinander sowie im „Internet der Dinge“ nimmt beständig zu und macht aus unserem Umfeld zunehmend eine intelligente Umgebung. Ambient Assisted Living (AAL) beschreibt eine Anwendungsdomäne von intelligenten Umgebungen. Sie ist darauf spezialisiert hilfsbedürftigen, oft älteren Menschen, sowie ihren formellen und informellen Betreuern durch AAL-Anwendungsfälle im Alltag Unterstützung zu bieten. Durch den Demografischen Wandel, den Mangel an medizinischem Personal und die steigenden Gesundheitskosten werden AAL-Umgebungen zukünftig für unsere Gesellschaft immer wichtiger.

Die Herausforderung hierbei ist es, die vielen heterogenen Komponenten (Hardware, Software und menschliche Dienstleistungen) der AAL-Anwendungsfälle in der AAL-Umgebung geeignet zu arrangieren, um auf die individuellen, schnell wechselnden Bedürfnisse der Nutzer einzugehen. Viele Hersteller von AAL-Lösungen bieten hierfür proprietäre Produkte oder Insellösungen. Diese mangelnde Interoperabilität

soll mit dem Potenzial eines wissensbasierten Konfigurationsprozesses überwunden werden und eine hohe Flexibilität beim Aufbau von AAL-Lösungen sicherstellen.

Aus dieser Motivation heraus adressiert diese Arbeit die folgenden zentralen Forschungsfragen:

Forschungsfrage 1: *Was sind die Anforderungen an ein flexibles Management von AAL-Umgebungen und die damit verbundene Konfiguration?*

Forschungsfrage 2: *Welche Vor- und Nachteile bietet eine ontologiebasierte Konfiguration von AAL-Anwendungsfällen für ein flexibles AAL-Umgebungsmanagement?*

Für die Beantwortung dieser beiden Fragen orientiert sich diese Arbeit an der Methodik des Design Science Research-Methoden und wendet für die Analyse, Design und Evaluationsphasen die Methodik der Fokusgruppen und der Living-Labs an.

Zur Forschungsfrage 1 wurden AAL-Umgebungen mit ihren Lösungen systematisch analysiert. Durch die Analyse konnte gezeigt werden, dass der Einflusskreis des flexiblen Managements weit größer ist als für bisherige AAL-Umgebungen mit einfachen Installationen angenommen. Zuerst sind die Akteure und ihre Anforderungen erfasst worden. Das Ergebnis sind fünf Akteurgruppen, die an der Konfiguration unmittelbar beteiligt sind. Im Rahmen von Fokusgruppen wurden mit repräsentativen Vertretern der Akteure Anforderungen erhoben und priorisiert. Insgesamt konnten 63 Anforderungen für flexibles Management von AAL-Umgebungen identifiziert werden.

Zudem wurden 43 existierende AAL-Plattformen auf ihre AAL-Tauglichkeit gegen die Pflichtanforderungen analysiert und ihre Flexibilität ausgewertet. Zehn der untersuchten AAL-Plattformen konnten die Pflichtanforderungen erfüllen und die identifizierten AAL-Referenzanwendungsfälle umsetzen. Dennoch fehlt allen Plattformen die geforderte Interoperabilität. Teilweise war die Auswahl der Hardware und der Anwendungsfälle eingeschränkt oder es fehlte die Möglichkeit, sie an den älteren Menschen anzupassen. Die Analyse ergibt somit, dass Ansätze bereits existieren, die allerdings keine flexible Konfiguration von Hardware, Software und den beteiligten Menschen bietet. Es fehlen Werkzeuge und Prozesse, um die jeweiligen Komponenten miteinander zu integrieren.

Die Analyse von 55 AAL-Anwendungsfällen aus elf Projekten zeigt Kerneigenschaften, die sich in zehn repräsentativen Kategorien für AAL-Lösungen zusammenfassen lassen. Diese AAL-Anwendungsfälle lassen sich orthogonal den vier Stufen der Konfigurationskomplexität zuordnen.

Nach Abschluss der Analysen konnte das Konfigurationsframework entwickelt und implementiert werden. Es unterstützt die Integration von AAL-Anwendungsfällen sowie deren Komponenten in die AAL-Umgebung. Hierzu ist ein Lifecycle entwickelt worden, der die AAL-Anwendungsfälle über ihre verschiedenen Phasen begleitet:

- Die **Entwicklung** definiert, welche Parameter zu konfigurieren sind, und implementiert sie im AAL-Anwendungsfall zur späteren Verwendung im Lifecycle.

- Die **Design**-Phase unterstützt bei der Auswahl der richtigen AAL-Anwendungsfälle auf Basis ihrer konfigurierbaren Parameter mit dem Ziel der bestmöglichen AAL-Lösung.
- Die **Installation** integriert die AAL-Anwendungsfälle, wie in der Design-Phase festgelegt, in der AAL-Umgebung.
- Die **Konfiguration** nimmt die Anpassungen der AAL-Anwendungsfälle entsprechend der AAL-Umgebung und der Anforderungen der Nutzer vor.
- Die **Wartung** reagiert auf Unregelmäßigkeiten in der AAL-Umgebung durch Anpassung der Konfiguration oder Austausch von ganzen Komponenten, um die Funktion der AAL-Anwendungsfälle aufrecht zu erhalten.

Der Kern des Konfigurationsframeworks ist eine wissensbasierte Repräsentation aller konfigurationsrelevanten Informationen. Dies ermöglicht eine flexible Unterstützung durch ein gemeinsames Verständnis und die Wiederverwendbarkeit der einzelnen Komponenten der AAL-Anwendungsfälle. So gelingt es, die Interoperabilität in AAL-Umgebungen zu steigern und flexibel auf die Bedürfnisse der Nutzer einzugehen.

Gemäß der Methodik des Design Science Research wurde das Framework und seine Konzepte entlang des Product Quality Models und Quality in Use Models evaluiert. In zwei Evaluationen konnten die Anforderungen sowie das Konfigurationsframework auf ihre Vor- und Nachteile hin geprüft werden. Die Evaluation der Anforderungen der Akteure, der Plattformen und der Anwendungsfälle liefern als Ergebnis eine Definition zum notwendigen Funktionsumfang der Konfiguration und eine Bewertung der Wichtigkeit der einzelnen Anforderungen. Diese Ergebnisse stellen die Basis für die Konzeption und Implementierung des Konfigurationsframeworks.

Zur Beantwortung der zweiten Forschungsfrage wurde das fertige Framework in einem Workshop mit 10 Domänenexperten evaluiert. Das Framework konnte in der Praxis bestehen und wird von den Teilnehmern als sinnvolle und nützliche Ergänzung zur Steigerung der Flexibilität in AAL-Umgebungen angesehen.

Es konnte gezeigt werden, dass insgesamt die Vorteile dieses Ansatzes überwiegen. Sind die entsprechenden Komponenten in der AAL-Umgebung semantisch modelliert, lassen sie sich flexibel miteinander arrangieren und bedürfnisgerecht konfigurieren. Gerade für AAL-Anwendungsfälle bedeutet die semantische Beschreibung ihrer Schnittstellen, der Hardware und der Akteure in der AAL-Umgebung einen Vorteil durch das flexible Arrangieren der Komponenten untereinander. Die einzelnen Komponenten sind in mehreren AAL-Anwendungsfällen nutzbar und durch die Konfiguration aufeinander abgestimmt.

Es konnte gezeigt werden, dass auf Basis der semantischen Beschreibung auch ein Automatisierungspotenzial besteht. Liegen die richtigen Informationen über die Komponenten der Umgebung bereit, können viele Konfigurationsschritte automatisch erfolgen. Hierzu dienen die zu den einzelnen Komponenten gehörenden Konfigurationsinformationen. In der Evaluation zeigt sich, dass die Aufgaben schneller und durch die Automatisierung auch fehlerfreier gelöst wurden.

Einer der großen Vorteile in einem medizinisch sensiblen System, wie einer AAL-Umgebung, ist die Möglichkeit durch die semantische Beschreibung flexibel auf

Ausnahmesituationen zu reagieren und entsprechend zu deeskalieren. Die semantische Beschreibung der Komponenten erlaubt den automatischen Austausch durch eine äquivalente Komponente zur Sicherstellung des fehlerfreien Betriebs der AAL-Umgebung.

Diese Vorteile erkaufte man sich in erster Linie mit einem gewissen Mehraufwand bei der Erstellung der semantischen Beschreibungen der Komponenten. Es konnte aber gezeigt werden, dass durch die Bereitstellung geeigneter Werkzeuge, Dokumentationen und Schulungsunterlagen der Entwickler soweit unterstützt wird, dass es nicht direkt als Nachteil wahrgenommen wird.

Die Trennung der konfigurationsrelevanten Daten in öffentliche, welche in der gesamten AAL-Umgebung verfügbar sind, und AAL-anwendungsfallsspezifische, private Daten, stellt einen Kompromiss zwischen Flexibilität und Datenschutz der Akteure dar.

Insgesamt überwiegen die Vorteile einer ontologiebasierten Konfiguration für das flexible Management. Die semantischen AAL-Umgebungen bieten eine geeignete Basis, welche ergänzt durch das in dieser Arbeit beschriebene Framework, die relevanten Komponenten geeignet erweitert und somit ihre flexible Verbindung zulässt.

Hierdurch ist gezeigt, dass ein semantisches Konfigurationsframework die Heterogenität in AAL-Umgebungen verringern kann. Dies erfolgt durch geeignete Prozesse, Werkzeuge und einen ontologiebasierten Austausch von Informationen zwischen den einzelnen Integrationsschritten. Hierbei spielt die Wiederverwendung von Informationen im Konfigurationslifecycle und über mehrere Anwendungsfälle hinweg und die damit verbundenen Automatisierungen eine Schlüsselrolle.

7.2 Ausblick

Die ersten Ziele zur Steigerung der Flexibilität in AAL-Umgebungen werden mit dem in dieser Arbeit entwickelten Konfigurationsframework erreicht. Das entwickelte Konfigurationsframework ist als Referenzimplementierung in das AAL-Referenzmodell von universAAL aufgenommen worden. Das Konfigurationsframework wird zusammen mit den anderen Komponenten des Referenzmodells von der Open Source Community AALOA¹ gewartet und weiterentwickelt.

Das Konfigurationsframework kann in seiner jetzigen Form direkt weiterverwendet werden und wird so beispielsweise im EU-Projekt ReAAL² eingesetzt. Es wird in 4000 Haushalten in Europa und deren AAL-Umgebungen zu einem breiten Feldtest von AAL-Anwendungsfällen einrichtet.

Aufbauend auf den Ergebnissen dieser Arbeit können neue Fragestellungen für anknüpfbare Forschungsarbeiten gestellt werden.

