TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Vienna University of Technology
Institute for Analysis
and Scientific Computing

Master Thesis

iOS Supported Visualisation and Control of a Smart-Home in the Context of AAL – Technologies

Matthias Gasser BSc.
Johann-Blobner-Gasse 8/4, 1120 Wien

Betreuer:	Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker
2. Betreuer:	Univ.Lektor Dipl.-Ing. Dr. Johannes Kropf

Wien, 18.10.2014
(Ort, Datum)

[illegible]
(Unterschrift Matthias Gasser)

Erklärung zur Verfassung der Arbeit

Matthias Gasser BSc.
Johann-Blobner-Gasse 8/4, 1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18.10.2014	[illegible]
(Ort, Datum)	(Unterschrift Matthias Gasser)

Acknowledgements

> “Grant me the strength to accept the things I cannot change, the courage to change the things I can and the wisdom to know the difference.
>
> St. Francis

Foremost, I would like to express my deepest gratitude to my two supervisors, Dr. Felix Breitenecker and Dr. Johannes Kropf for supporting the idea of a mobile AAL application in a new field and all people at the Institute for Analysis and Scientific Computing and Austrian Institute of Technology I have been aquatinted with. Especially Lukas Roedl and all other members of the H.O.M.E.R. middleware software development team.

A special mention and great thanks is dedicated to my beloved strong mother Magdalena who passed away after she bravely fought cancer and to my fiancé Annika and my beloved dad Norbert, all of whom I owe to, to be the person who I am today. Also to my sister and her husband, Ilona and Florian which supported me relentless on (Bio)-Chemical topics throughout my master studies. Many thanks to my grandparents Tate, Nune, Vota, Mame, who laid to rest while writing this thesis and I'll never forget.

Last but not least a big thank you to all of my fabulous friends and many other people I am unfortunately unable to name all.

“The traveler was active, he went strenuously in search of people, of adventure, of experience. The tourist is passive; he expects interesting things to happen to him.

Daniel Boorstin

Abstract

There is a distinct trend towards an aged society living in single person households. This creates an increasing demand for new support, guidance and assistance services especially for the elderly. Ambient Assisted Living (AAL) copes with this social trend. The sector of tablet devices also gained lately a major spark of interest. This thesis elaborates the synergy effects of AAL and mobile tablet devices. Current solutions are available for standard PC tablet hardware. In the course of this thesis a novelty AAL mobile application – to control and visualise a smart home – is being developed for an iOS iPad tablet device. The design process is thereby platform agnostic and besides implementation details similar to non iOS devices.
A simple model to analyse the daily activities of the inhabitant is shown by an example of benchmark sensor data.
The requirements of the software are deducted by analysis of existing solutions and by the elaboration of the AAL environment and the technical environment, especially in terms of the mobile connectivity and communication interface.
Based on this requirements is the software design steps and implementation work discussed in detail.
The product of this – a new mobile application – may be the foundation for further developments and adoption to different AAL middleware systems.

Abstract

Es gibt einen klaren Trend einer alternden Gesellschaft, welche oft in Ein-Personen-Haushalten lebt. Dies schürt einen stark wachsenden Bedarf für neue Hilfs-, Unterstützung und Assistenzsysteme. Das Gebiet der Altersgerechte Assistenzsysteme für ein selbstbestimmtes Leben – abgekürzt mit AAL, Engl. Ambient Assisted Living – versucht diese notwendige Unterstützung zu bieten. Gleichzeitig erfährt der mobile Tablet Geräte Sektor eine hohe Nachfrage. Diese Arbeit versucht mögliche Synergie Effekte zwischen AAL und den neuen mobilen Tablets auszuarbeiten. Derzeitige Lösungen basieren noch auf PC Tablet Hardware.

Im Zuge dieser Arbeit wird eine Neuartige AAL Anwendung für ein iOS basierendes mobiles Tablet entwickelt, um ein Smart Home zu steuern und zu visualisieren. Dabei ist der Design Prozess möglichst unabhängig von der verwendeten Hardware durchgeführt und sollte somit auch für andere Plattformen geeignet sein.

Ein Model zur Analyse der Aktivitäten eines typischen Tages wird anhand von Beispiel Daten gezeigt.

Die Anforderungen an die Software wurden aus der Analyse von bestehenden Lösungen und durch Aufarbeitung der AAL Umgebung, sowie des technischen Umfelds, insbesondere der mobilen Konnektivität und Kommunikationsschnittstelle abgeleitet.

Basierend auf diesen Anforderungen werden die Software Design Schritte und Implementierung im Detail herausgearbeitet.