Zur Nutzung der Vorteile von Cloud Computing und Big Data ergeben sich neue Herausforderungen für die Konfiguration der AAL-Umgebungen. Die Verlagerung von Teilen der Software und Nutzerdaten aus der AAL-Umgebung in die Cloud verlangt eine Anpassung des Konfigurationsframeworks und dessen Lifecycles sowie eine Überarbeitung beziehungsweise Erweiterung der Konzepte. Die in der Cloud

¹<http://www.aaloa.org> (Zugriff am 21.11.2014)

²<http://www.cip-reaal.eu/> (Zugriff am 21.11.2014)

zusammenlaufenden Informationen aus verschiedenen AAL-Umgebungen können durch Ansätze aus Big Data weiterverarbeitet werden und einen Nutzen für die einzelne Umgebung erzeugen (z.B. Feueralarm über mehrere Wohnungen in einem Wohnhaus). Da aber gerade im AAL-Kontext höchst sensible Informationen über die Nutzer (z.B. medizinische Daten) anfallen besteht hier weiterer Forschungsbedarf. Im Zuge der Konfiguration muss nicht nur auf deren geeignete Persistierung und Rechteverwaltung geachtet werden, sondern auch Missbrauch aktiv vorgebeugt werden und rechtliche Rahmenbedingungen beachtet werden. Hierzu gehören neue Konzepte, die es erlauben, dass auch nicht-techniknahe Personen erfassen können, welche Daten sie über sich preisgeben und was daraus interpretierbar ist. Die Teilung der Daten in Konfigurationsparameter, welche nur durch den authentifizierten AAL-Anwendungsfall zugänglich sind und Wissensobjekte, die in der gesamten AAL-Umgebung zu Verfügung stehen, ist hierbei nur der erste Schritt.

Ein weiteres ausbaufähiges Themenfeld stellt sich im Bereich der Automatisierung der Konfiguration. Es kann vor allem in den späteren Phasen des Konfigurationslifecycle auf die gesammelten Informationen aus früheren Phasen oder bereits in der AAL-Umgebung vorliegende Informationen zugegriffen werden. So könnte eine bessere Automatisierung der Installation, der Personalisierung und der Wartung erreicht werden. Besonders das automatische Laden von Notfallkonfigurationen zur Reaktion auf Unregelmäßigkeiten in der AAL-Umgebung, wie Sensorausfall, ist eine der großen Stärken der Automatisierung. Meist stellt sich aber schon vorab die Frage, wie sehr man bestimmten Informationen vertrauen darf. Hier wäre es interessant zu beleuchten, wie eine Konfiguration unter Unsicherheit und damit verbundenen Reasoning auf der Ontologie aussehen könnte, welche die Robustheit der AAL-Umgebung weiterhin garantieren.

Durch die technische und medizinische Entwicklung werden sich auch die AAL-Anwendungsfälle sowie die AAL-Umgebungen weiterentwickeln. Hier zeichnet sich der Bedarf ab, wie sich die Ontologien zur Hardware-, Software- und Umgebungsbeschreibung im Sinne einer Evolution weiter entwickeln können und gleichzeitig vorhandene AAL-Umgebungen mit deren Installationen abwärtskompatibel unterstützen. Die Möglichkeiten der momentan verwendeten Ontologie-Repositories reichen hierfür nicht aus und müssen in Richtung eines Ontologie-Evolutionsmanagements erweitert werden.

Literaturverzeichnis

- [ABBC⁺11] C. Ardito, B. Barricelli, P. Buono, M. Costabile, R. Lanzilotti, A. Piccinno und S. Valtolina. An Ontology-Based Approach to Product Customization. In *End-User Development*, Band 6654, S. 92–106. Springer, 2011.
- [AKBK09] A. Andrushevich, R. Kistler, M. Bieri und A. Klapproth. ZigBee / IEEE 802 . 15 . 4 Technologies in Ambient Assisted Living Applications. In *3rd European ZigBee Developers' Conference (EuZDC)*, Munich, 2009.
- [Amig14] Amigo. Amigo Project Page: <http://www.hitech-projects.com/euprojects/amigo/index.htm>, Letzter Zugriff: Juni 2014.
- [BaPD05] P. Ballon, J. Pierson und S. Delaere. Test and experimentation platforms for broadband innovation: Examining European practice. In *16th European Regional Conference by the International Telecommunications Society (ITS)*, 2005, S. 1–22.
- [Baum08] J. Baum. IST Amigo Project, Deliverable D4.7: Intelligent User Services, 2008.
- [Benc06] I. Benc. Middleware Platform for eMPOWERing cognitive disabled and elderly D1.3: Service lifecycle model, 2006.
- [BKHS09] B. Bergvall-Kareborn, M. Holst und A. Stahlbrost. Concept Design with a Living Lab Approach. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 2009, S. 1–10.
- [BoCo08] D. Bonino und F. Corno. DogOnt - Ontology Modeling for Intelligent Domestic Environments. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin und K. Thirunarayan (Hrsg.), *The Semantic Web - ISWC 2008 Lecture Notes in Computer Science Volume 5318*, Band 5318 der *Lecture Notes in Computer Science*, Berlin, Heidelberg, 2008. Springer, S. 790–803.
- [BoDo06] J. Bortz und N. Doering. *Forschungsmethoden und Evaluation: Für Human- und Sozialwissenschaftler*. Springer. 2006.
- [BWRAA⁺07] J. P. Bantle, J. Wylie-Rosett, A. L. Albright, C. M. Apovian, N. G. Clark, M. J. Franz, B. J. Hoogwerf, A. H. Lichtenstein, E. Mayer-Davis, A. D. Mooradian und M. L. Wheeler. Nutrition Recommendations and Interventions for Diabetes: a position statement of the

- American Diabetes Association. *Diabetes care* Band 30 Suppl 1, Januar 2007, S. S48–65.
- [CaNM83] S. K. Card, A. Newell und T. P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA. 1983.
- [ChAp07] H. W. Chesbrough und M. M. Appleyard. Open Innovation and Strategy. *California Management Review* Band 50, 2007, S. 57–77.
- [Ches06] H. Chesbrough. Open Innovation: A New Paradigm for Understanding Industrial Innovation. *Open innovation: researching a new paradigm*, 2006, S. 1–12.
- [CoDa04] D. Cook und S. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley. 2004.
- [Cont13] Contronics. Contronics GmbH Hausautomation und Gebäudeautomation: <http://www.contronics.de>, Letzter Zugriff: August 2013.
- [Core07] CoreLabs. Living Labs Roadmap 2007-2010: Recommendations on Networked Systems for Open User-Driven Research. Technischer Bericht, Lulea University of Technology, Centrum for Distance Spanning Technology, Lulea, 2007.
- [CTPRC09] C. Cetina, P. Trinidad, V. Pelechano und A. Ruiz-Cortés. Customisation along Lifecycle of Autonomic Homes. In *3rd International Workshop on Dynamic Software Product Line (DSPL09)*, 2009.
- [dCus14] N. de Cusa. *De concordantia catholica; libri tres patris Nicolai de Cusa*. Ascensius, Paris. 1514.
- [Denk14] R. Denker. ReAAL - Presentation of use-case collection: http://www.cip-reaal.eu/fileadmin/content/press/MACSI_2014_S2.1_02_Roelker-Denker.pdf, Letzter Zugriff: Juni 2014.
- [DoYS11] M. Dong, D. Yang und L. Su. Ontology-based service product configuration system modeling and development. *Expert Systems with Applications* Band 38, 2011, S. 11770–11786.
- [Eber08] B. Eberhardt. VDE-Positionspapier: Intelligente Assistenz-Systeme. Technischer Bericht, VDE, Frankfurt, 2008.
- [Eber09] B. Eberhardt. Zielgruppen für AAL. Technischer Bericht, VDE, Frankfurt, 2009.
- [Eber11] B. Eberhardt. AAL-Anwendungsszenarien. Technischer Bericht, VDE, Frankfurt, 2011.
- [Empi07] Empirica. Service orientated programmable smart environments for older Europeans - D1.2.2 SOPRANO Requirements Specification, 2007.

- [ENOK05] M. Eriksson, V.-p. Niitamo, N. Oyj und S. Kulkki. State-of-the-art in utilizing Living Labs approach to user- centric ICT innovation - a European approach . *Technology* Band 1, 2005, S. 1–13.
- [ENOL14] ENOLL. European Network of Living Lab. Technischer Bericht, ENOLL, 2014.
- [esUG14] embedded systems UG. Vera Logic Machine Plattform for KNX/EIB, EnOcean, Modbus, DALI, BacNet: <http://openrb.com>, Letzter Zugriff: Juni 2014.
- [Geor05] N. Georgantas. IST Amigo Project, Deliverable D2.1: Specification of the Amigo Abstract Middleware Architecture, 2005.
- [GrCa87] R. B. Grady und D. L. Caswell. *Software Metrics: Establishing a Company-wide Program*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 1987.
- [Grub95] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies* 43(5-6), November 1995, S. 907–928.
- [Grun10] A. Grunwald. *Technikfolgenabschätzung: Eine Einführung*. edition sigma. 2010.
- [Guil13] S. Guillen. universAAL Consortium. D4.1, Innovative AAL Service Specification, 2013.
- [GuLa07] Q. Gu und P. Lago. A stakeholder-driven service life cycle model for SOA. In *2nd International Workshop on Service-oriented Software Engineering in Conjunction with the 6th ESEC/FSE Joint Meeting - IW-SOSWE '07*, New York, September 2007. ACM Press, S. 1–7.
- [HMPr04] A. R. Hevner, S. T. March, J. Park und S. Ram. Design science in information systems research. *MIS Quarterly* 28(1), März 2004, S. 75–105.
- [Howe06] J. Howe. The Rise of Crowdsourcing. *North* Band 14, 2006, S. 1–5.
- [IsGVG08a] K. V. Isacker, M. Goranova-Valkova und P. Grudeva. OASIS – Open architecture for Accessible Services Integration and Standardization D1.6.1: OASIS Architecture and Component Specification, 2008.
- [IsGVG08b] K. V. Isacker, M. Goranova-Valkova und P. Grudeva. OASIS – Open Architecture for Accessible Services Integration and Standardization D5.5.1: Guidelines on UCD Requirements Extraction, 2008.
- [ISO/10] ISO/IEC. ISO/IEC 25010 - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models, 2010.

- [JVSC⁺06] I. Journal, K.-b. S. Vol, W. Scientific, P. Company, A. A. Dpto, S. Inform, E. Superior und T. L. Dpto. Consistency in the analytic hierarchy process: a new approach. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 14(4), 2006, S. 445–459.
- [KaNF08] F. Kawsar, T. Nakajima und K. Fujinami. Deploy spontaneously: supporting end-users in building and enhancing a smart home. *Proceedings of the 10th International Conference on Ubiquitous computing (2008)*, September 2008, S. 282–291.
- [KaRy97] J. Karlsson und K. Ryan. A cost-value approach for prioritizing requirements. *Software, IEEE* 14(5), Sep 1997, S. 67–74.
- [Kier01] D. Kieras. Using the Keystroke-Level Model to Estimate Execution Times. Technischer Bericht, MIT, 2001.
- [Kirw13] M. Kirwan. Interoperability design guidelines for personal health systems. Technischer Bericht, Continua, 2013.
- [Kön13] R. König. Home of FHEM: <http://www.fhem.de>, Letzter Zugriff: August 2013.
- [Lamn05] S. Lamnek. *Qualitative Sozialforschung. Methoden und Techniken*. Psychologie Verlags Union. 2005.
- [LaSm10] K. Larson und D. Smithwick. Beyond the Configurator : Collecting accurate data for an architectural design recommendation engine. In *Working Paper*, 2010, S. 1–7.
- [LeHT14] L. Lefort, C. Henson und K. Taylor. Semantic Sensor Network XG Final Report: <http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/>, Letzter Zugriff: Juni 2014.
- [LMDM⁺12] M. Lipprandt, A. Marinc, T. Dutz, G. Moritz, M. Eichelberg, R. Wichert und A. Hein. Evaluation of AAL Middleware Platforms. In *Tomorrow in Sight: From Design to Delivery. Proceedings of the 4th AAL Forum Eindhoven*, Dezember 2012, S. 197–203.
- [Loxo14] Loxone. Loxone GmbH - Smart Home Automation: <http://www.loxone.com>, Letzter Zugriff: Juni 2014.
- [MaMG02] M. L. Markus, A. Majchrzak und L. Gasser. A Design Theory for Systems that Support Emergent Knowledge Processes. *MIS Quarterly* 26(3), 2002, S. 179–212.
- [MBOM⁺06] D. Martin, M. Burstein, N. Ora Lassila, D. McDermott, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan und K. Sycara. OWL-S: Semantic Markup for Web Services. Technischer Bericht, SRI International, 2006.
- [MeNa08] M. Meuser und U. Nagel. ExpertInneninterviews - vielfach erprobt, wenig bedacht : ein Beitrag zur qualitativen Methodendiskussion, September 2008.

- [MiCa14] MiCasaVerde. Vera Smart Home Controllers: <http://www.micasaverde.com>, Letzter Zugriff: Juni 2014.
- [MLMB⁺06] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown und R. Metz. OASIS Open Technical Committee for SOA-RM, Reference model for service oriented architecture 1.0, 2006.
- [MPOW14] MPOWER. MPOWER Project Page: <http://www.sintef.no/Projectweb/MPOWER/>, Letzter Zugriff: Juni 2014.
- [MuRW08] K. Mueller, P. Rumm und R. Wichert. Persona - ein EU-Projekt für Unabhängigkeit und Lebensqualität im Alter. In *Ambient Assisted Living. 1. Deutscher Kongress mit Ausstellung 2008. Technologien - Anwendungen - Management*, 2008.
- [Nara13] J.-C. Naranjo. universAAL Consortium. D1.3, universAAL Reference Architecture, 2013.
- [Niel12] J. Nielsen. How Many Test Users in a Usability Study? <http://www.nngroup.com/articles/how-many-test-users/>, 2012.
- [NoMo06] U. Norbistrath und C. Mosler. Tool Support for the eHome Specification, Configuration, and Deployment Process. *MSPE*, 2006, S. 109–122.
- [OASI14] OASIS. OASIS Project Page: <http://www.oasis-project.eu/>, Letzter Zugriff: Juni 2014.
- [open14a] openhab. openhab - empowering the smart home: <http://www.openhab.org>, Letzter Zugriff: Juni 2014.
- [Open14b] OpenRemote. Open Source for Internet of Things: <http://www.openremote.com>, Letzter Zugriff: Juni 2014.
- [opti13] optimAAL. Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS. optimAAL Kompetenz: <http://www.aal-kompetenz.de/>, Letzter Zugriff: Juni 2013.
- [oTec14] G. I. of Technology. Aware Home Research Initiative (AHRI), 2014.
- [PaMa01] P. Paper und P. Markopoulos. Towards a Living Lab research facility and a ubiquitous computing research programme. *Proceedings of the Conference for Human-computer Interaction (CHI)*, 2001, S. 3–4.
- [Raal13] Raali. OFFIS - Institut für Informatik. Roadmap AAL - Interoperabilität: <http://www.raali.de/>, Letzter Zugriff: April 2013.
- [Ram13] R. Ram. universAAL Consortium. D1.3, Laboratory Installation, 2013.
- [RaZS11] A. Rashid, T. Zentek und O. Strnad. Living Labs als Forschungsansatz für Ambient Assisted Living (AAL). In M. Eibl und M. Ritter (Hrsg.), *Mensch & Computer Workshopband*. Universitätsverlag Chemnitz, 2011, S. 335–340.

- [RFGIS⁺13] R. Ram, F. Furfari, M. Girolami, G. Ibaiez-Sanchez, J.-P. Lazaro-Ramos, C. Mayer, B. Prazak-Aram und T. Zentek. universAAL: Provisioning Platform for AAL Services. In A. van Berlo, K. Hallenborg, J. M. C. Rodriguez, D. I. Tapia und P. Novais (Hrsg.), *ISAmI*, Band 219 der *Advances in Intelligent Systems and Computing*. Springer, 2013, S. 105–112.
- [Rich10] E. Richter-Kuhlmann. Die Lücken werden größer. *Deutsches Ärzteblatt* 107(September), 2010, S. 1670–1672.
- [RivdM14] J. Rivera und R. van der Meulen. Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020: <http://www.gartner.com/newsroom/id/2636073>, Letzter Zugriff: Juni 2014.
- [Robl08] M. G. Robledo. Accessibility and Usability Validation Framework for AAL Interaction Design Process, D6.1 - Project Description, 2008.
- [RRRZ12] A. Rashid, C. Reichelt, N. Röhl und T. Zentek. Living Labs als Forschungsinstrument für Ambient Assisted Living Technologien. *i-com* 11(3), 2012, S. 24–29.
- [RuWo14] V. Rupp und B. Woodworth. Open Source Automation: <http://www.opensourceautomation.com>, Letzter Zugriff: Juni 2014.
- [RyJC13] E. T. Ryan, D. R. Jacques und J. M. Colombi. An ontological framework for clarifying flexibility-related terminology via literature survey. *Systems Engineering* 16(1), März 2013, S. 99–110.
- [Saat80] T. L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill. 1980.
- [Sala11] P. Sala. universAAL Consortium. D1.1, Reference Use Cases for AAL, 2011.
- [Saur09] J. Sauro. Estimating Productivity: Composite Operators for Keystroke Level Modeling. In J. Jacko (Hrsg.), *Human-Computer Interaction. New Trends*, Band 5610 der *Lecture Notes in Computer Science*, S. 352–361. Springer Berlin Heidelberg, 2009.
- [SaWe98] D. Sabin und R. Weigel. Product Configuration Frameworks - A Survey. *IEEE Intelligent Systems and their Applications* Band 13, 1998.
- [Scha08] J. Schaefer. Accessibility and Usability Validation Framework for AAL Interaction Design Process, D1.1 - Usability Requirements, 2008.
- [SCHK⁺07] H. Schaffers, M. Cordoba, P. Hongisto, T. Kallai, C. Merz und J. Van Rensburg. Exploring business models for open innovation in rural living labs, Juni 2007.
- [Stat11] Statistisches Bundesamt. *Demografischer Wandel in Deutschland*. Statistische Ämter des Bundes und der Länder, Wiesbaden. 2011.

- [Stat14] Statistisches Bundesamt. *Gesundheit - Ausgaben - Fachserie 12 Reihe 7.1.1 - 2012*, Band 49. Statistisches Bundesamt, Wiesbaden. 2014.
- [Stum97] M. Stumptner. An overview of knowledge-based configuration. *AI Communications* 10(2), April 1997, S. 111–125.
- [Symc14] Symcon. iP-Symcon GmbH Automatisierungssoftware: <http://www.ip-symcon.de>, Letzter Zugriff: Juni 2014.
- [TFFVH⁺12] M.-R. Tazari, F. Furfari, A. Fides Valero, S. Hanke, O. Höftberger, D. Kehagias, M. Mosmondor, R. Wichert und P. Wolf. The universal Reference Model for AAL. In J. et al. Augusto (Hrsg.), *Handbook of Ambient Assisted Living*, S. 610 – 625. IOS Press, 2012.
- [TuSK13] R. Tuma, B. Schnettler und H. Knoblauch. *Videographie: Einführung in die interpretative Videoanalyse sozialer Situationen*. Springer Fachmedien Wiesbaden. 2013.
- [univ14] universAAL. universAAL Project Consortium Description: <http://www.universaal.org>, Letzter Zugriff: September 2014.
- [Vaal14] Vaalid. Vaalid Project Page: <http://www.vaalid-project.org/>, Letzter Zugriff: Juni 2014.
- [vdBCW10] G. van den Broek, F. Cavallo und C. Wehrmann (Hrsg.). *AALIANCE Ambient Assisted Living Roadmap*. IOS Press. 2010.
- [VDE 12a] VDE Verband der Elektrotechnik Elektronik Informationstechnik. Die deutsche Normungs-Roadmap AAL. Technischer Bericht, DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE, Frankfurt, 2012.
- [VDE 12b] VDE Verband der Elektrotechnik Elektronik Informationstechnik. Service Wohnen zu Hause - Kriterien für die Auswahl und Installation von AAL-Komponenten, 2012.
- [VDE 12c] VDE Verband der Elektrotechnik Elektronik Informationstechnik. Service Wohnen zu Hause - Qualitätskriterien für Anbieter, Dienstleistungen und Produkte für technikunterstütztes Leben (AAL), 2012.
- [vHip95] E. von Hippel. Sources of Innovation - Contents. In *The Sources of Innovation*, S. 218. Oxford University Press, 1995.
- [vHip05] E. von Hippel. Democratizing innovation: The evolving phenomenon of user innovation. *Journal für Betriebswirtschaft* Band 55, 2005, S. 63–78.
- [VRLT10] P. A. Valiente-Rocha und A. Lozano-Tello. Ontology-based expert system for home automation controlling. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Band 6096 LNAI, 2010, S. 661–670.

- [Wald07] S. Walderhaug. Middleware Platform for eMPowering cognitive disabled and elderly D1.1: Overall Architecture, 2007.
- [WBKL+13] R. Welge, B.-H. Busch, K. Kabitzsch, J. Laurilaepe, S. Heusinger, M. Lipprandt, E. Eichenberg, H. Engeli, M. Gök, G. Moritz und A. Hein. Ontologiebasierte Repräsentation von Integrationsprofilen Representation of Integration Profiles using an Ontology Kurzfassung Interoperabilität – Eine Herausforderung für AAL Anwendungsfallbasierte Integrations- Wissensmanagement zur Lösung von. In *Ambient Assisted Living - Advanced Technologies and Societal Change*, 2013, S. 397–406.
- [Weis09] C. Weiss. Altersgerechte Assistenzsysteme für ein gesundes und unabhängiges Leben. Technischer Bericht, Bundesministerium für Bildung und Forschung (BMBF), 2009.
- [WMSSr+12] S. Walderhaug, M. Mikalsen, D. Salvi, I. Svagård, D. Ausen und A. Kofod-Petersen. Towards quality assurance of AAL services. In *Studies in Health Technology and Informatics*, Band 177, 2012, S. 296–303.
- [YMWZ09] D. Yang, R. Miao, H. Wu und Y. Zhou. Product configuration knowledge modeling using ontology web language. *Expert Systems with Applications* Band 36, 2009, S. 4399–4411.
- [ZeMR13] T. Zentek, A. Marinc und A. Rashid. Methods and Tools for Ontology-based Configuration Processes of AAL Environments. In R. Wichert und H. Klausning (Hrsg.), *Ambient Assisted Living - Advanced Technologies and Societal Change*, S. 213–224. Springer, Berlin Heidelberg, 2013.
- [ZeRa11] T. Zentek und A. Raschid. Hardwareabhängige Projektevolution: Vom Mobiltelefon zur Applikationsplattform. Wie entwickle ich (zukünftig) eine App für AAL-Umgebungen? In *Telemed 2011*, 2011.
- [ZeRa12] T. Zentek und A. Rashid. Methoden und Werkzeuge zur Konfiguration von AAL Anwendungen zur Steigerung von Innovationen in stationärer und ambulanter Pflege. In U. Goltz, M. A. Magnor, H.-J. Appelrath, H. K. Matthies, W.-T. Balke und L. C. Wolf (Hrsg.), *GI-Jahrestagung*, Band 208 der LNI. GI, 2012, S. 1215–1227.
- [ZeWo11] T. Zentek und P. Wolf. Ontologiebasierte Konfigurationsprozesse in AAL Umgebungen. In *Informatik 2011 Lecture Notes in Informatics, Band P192*, Berlin, 2011. S. 172–181.
- [ZRWK09] T. Zentek, A. Rashid, P. Wolf und C. Kunze. Mit Plug&Play zur intelligenten Wohnung: Ein Referenzmodell zum Einrichten und Verwalten einer Ambient Assisted Living Umgebung. In S. Fischer, E. Maehle und R. Reischuk (Hrsg.), *GI Jahrestagung*, Band 154 der LNI. GI, 2009, S. 956–969.