Das Endergebnis dieses Prozesses – eine neue Mobile Application – kann möglicherweise die Grundlage für weitere Entwicklungen und Anpassung an weitere AAL Systeme dienen.

Contents

Part I.

Introduction

1. Introduction

This thesis joins the fields of Ambient Assisted Living with the trend of modern tablet devices. The term Ambient Assisted Living covers a broad spectrum of applications governed by the aim to sustain independency of people even at grown ages. A decent definition of the term AAL is given by Steg et al.:

> "AAL aims to prolong the time people can live in decent way in their own home by increasing their autonomy and self-confidence, the discharge of activities of daily living, to monitor and care for the elderly or ill person, to enhance the security and to save resources."[1]

The main focus of this work lies on evaluating the environment of AAL with the new possibility of highly capable and mobile tablet devices. An example design and implementation process of a graphical user interface on a current tablet will be introduced. This application will be bound to an existing AAL middleware which allows remote and onsite monitoring, as well as simple emergency alarm capabilities but under a reusable and middleware agnostic point of view.
This thesis is divided into three parts, firstly the introduction, guiding the reader to the topics of AAL, secondly the background, which addresses current state-of-the-art solutions and related work. It also discusses the requirements for the software and hardware, in respect to the analysation of the AAL field. Afterwards, to be able to make design decisions, the environment of the requirements is being assessed. The implementation part covers the design decisions and the methods, which are used for implementation and are also ultimately implemented. After all the lessons learned are discussed and an outlook is given.

1.1. Motivation

There is a distinct trend towards an aged society living in single person households. A demographic change, often depicted as ageing society, especially impacts the industrialised countries, concludes to the fact that the populations average age will increase sustainably. For example the world population of the 60-year and older group gained from 99 million in 1950 to 248 million in 2000 and is expected to be 298 million by the year 2050. The 50 and older population from 2000 to 2050 will grow by 68 times faster than the rate of growth for the total population.[2]

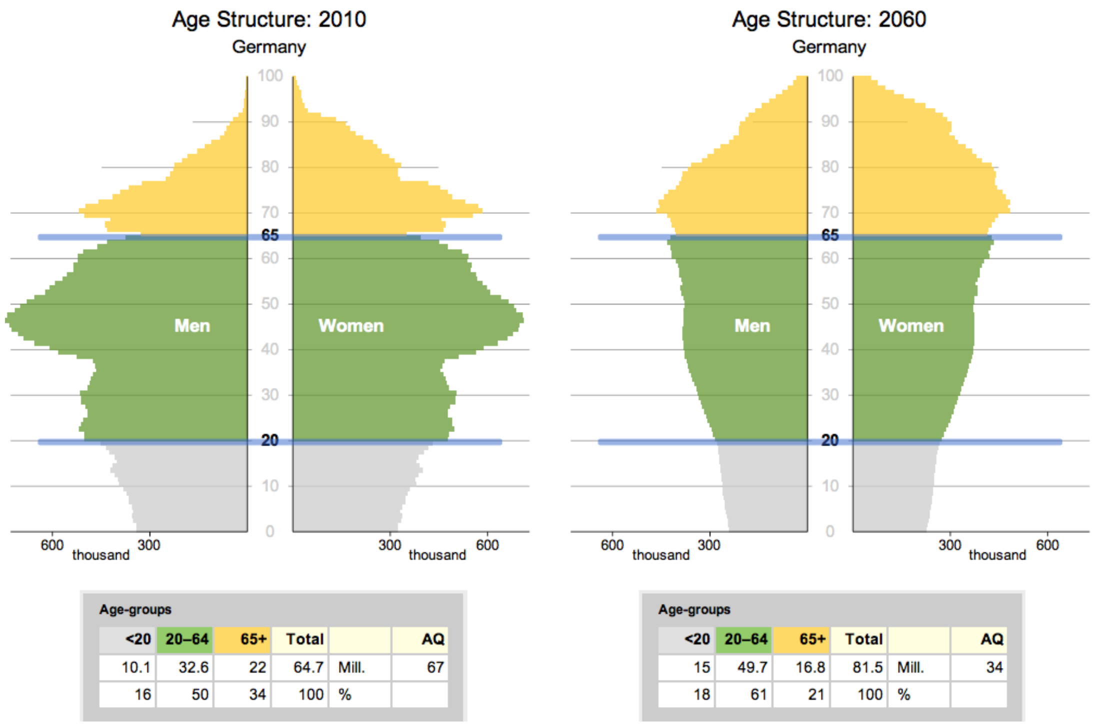
1.1.1. Social Environment - Life Situation of the Elderly