- [ZYRR14] T. Zentek, C. Yumusak, C. Reichelt und A. Rashid. Which AAL Middleware matches my requirements? An Analysis of Current Middleware Systems and a Framework for Decision-Support. In R. Wichert und H. Klausning (Hrsg.), *Ambient Assisted Living - Advanced Technologies and Societal Change*, S. 111–125. Springer, Berlin Heidelberg, 2014.

A. Material zu den Evaluationen

A.1 Integrationsanleitung für den „Nutritional Advisor“

In diesem Kapitel finden sich die Kurzanweisungen für die Mitarbeiter in den Living Labs, welche am Pretests teilnehmen. Weiterführendes Material findet sich in [Ram13].

Installation Instructions – uStore and uCC

The Nutritional Advisor service is installed from uStore by using the uCC tool that was installed in the previous step by the runner. The instructions for searching, purchasing and installing a service from uStore are detailed in D3.4-C part V – Manuals.

- 1 Open section '2.1.4 Purchase AAL Services' in D3.4-C Part V – Manuals and follow the steps
 - 1.1 Step #2 – The Nutritional Advisor service is located in the catalog under the category 'AAL Services > Nutrition Services'
- 2 Open section '2.1.5 Download and install AAL Services' in D3.4-C Part V – Manuals and follow the steps
 - 2.1 Step #5 (download of the service) – skip
 - 2.2 Step #8 (user preferences) – All the fields are already fulfilled
- 3 Currently the installation of a service by the uCC still needs some manual configurations to be done. Follow the steps to complete the installation process:
 - 3.1 Stop the prototype, go to the running console and type 'stop 0'
 - 3.2 Add the Nutritional Advisor service to the main menu of universAAL by copying all the files under the protoOdense.runner/runner/configurations/ ui.dm.mobile/Backup_with_NA folder

to the protoOdense.runner/runner/
configurations/ui.dm.mobile folder

3.3 Copy the jars that are included in the protoOdense.runner/runner/
bundles/NutritionalAdvisor folder to the protoOdense.runner/
runner/bundles folder

3.4 Copy the config.ini file from the protoOdense.runne/runner/felix/
Runner_with_NA folder to the protoOdense.runne/runner/felix folder

3.5 Restart the prototype - open a command prompt and go to protoOden-
se.runner/runner and execute the command run.bat

4 Verify that the Nutritional Advisor service was added to the universAAL main
menu

A.2 Fragebogen zum Pretest

Im folgenden wird der Online-Fragebogen des Pretests abgedruckt:

Questionnaire Service Installation

In order to evaluate the service installation process, please take some time and
answer the following questions.

Q_{pre0}: Name [optional]:

Q_{pre1}: What is your role in universAAL? [Drop down] Choose One:

Q_{pre2}: Which method for installation are you performing? [Drop down] Choose
One:

Q_{pre3}: Have you performed any other uAAL installation method before? [Yes/No]

Q_{pre4}: If yes, which one(s)? Select the one(s) you have already installed and order
them from the least challenging installation to the most challenging [Drop
downs] Installation Method 1: Installation Method 2: Installation Method 3:

Q_{pre5}: Did you succeed with the installation and the service(s) is/are finally run-
ning? [Yes/No]

Q_{pre6}: What was/were the problem(s)/challenge(s) you encountered in the instal-
lation process? [Open question]

Q_{pre7}: How long did the installation process approximately take? [Answer in mi-
nutes]

Q_{pre8}: What were the main challenges for installation? [open]

Q_{pre9}: How often did you need support by the developers and contacted them in
the installation process? [numeric value]

Q_{pre10}: Did you encounter any problem(s) with the provided documentation?
[Yes/No]

Q_{pre11}: If yes, what was/were the problem(s)? [Open]

Q_{pre12}: Any other comments? [open]

A.3 Fragebogen zu der Anforderungsevaluation

Krakow, 9th universAAL Plenary Meeting, 6-9 November 2012	Prototype Demonstration R1.3																												
<p><i>Dear partner,</i> Goal of this questionnaire is to assess the requirements of the uCC (universAAL Control Center) from a deployer's point of view. Please answer the questions from his viewpoint and keep in mind that the deployer is not an end-user, but a skilled professional, who manages and executes the deployment process of an AAL service. This process includes the purchase, installation and configuration of an AAL service, which results in a running and personalized service for the end-user. Purpose of the uCC is to assist the deployer in each phase of the life cycle of the AAL service (from purchase to uninstallation).</p> <p>Thank you for participating in this survey.</p>																													
BLOCK II: uCC Requirements Evaluation																													
Personal Information																													
<p>Q1: What is your primary role in regards to AAL?</p> <p> <input type="radio"/> Developer <input type="radio"/> Deployer <input type="radio"/> Other: _____ </p> <p>Q2: How long have you been working in the AAL domain?</p> <p>_____ years</p> <p>Q3: What gender are you?</p> <p> <input type="radio"/> Female <input type="radio"/> Male </p> <p>Q4: How old are you?</p> <p>_____ years</p>																													
Shop for AAL services in the uStore																													
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;"></th> <th style="width: 10%;">Strongly Disagree</th> <th style="width: 10%;">Disagree</th> <th style="width: 10%;">Neutral</th> <th style="width: 10%;">Agree</th> <th style="width: 10%;">Strongly Agree</th> <th style="width: 10%;">Don't know</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;"> Q5: I as a deployer would like to have a one-click-installation of AAL applications from the uStore, i.e. the complete integration in the environment is done by one click. </td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> </tr> <tr> <td style="padding: 5px;"> Q6: I the deployer would like to have always a manual way (without using the uCC) to process a task (e.g. install, configure etc.). </td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> </tr> <tr> <td style="padding: 5px;"> Q7: For me as a deployer it would not be necessary to access the uStore directly in the uCC without an external browser. </td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> </tr> </tbody> </table>		Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Don't know	Q5: I as a deployer would like to have a one-click-installation of AAL applications from the uStore, i.e. the complete integration in the environment is done by one click.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Q6: I the deployer would like to have always a manual way (without using the uCC) to process a task (e.g. install, configure etc.).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Q7: For me as a deployer it would not be necessary to access the uStore directly in the uCC without an external browser.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Don't know																							
Q5: I as a deployer would like to have a one-click-installation of AAL applications from the uStore, i.e. the complete integration in the environment is done by one click.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																							
Q6: I the deployer would like to have always a manual way (without using the uCC) to process a task (e.g. install, configure etc.).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																							
Q7: For me as a deployer it would not be necessary to access the uStore directly in the uCC without an external browser.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																							
1 / 3																													

Abbildung A.1: Fragebogen Seite 1

Krakow, 9th universAAL Plenary Meeting,
6-9 November 2012

Prototype Demonstration R1.3

Installation of AAL services						
	<i>Strongly Disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly Agree</i>	<i>Don't know</i>
<i>Q8: I the deployer would give up some flexibility in the configuration process to optimize the required service installation time with the uCC.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q9: I as a deployer rather have low resource requirements for the uCC than complex fallback mechanisms (e.g. data recovery in case of a system failure during the application integration).</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q10: I in the role of a deployer would like to use mobile devices (e.g. cell phone, tablet-pc) for the deployment process.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Configuration and personalization of AAL services to end-user needs						
	<i>Strongly Disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly Agree</i>	<i>Don't know</i>
<i>Q11: There should always be a default configuration to support me in my role as a deployer.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q12: During the configuration process, I (the deployer) would trade some limitation in my choice of features for error protection (e.g. pre-filled values etc.).</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q13: I think the uCC is not required to validate the user-set configuration details of a service.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Uninstallation of AAL services						
	<i>Strongly Disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly Agree</i>	<i>Don't know</i>
<i>Q14: I am the deployer. I would like to have the possibility to revoke the uninstallation of AAL services.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Abbildung A.2: Fragebogen Seite 2

Krakow, 9th universAAL Plenary Meeting,
6-9 November 2012

Prototype Demonstration R1.3

General questions about the uCC related to AAL service integration						
	<i>Strongly Disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly Agree</i>	<i>Don't know</i>
<i>Q15: It is okay for the uCC to be difficult and complex, since the user needs extensive functionality.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q16: The uCC should also be extendible with plug-in functionality (e.g. log view monitoring, editing user and space profiles) apart from installing, managing and uninstalling services.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q17: The uCC system resources requirements are of little importance.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q18: To use the uCC properly, the user needs a particular amount of training.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q19: The uCC usability (GUI, error tolerance etc.) is more important than the performance efficiency.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<i>Q20: The uCC should be extended to manage human- and hardware resources, even if this means losing some usability and focus on core features.</i>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
General comments						

THANK YOU!

3 / 3

Abbildung A.3: Fragebogen Seite 3

A.4 AHP-Berechnung

In der Abbildung A.4 sind die Berechnungen zur AHP-Matrix zu sehen auf deren Basis das Ranking der Qualitätsanforderungen erstellt wurde.

Participant	Q1	Q1_01_0th	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Update	CR	Reliability	Usability	Performance	Efficiency	Functional Suitability	CR	GCI	lambda_max	VALID
1	0	End user representative	5	2	27	3	1	1	2	2	3	2	0	1	0	3	1	1	1	1	1	3	Parameters	0.1490885	25.30%	26.49%	23.07%	23.07%	0.1490885	0.0984671	4.4047539	1
2	0	Project Manager	3	2	35	2	1	1	2	3	3	2	1	3	-1	3	-1	2	0	0	2	Consistency threshold	0.0523089	28.73%	34.79%	17.84%	16.63%	0.0523089	0.0356294	4.1836483	1	
3	0	Researcher	8	1	39	3	2	2	1	2	2	2	0	2	0	2	0	2	0	0	2	0.80	0.02247898	28.81%	33.81%	15.90%	16.90%	0.02247898	0.01516939	4.06068783	1	
4	1		5	2	33	3	-1	2	2	2	3	2	3	2	0	2	3	-1	2	3	-2	0.07978353	32.14%	28.81%	18.81%	20.24%	0.07978353	0.0531498	4.21541554	1		
5	1		3	2	27	3	1	2	2	1	3	2	3	3	2	-1	2	-1	2	2	2	min answered questions	0.11375522	24.66%	35.62%	25.62%	25.62%	0.11375522	0.07501948	4.30713908	1	
6	1		6	2	43	3	0	2	0	3	3	2	0	3	0	3	1	0	-1	2	8	0.02241513	34.58%	24.58%	20.42%	20.42%	0.02241513	0.0151308	4.06053165	1		
7	0		1	2	31	3	3	0	2	2	-1	3	2	2	0	0	-1	3	2	3	-1	show paper Q	0.09021668	23.77%	30.91%	15.99%	15.99%	0.09021668	0.05999003	4.24338505	1	
8	1		5	2	39	3	0	2	0	2	2	2	0	2	0	0	3	2	0	0	3	show stragglers	0.05732217	33.63%	26.49%	16.82%	16.82%	0.05732217	0.03832009	4.15476985	1	
9	1		2	2	31	3	3	0	2	1	2	1	2	1	2	0	0	0	3	0	3	1	0	23.95%	25.38%	29.50%	29.50%	0.06878639	0.04580588	4.18577075	1	
10	0		3	1	38	2	1	0	1	2	1	2	1	2	1	2	2	0	0	0	3	1	25.00%	25.00%	17.71%	17.71%	0.0568829	0.03807025	4.15353834	0		
11	1		4	2	33	3	1	3	1	2	3	3	1	0	3	2	-1	3	2	-1	3	show online Q	0.0568829	40.63%	23.96%	17.71%	17.71%	0.0568829	0.03807025	4.15353834	1	
12	1		0	3	2	30	3	2	1	1	3	2	2	0	2	0	2	0	1	2	2	1	0.0449335	29.29%	29.29%	20.71%	20.71%	0.0449335	0.03026637	4.11312044	1	
13	2		0	2	55	3	0	2	3	3	1	-1	3	-1	1	0	-1	0	-1	0	2	Ci alpha	0.10599835	21.16%	35.30%	20.66%	23.86%	0.10599835	0.06980951	4.28615554	1	
14	1		5	2	45	3	2	1	0	2	3	1	0	2	0	2	0	2	0	3	0	0.150	16.90%	33.81%	16.90%	16.90%	0.02247898	0.01516939	4.06068783	1		
15	0	Technical manager	3	2	31	2	1	0	2	1	0	2	0	2	0	1	2	2	1	2	1	CI Threshold	0.02631132	29.13%	29.13%	18.38%	20.46%	0.02631132	0.0174862	4.07104056	1	
16	0	UAAI: WP8/WP9/WP7 Management	5	1	33	2	0	1	2	-1	1	2	-1	0	0	0	0	0	0	0	0	-1	4.409245	37.95%	37.95%	14.83%	14.83%	0.12866889	0.08442847	4.34745459	1	
17	0	Licensing	3	2	50	2	2	2	3	3	2	3	1	1	1	1	1	1	1	1	2	1	21.07%	24.64%	26.49%	21.07%	21.07%	0.12866889	0.08442847	4.34745459	1	
18	0	Researcher	3	2	31	3	0	2	2	2	2	2	2	2	1	3	0	2	2	1	2	1	21.83%	25.40%	29.64%	23.61%	23.61%	0.09010782	0.05978638	4.24329112	1	
19	0		4	2	30	3	0	3	2	1	3	2	2	2	-1	3	0	0	0	0	0	1	26.75%	20.32%	28.51%	24.42%	24.42%	0.14914184	0.09749327	4.40268297	1	
20	1		2	2	48	0	3	1	2	0	2	2	2	2	2	3	0	0	0	0	0	1	20.42%	34.58%	20.42%	20.42%	20.42%	0.02241513	0.0151308	4.06053165	0	
21	2	Nachzügler Krakow	8	1	35	3	0	1	0	-1	3	0	0	-1	2	1	0	0	-1	1	1	26.22%	26.22%	14.69%	14.69%	29.31%	29.31%	0.1196496	0.07891455	4.33206392	1	
22	0	Project Manager	6	2	35	3	3	1	2	2	1	3	3	0	2	3	2	3	2	3	1	3	23.89%	27.15%	24.05%	25.80%	25.80%	0.1289224	0.20779324	4.88809048	1	
23	1		2	1	39	3	2	2	2	2	3	2	0	1	2	2	2	2	2	2	3	3	24.85%	24.70%	24.85%	26.52%	26.52%	0.21915579	0.14192666	4.59171254	0	
24	1	online questionnaire	3	2	32	3	0	3	0	2	3	3	-1	2	0	3	2	2	2	2	0	1	35.50%	35.50%	14.50%	14.50%	0.01527461	0.01033605	4.04124146	1		
25	1		1	2	30	2	3	1	3	3	2	0	2	2	2	1	2	1	2	2	1	1	26.49%	25.30%	25.15%	23.07%	23.07%	0.14990885	0.09846471	4.4047539	1	
26	2		3	2	29	3	2	2	1	0	3	1	2	-1	1	2	-1	1	0	1	2	1	29.64%	29.64%	26.64%	21.07%	21.07%	0.0224478	0.01515654	4.06066306	1	
27	1		2	2	34	2	2	1	2	2	1	1	1	2	0	2	1	2	1	2	-1	3	15.95%	20.37%	33.38%	30.30%	30.30%	0.03213197	0.02162906	4.08675631	1	

Abbildung A.4: AHP-Berechnung

A.5 Fragebogen zur Frameworkevaluation

Berlin, 6. AAL-Kongress, 21. Januar 2013 universAAL Workshop

Danke für ihre Teilnahme am universAAL Workshop,

hoffentlich konnten wir Ihnen universAAL etwas näher bringen. In Ihrer Mappe finden Sie weiterführende Unterlagen mit Informationen zu den Themen des Workshops, sowie eine CD, damit Sie alles nochmal in Ruhe zu Hause ausprobieren können.

Für unsere Evaluation möchten wir Sie bitten, folgenden kurzen Feedbackbogen auszufüllen. Wir werden Ihre Angaben anonymisiert im Rahmen des Projektes universAAL auswerten.

Vielen Dank!
Ihr Tom Zentek

Informationen zu Ihrer Person

Was ist Ihre Rolle im AAL Umfeld?

primär Developer
(Entwicklung von AAL Diensten)

primär Deployer
(Integration von AAL Diensten in die AAL Umgebung)

Wie lange arbeiten Sie schon im AAL Umfeld?

_____ Jahre

Feedback

Umsetzung: Wie finden Sie die technische Umsetzung der Konfigurationskonzepte (z.B. Configitem, Validator, Listener)?	Einfach	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Aufwendig
Funktionalität: Wie schätzen sie den Funktionsumfang der vorgestellten Konzepte ein?	Unvollständig	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Vollständig
Leistungsfähigkeit: Können Sie Ihre AAL Dienste mit den vorgestellten Konfigurationskonzepten umsetzen?	Niedrig	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Hoch
Nachvollziehbarkeit: Wie verständlich sind die vorgestellten Konzepte?	Unverständlich	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Verständlich
Komplexität: Wie schätzen sie die Vielfalt der vorgestellten Konfigurationsmerkmale ein?	Niedrig	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Hoch
Gesamteindruck: Wie finden Sie die vorgestellten Konzepte und Umsetzungen insgesamt?	Unattraktiv	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Attraktiv

Weitere Kommentare

Abbildung A.5: Fragebogen Seite 1

A.6 Konfigurationsaufgaben der Frameworkevaluation

Dieser Anhang präsentiert die Unterlagen, die zur Bearbeitung der Aufgaben während der Evaluation den Konfigurationsframeworks während der Hands-On-Phase bereitgestellt wurden. Mit ihnen konnte jeder Teilnehmer die erlernten Konfigurationsprozesse und Werkzeuge selber ausprobieren. Ein Teil der Evaluationsauswertung stützt sich auf die hierbei produzierten Daten.

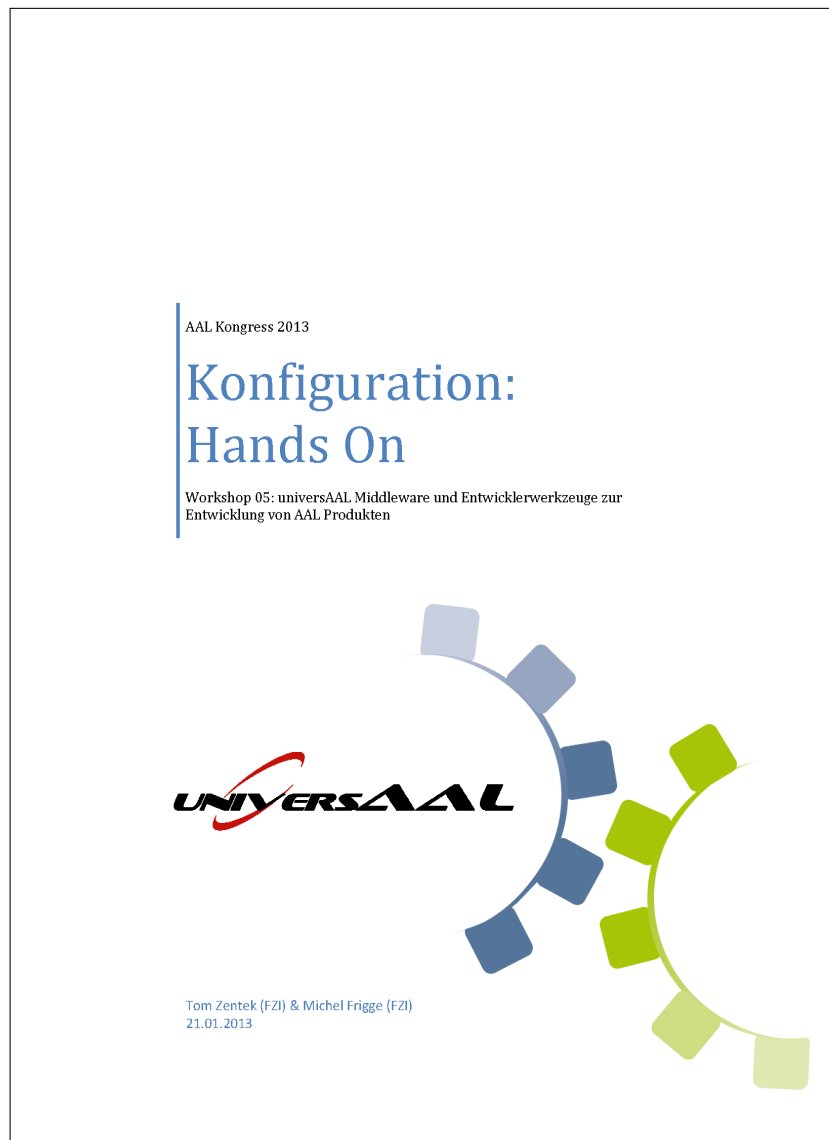


Abbildung A.6: Konfigurationsaufgaben Seite 1

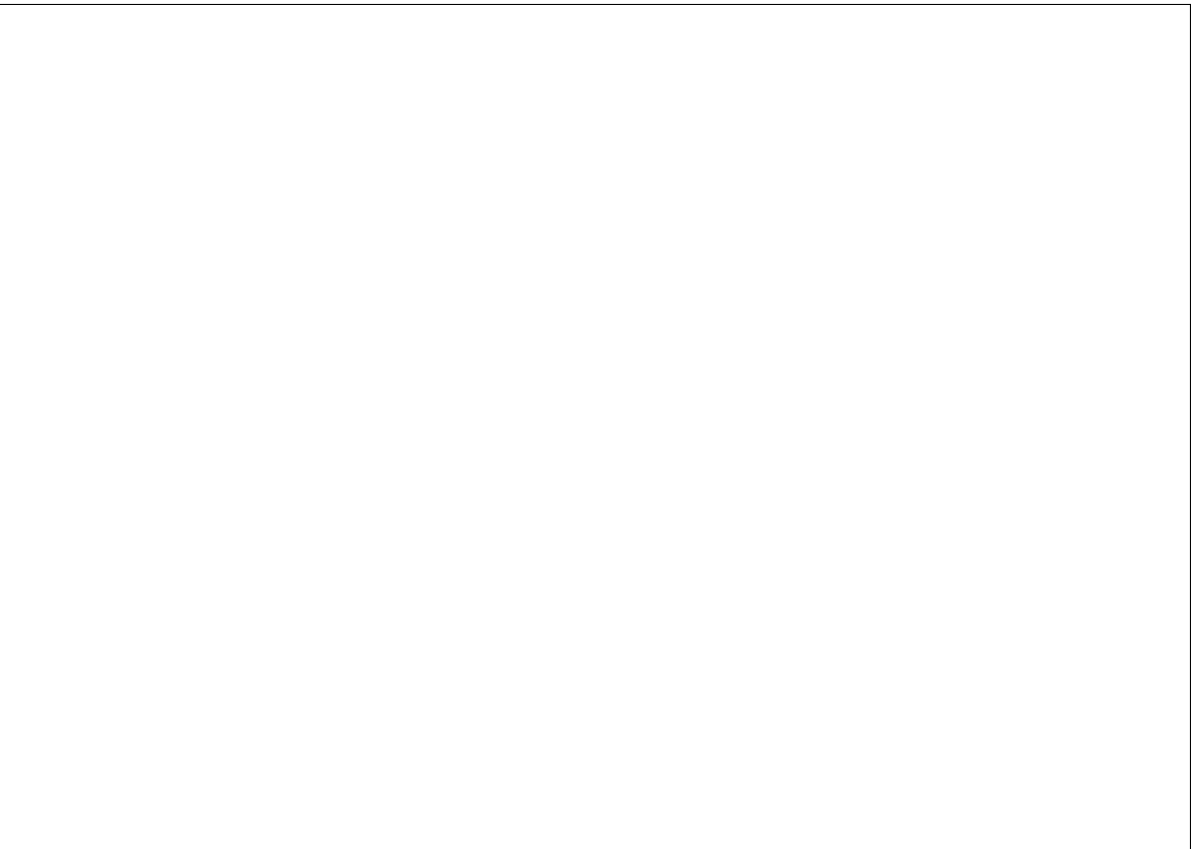


Abbildung A.7: Konfigurationsaufgaben Seite 2

Inhaltsverzeichnis

Einleitung.....	4
Aufgabe 1: Wettervorhersage.....	5
Teilaufgabe 1.1: Konfigurationsoption anlegen und auslesen.....	5
Teilaufgabe 1.2: Einstellungen testen.....	7
Teilaufgabe 1.3: Eingabe durch einen Standard-Validator überprüfen.....	8
Teilaufgabe 1.4: Konfigurationsoption bei aktiviertem Weiter-Plugin anzeigen.....	10
Aufgabe 2: Schriftfarbe anpassen.....	12
Teilaufgabe 2.1: Die Konfigurationsoption anlegen und auslesen.....	12
Teilaufgabe 2.2: Eingabe durch einen Validator überprüfen.....	14
Teilaufgabe 2.3: Implementierung eines Listeners.....	17

Abbildung A.8: Konfigurationsaufgaben Seite 3

Einleitung

Durch den Hands-On Teil dieses Workshops soll Ihnen die Möglichkeit gegeben werden, die zuvor vermittelten Grundlagen zu den Konfigurationsmechanismen in univarsAAL zu vertiefen. Hierbei werden sie einfache Konfigurationsparameter erstellen und testen. Danach verwenden Sie Validatoren, um die Eingaben auf Richtigkeit zu prüfen. Abschließend werden wir Listener ausprobieren, um direkt auf Konfigurationseingaben zu reagieren.

Die Beispiele werden anhand eines Infoframe Anwendungsfalls gestaltet. Der Infoframe ist hierzu mit geeigneten kommentierten Blöcken vorbereitet. Der Moderator wird die einzelnen Teilaufgaben zunächst erklären und demonstrieren, um anschließend die Möglichkeit zu geben sie selbst umzusetzen.

Alle Beispiele befinden sich auch auf der beiliegenden CD.

4

Aufgabe 1: Wettervorhersage

Die erste Aufgabe dient zum Kennenlernen des Konfigurationsprozesses und den damit verbundenen grundlegenden Konzepten. Bei dieser Aufgabe wird eine Konfigurationsoption hinzugefügt, die es erlaubt die Anzahl der Wettervorhersagen anzupassen.

Folgende Aspekte werden behandelt:

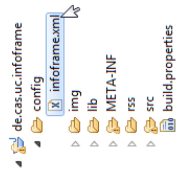
- Einfügen einer vordefinierten Konfigurationsoption
- Auslesen der Konfigurationsoption
- Anlegen und Konfigurieren eines Standard-Validators
- Ändern eines `OnConfigurationChangedListeners`

Teilaufgabe 1.1: Konfigurationsoption anlegen und auslesen

Im ersten Schritt wird hierzu in der Konfigurationsdefinition des Infoframes ein neues `SimpleConfigItem` angelegt. Gehen Sie dazu folgendermaßen vor:

1.1.1

Navigieren Sie im „Project Explorer“ in den Ordner „de.cas.uc.infoframe-config“ und führen Sie einen Doppelklick auf die Datei „infoframe.xml“ aus.



1.1.2

Scrollen Sie in der geöffneten Datei zur Kategorie mit der `id` „weatherPlugin“. Entfernen Sie hier die Kommentarzeichen (`<!--` bzw. `-->`) des `SimpleConfigItems` mit der `id` „forecast“.

5

Abbildung A.9: Konfigurationsaufgaben Seite 4

Abbildung A.10: Konfigurationsaufgaben Seite 5

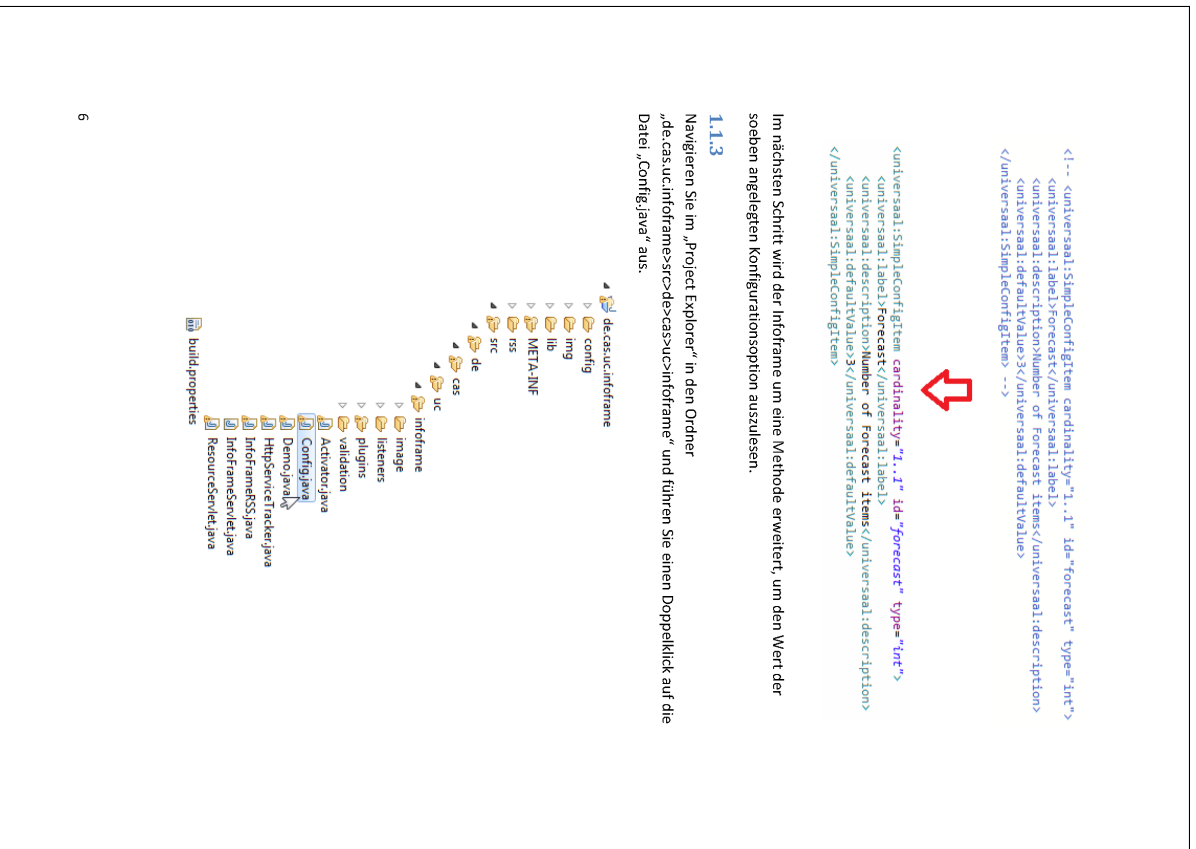


Abbildung A.11: Konfigurationsaufgaben Seite 6



Abbildung A.12: Konfigurationsaufgaben Seite 7

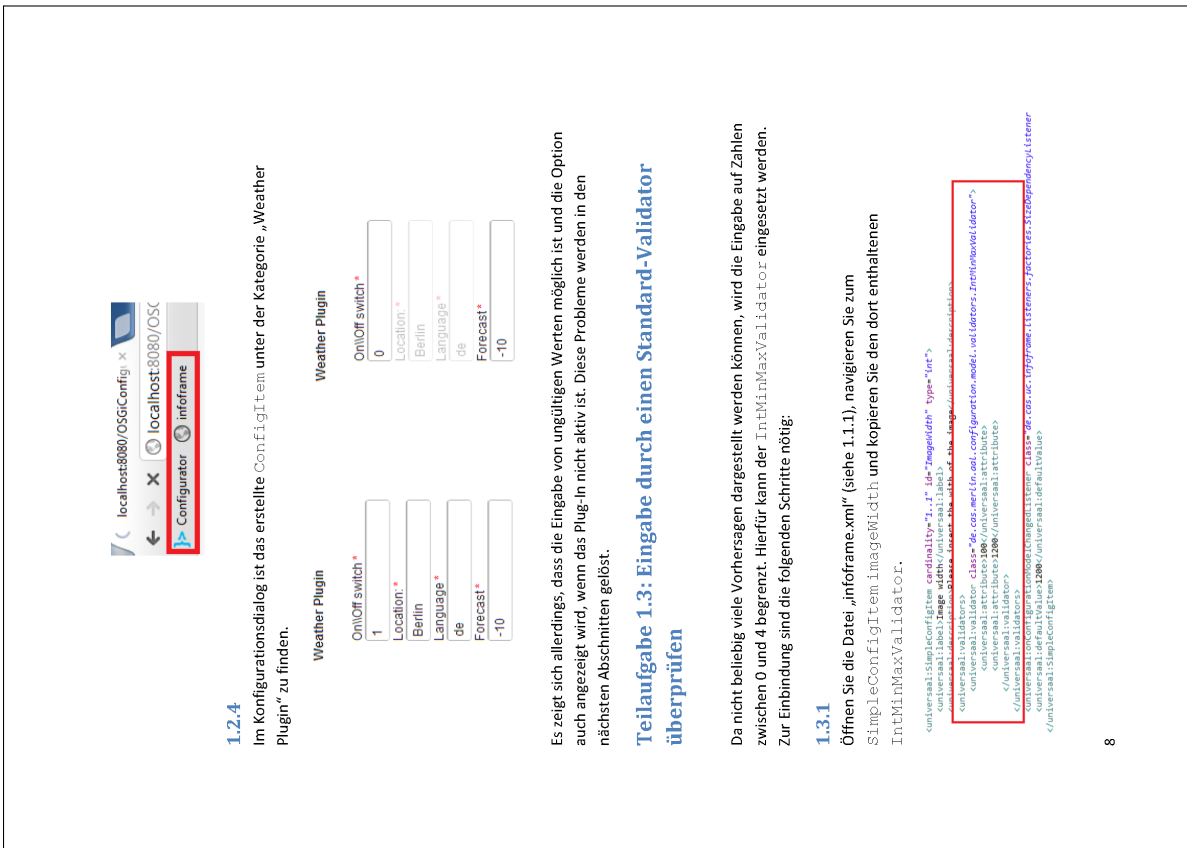


Abbildung A.13: Konfigurationsaufgaben Seite 8

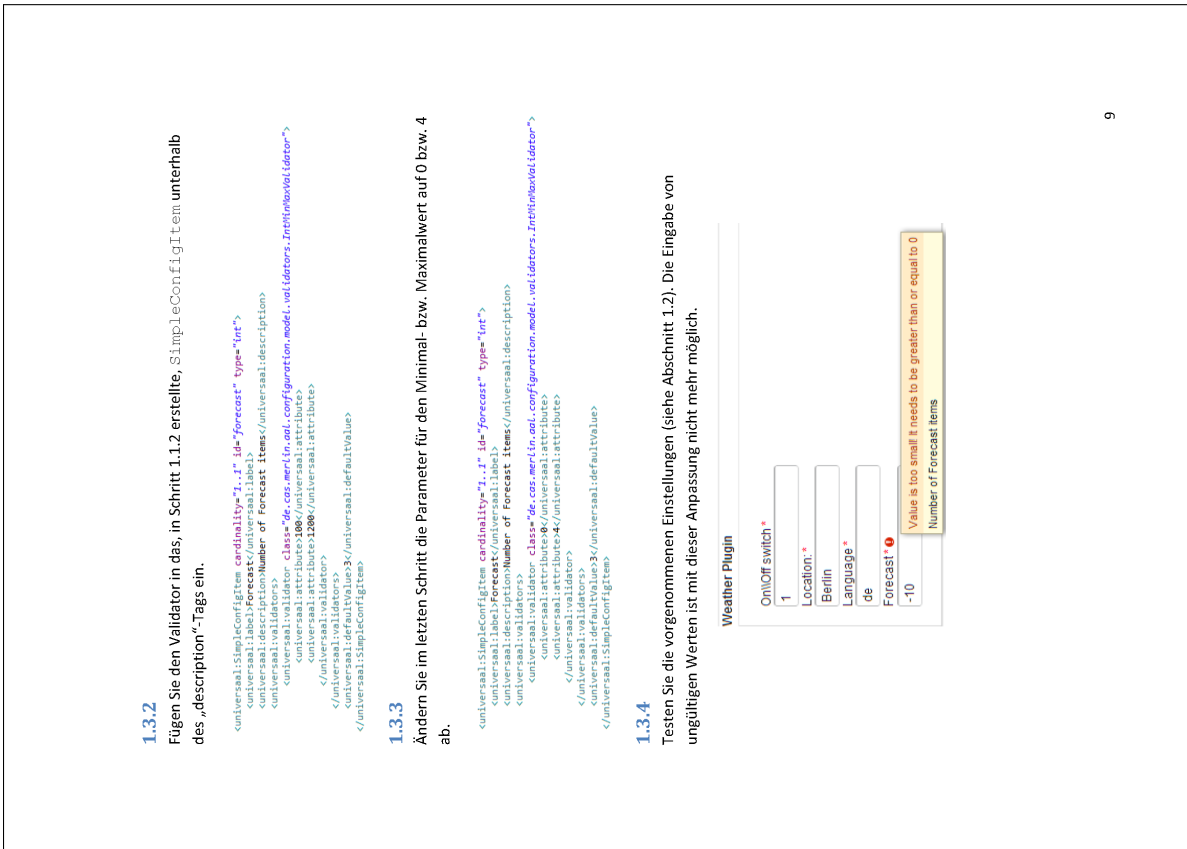


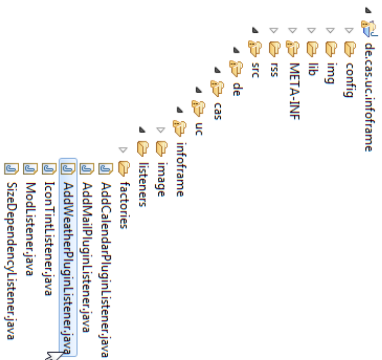
Abbildung A.14: Konfigurationsaufgaben Seite 9

Teilaufgabe 1.4: Konfigurationsoption bei aktiviertem Wetter-Plugin anzeigen

Das Zeigen dieser Konfigurationsoption ist nur sinnvoll wenn das Wetter-Plugin aktiviert ist. Durch Anpassung des `AddWeatherPluginListener.java` kann das gewünschte Verhalten realisiert werden.

1.4.1

Navigieren Sie im „Project Explorer“ in den Ordner „de.cas.uic.informframe“ und führen Sie einen Doppelklick auf die Datei „AddWeatherPluginListener.java“ aus.




1.4.2

Entfernen Sie in der `onConfigurationChanged` Methode die beiden Kommentarmarken („//“).

10

```

@Override
public void configurationChanged(ConfigurationRegistry registry,
    ConfigurationOption option) {
    if (!option.equals(option.getValue())) {
        registry.getConfigurationForId("WeatherStationLocation").setIsActive(true);
        registry.getConfigurationForId("Language").setIsActive(true);
        registry.getConfigurationForId("Forecast").setIsActive(true);
    } else {
        registry.getConfigurationForId("WeatherStationLocation").setIsActive(false);
        registry.getConfigurationForId("Language").setIsActive(false);
        registry.getConfigurationForId("Forecast").setIsActive(false);
    }
}
    
```



```

@Override
public void configurationChanged(ConfigurationRegistry registry,
    ConfigurationOption option) {
    if (!option.equals(option.getValue())) {
        registry.getConfigurationForId("WeatherStationLocation").setIsActive(true);
        registry.getConfigurationForId("Language").setIsActive(true);
        registry.getConfigurationForId("Forecast").setIsActive(true);
    } else {
        registry.getConfigurationForId("WeatherStationLocation").setIsActive(false);
        registry.getConfigurationForId("Language").setIsActive(false);
        registry.getConfigurationForId("Forecast").setIsActive(false);
    }
}
    
```

1.4.3

Testen Sie die vorgenommenen Änderungen (siehe Abschnitt 1.2).

Weather Plugin

ON/Off switch *

Location *

Language *

Forecast *

Weather Plugin

ON/Off switch *

Location *

Language *

Forecast *

11

Abbildung A.15: Konfigurationenaufgaben Seite 10

Abbildung A.16: Konfigurationenaufgaben Seite 11

Abbildung A.18: Konfigurationsaufgaben Seite 13

2.1.1.2 Nehmen Sie nun folgende Anpassungen an dem SimpleConfigItem vor:

Feld	Neuer Wert
Id	FontColor
Type	string
Label	The font color
Description	Please insert the font color
defaultValue	White

Entfernen Sie weiterhin den Validator und Listener aus dem SimpleConfigItem.

Der xml Eintrag sollte nun folgendermaßen aussehen.

```
<universaal:SimpleConfigItem cardinality="1..1" id="FontColor" type="string">
<universaal:label>The font color</universaal:label>
<universaal:description>Please insert the font color.</universaal:description>
<universaal:defaultValue>White</universaal:defaultValue>
</universaal:SimpleConfigItem>
```

2.1.1.3

Erstellen Sie zum Auslesen des Wertes die Methode getFont_color() in der Klasse Config (siehe 1.1.3 und 1.1.4). Beachten Sie, dass diese Methode bereits existiert. Löschen Sie zunächst die Methode oder nehmen Sie die nötigen Änderungen direkt in dieser Methode vor.

```
public Color getFont_color(){
return color.WHITE;
}

public Color getFont_color(){
String c = config.getString("FontColor", "");
Color result = Demo.getColorFromString(c);
return (result == null) ? Color.WHITE : result;
}
```

2.1.1.4

In der Konfiguration ist nun in der Kategorie „Main Parameters“ die neue Konfigurationsoption zu finden.

Aufgabe 2: Schriftfarbe anpassen

Bei dieser Aufgabe wird der Inframe um eine Konfigurationsoption zur Festlegung der Textfarbe erweitert. Diese Aufgabe dient zur Vertiefung der in Aufgabe 1 erlernten Konzepte und Einführung in einige fortgeschrittene Konzepte. Folgende Aspekte werden in dieser Aufgabe behandelt:

- Selbst definierte Konfigurationsoption anlegen und auslesen
- Einen eigenen Validator implementieren und in die Konfigurationsoption einbinden
- Implementierung eines eigenen Listeners und Einbindung in die Konfigurationsoption

Teilaufgabe 2.1: Die Konfigurationsoption anlegen und auslesen

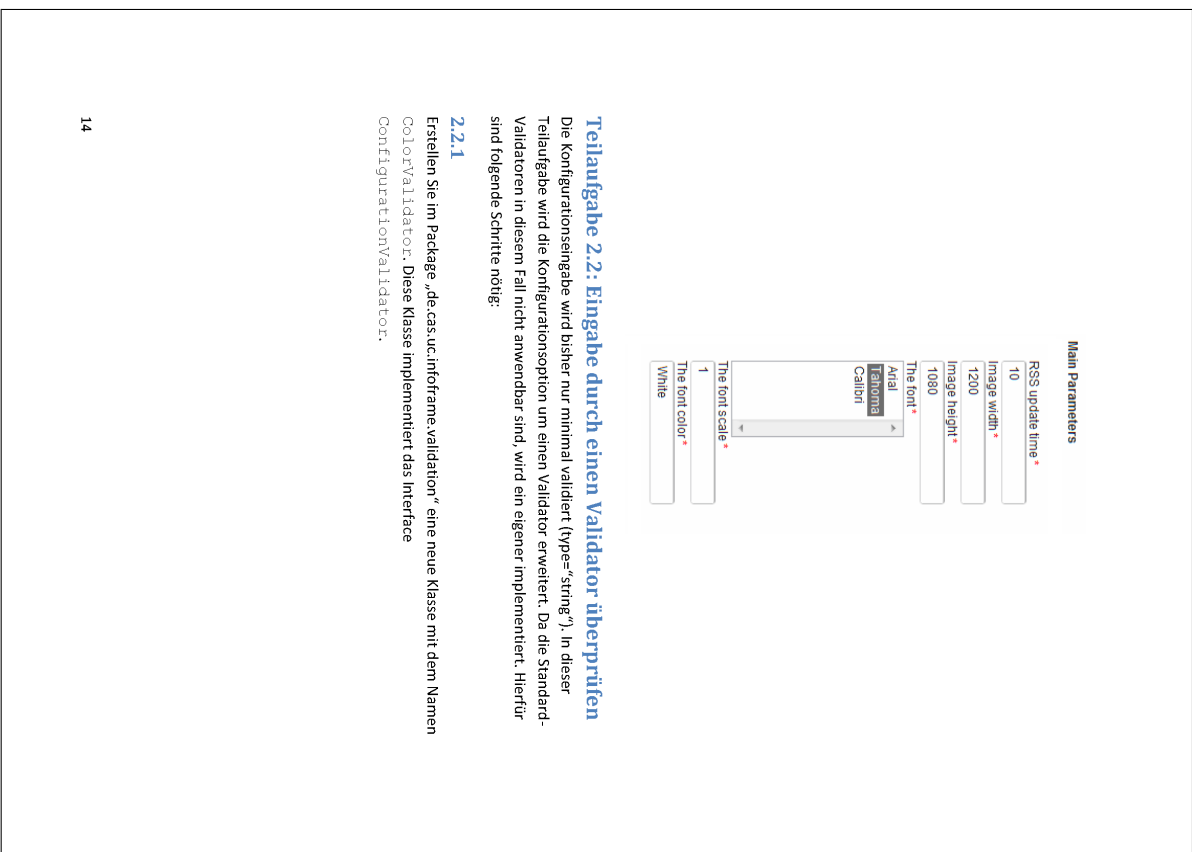
Im ersten Schritt wird ein neues SimpleConfigItem, durch kopieren und ab ändern eines bestehenden SimpleConfigItem, angelegt. Die hierfür nötigen Schritte werden im nächsten Abschnitt näher erläutert.

2.1.1

Öffnen Sie die Datei „inframe.xml“ (siehe 1.1.1), kopieren Sie das SimpleConfigItem „FontScale“ und fügen Sie dieses unmittelbar dahinter ein.

```
<universaal:SimpleConfigItem cardinality="1..1" id="FontScale" type="double">
<universaal:label>The font scale</universaal:label>
<universaal:description>Please insert the font scale.</universaal:description>
<universaal:validators>
<universaal:validator class="de.cos.uc.infi.frame.model.validation.DoubleMinMaxValidator">
<universaal:attribute name="min" value="0.5"/>
<universaal:attribute name="max" value="5.0"/>
</universaal:validator>
</universaal:validators>
<universaal:listener class="de.cos.uc.infi.frame.listeners.FactoryListeners">
<universaal:attribute name="factoryClass" value="de.cos.uc.infi.frame.listeners.FactoryListenersFactory"/>
</universaal:listener>
</universaal:SimpleConfigItem>
<universaal:SimpleConfigItem cardinality="1..1" id="FontColor" type="double">
<universaal:label>The font color</universaal:label>
<universaal:description>Please insert the font scale.</universaal:description>
<universaal:validators>
<universaal:validator class="de.cos.uc.infi.frame.model.validation.DoubleMinMaxValidator">
<universaal:attribute name="min" value="0.5"/>
<universaal:attribute name="max" value="5.0"/>
</universaal:validator>
</universaal:validators>
<universaal:listener class="de.cos.uc.infi.frame.listeners.FactoryListeners">
<universaal:attribute name="factoryClass" value="de.cos.uc.infi.frame.listeners.FactoryListenersFactory"/>
</universaal:listener>
</universaal:SimpleConfigItem>
```

Abbildung A.17: Konfigurationsaufgaben Seite 12



Teilaufgabe 2.2: Eingabe durch einen Validator überprüfen

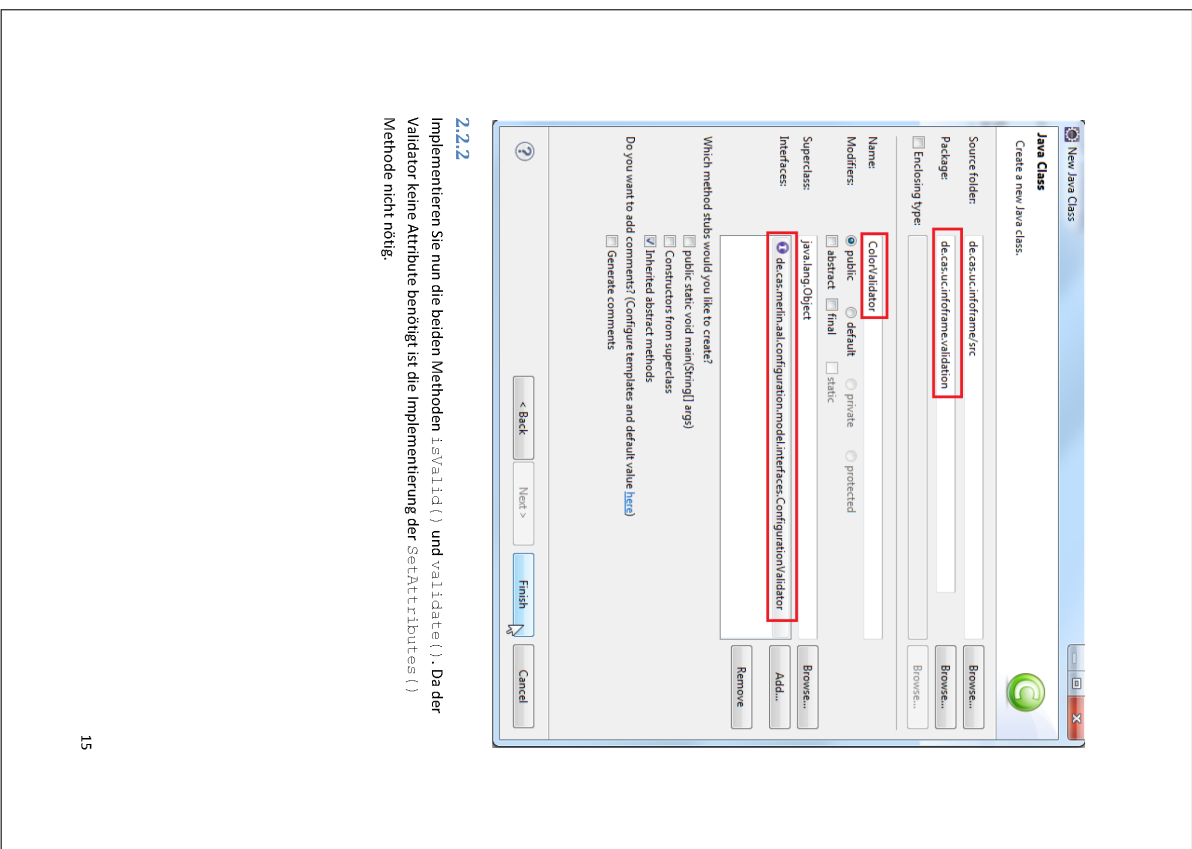
Die Konfigurationseingabe wird bisher nur minimal validiert (type="string"). In dieser Teilaufgabe wird die Konfigurationsoption um einen Validator erweitert. Da die Standard-Validatoren in diesem Fall nicht anwendbar sind, wird ein eigener implementiert. Hierfür sind folgende Schritte nötig:

2.2.1

Erstellen Sie im Package „de.cas.uc.inframe.validation“ eine neue Klasse mit dem Namen `ColorValidator`. Diese Klasse implementiert das Interface `ConfigurationValidator`.

14

Abbildung A.19: Konfigurationsaufgaben Seite 14



2.2.2

Implementieren Sie nun die beiden Methoden `isValid()` und `validate()`. Da der Validator keine Attribute benötigt ist die Implementierung der `setAttributes()` Methode nicht nötig.

15

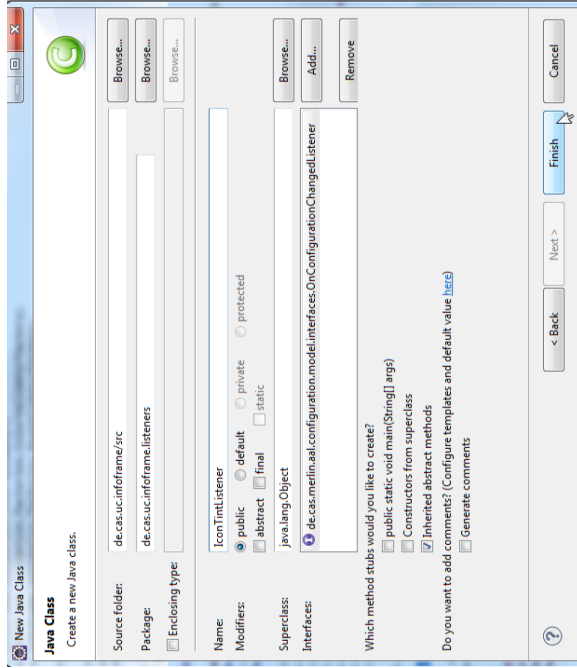
Abbildung A.20: Konfigurationsaufgaben Seite 15

Teilaufgabe 2.3: Implementierung eines Listeners

In der letzten Teilaufgabe werden die Mail-, RSS- und Kalender-Icons passend zur Schriftfarbe eingefärbt. Hierzu implementieren wir einen Listener, der auf Änderungen der Schriftfarbe reagiert.

2.3.1

Erstellen Sie im Package „de.cas.uc.infraframe.listeners“ eine neue Klasse mit dem Namen IconTintColorListener. Diese Klasse implementiert das Interface OnConfigurationChangeListener.



2.3.2

Implementieren Sie im nächsten Schritt die Methode `configurationChanged()`.

Abbildung A.22: Konfigurationsaufgaben Seite 17

```
public class ColorValidator implements ConfigurationValidator {
    @Override
    public boolean isValid(ConfigurationRegistry registry, Value value) {
        Color c = Demo.getColorFromString(value.getValue());
        return c == null ? false : true;
    }

    @Override
    public void validate(ConfigurationRegistry registry, Value value) throws ValidationException {
        if (!isValid(registry, value)) {
            throw new ValidationException("This is not a valid color!");
        }
    }

    @Override
    public void setAttributes(String[] attributes) {
        // nothing to do here
    }
}
```

2.2.3

Bevor der Validator verwendet werden kann muss dieser als OSGI Service registriert werden. Ergänzen Sie dazu den im Screenshot gezeigten Eintrag in der Klasse `Activator`.

```
public void start(BundleContext context) throws Exception {
    createRegistry();
    configMgr = FrameworkUtil.getBundle(this.getClass()).getResource("/config/infraframe.xml");
    ServiceReference<ConfigurationRegistry> reference = context.getServiceReference(ConfigurationRegistry.class);
    ConfigurationRegistry registry = (ConfigurationRegistry) context.getService(reference);
    configMgr.registerConfigurationDefinition(context);
    //register the listeners and validators as osgi services
    context.registerService(AddColorPluginListener.class.getName(), new AddColorPluginListener(), null);
    context.registerService(DeleteColorPluginListener.class.getName(), new DeleteColorPluginListener(), null);
    context.registerService(ResetColorPluginListener.class.getName(), new ResetColorPluginListener(), null);
    context.registerService(AddColorValidator.class.getName(), new AddColorValidator(), null);
    context.registerService(DeleteColorValidator.class.getName(), new DeleteColorValidator(), null);
}
```

2.2.4

Integrieren Sie abschließend den Validator in der „`infraframe.xml`“.

```
<universal:SimpleConfigItem cardinality="1..1" id="FontColor" type="string">
  universal:label:The font color</universal:label>
  universal:description:Please insert the font color.</universal:description>
  universal:defaultValue:white</universal:defaultValue>
  <universal:validator class="de.cas.uc.infraframe.validator.ColorValidator"></universal:validator>
</universal:SimpleConfigItem>
```

2.2.5

Die Konfiguration lässt nun keine ungültigen Farben mehr zu.



Abbildung A.21: Konfigurationsaufgaben Seite 16

```

@Override
public void configurationChanged(ConfigurationRegistry registry,
    ConfigurationOption option) {
    Color color = Demo.getColorFromConfig(option.getValue());
    BufferedImage[] icons = Demo.getIcons();
    Icons=demo.getColorAndSvgImages(icons, color);
}

```

2.3.3

Analog zum Validator (siehe 2.2.3) wird auch der Listener als Service registriert

```
context.registerService(IconIntListener.class.getName(), new IconIntListener(), null);
```

2.3.4

Abschließend wird der Listener in der `FontColor` Option eingebunden, um auf Änderungen dieser Konfigurationsoption reagieren zu können.

```

<universal:SimpleConfigItem cardinality="1..1" id="FontColor" type="string">
  <universal:Label>The font color</universal:Label>
  <universal:description>Please insert the font color</universal:description>
  <universal:validators>
    <universal:validator class="de.cas.uci.infoware.util.darton.ColorValidator"></universal:validator>
    <universal:validator class="de.cas.uci.infoware.validator.ColorValidator"></universal:validator>
  </universal:validators>
  <universal:listener class="de.cas.uci.infoware.listeners.IconIntListener">
    </universal:listener>
  </universal:SimpleConfigItem>

```

18