1.1.1.1. Demographic Development

Demographic development depends obviously on the area observed. As of the end of 2011 the world population crossed a new milestone of estimated 7 billion inhabitants. Europe's share of this population is about 12%, whereby there is an adverse trend to drop to 9% by 2050. There is a clear diversity of demographic trends within Europe, having a population decrease in Germany and Russia, while France and the United Kingdom are experiencing a boom in population due to migration. The European Council prognoses a triplication of the population over the age of 60 by the year 2050. Which *will be over one third of Europe's population[3]. The German DESTATIS Statistisches Bundesamt* published a projection of the age distribution for Germany based on the following assumptions:[4]

- Nearly constant birth rate at 1.4 children per woman
- Life expectancy of newborn children in 2060 85 years for boys and 89.2 years for girls
- Annual net migration +100 000 persons

Figure 1.1 shows exemplarily the over-ageing of the german society by comparing the age distribution of 2010 versus the projection of 2060. This emphasises a strong need of technologies to support the needs of the elderly to maintain a secure and self-dependent life on their own.



Age-groups (2010)

<20	20–64	65+	Total		AQ
10.1	32.6	22	64.7	Mill.	67
16	50	34	100	%	

Age-groups (2060)

<20	20–64	65+	Total		AQ
15	49.7	16.8	81.5	Mill.	34
18	61	21	100	%	

Figure 1.1.: Age structure Germany 2010 vs 2060[4]

1.1.1.2. Circumstances of the Elderly

The ageing is a complicated process driven by physical, psychical influences, environmental, social and institutional surroundings[5]. It is very important to consolidate the physical and social resistance to sustain the decision-making and responsibility on a satisfying level. However, this goal is hard to achieve, due to recession of cognitive performance and neuronal perception, influencing the commemoration of the elderly.

The portion of single households are increasing with a progressive age. There is a strong relationship between the type of housing, the gender and the likelihood of the need for assistance. Living alone draws many accumulating problems like isolation and loneliness whereby there is a strong need for social contacts and care. The image of the elderly has changed over the past years, today the aged feel more rejuvenated and todays 70-year-old have different capabilities than 70-year-old 40 years ago. Gerontological science shows that the loss of abilities or human frailty is not linked to a specific age, but more depended of previous lifestyle, economic status and educational background.[6]

1.1.1.3. Health and Care of the Elderly

Health of the Elderly is by the definition of Kruse[7, Page 513] a multidimensional construct which is given by the following dimensions:

- lack of diseases or symptoms
- an optimal functional state
- an active, autonomous, personal satisfying way of life
- a successful outcome of burdens
- an individually tailored system of medical, care and social support

Obviously the topic discussed in this thesis has an influence on most items of this construct. By sustaining the independence through assistive methods in an more active, autonomous and personal satisfying way of life can be achieved. Still said there is no silver bullet solution yet, however, there are many promising projects currently in development, which will be discussed in section 2.1.

1.1.1.4. Functional Groups

Recently a group of professionals in the United States introduced an alternative way of defining functional groups to classify persons. This is an easier approach in comparison to the previously stated multidimensional construct of Kruse. This definition brings the classification of general health situation of the elderlies in line with their mobility capabilities[8].

Go Go's	The persons are able to conduct the daily activities, either under their own power, or by using an aid like a walking stick etc.
Slow Go	The persons are still able to be on their own, but at a slower pace, usually accompanied by supporting aids
No Go	The persons needs perpetually assistance by a caregiver

This work focuses mainly on the *Go Go's* and *Slow Go's* because these groups are suspected to live in an AAL supported home in the future. *No Go's* are still tied to a care around the clock care provider, who may also take advantage of a tablet AAL supported environment for monitoring purposes.

1.1.2. Ambient Assisted Living

This aged society drives an increasing demand for new support, guidance and assistance services especially for the group of elderly. Ambient Assisted Living (AAL)[9] copes with this social trend, by embracing technologies which provide various functionality to support people in carrying out daily activities and to provide monitoring for medical reasons. Some important goals of AAL are[10]:

Comfort	ease the compliance of daily activities
Safety / Security	securing of the dwelling
Health	provide health related functions like emergency detection and prevention, treatment monitoring

These goals demand several key components for an AAL system. *Safety/Security* and *Comfort* require mostly a simple set of sensors like motion detection, temperature sensor and actuators like a software controlled power switch. The *Health* area on the other hand requires rather more medical oriented sensor sets like heart rate sensors, fall detection, body temperature, blood pressure etc.
The *AAL Middleware* provides the computational background to receive sensor events, send control messages to the actuators, provide access for a various type

of interfaces like computers, web-services, tablets, phones, dedicated devices. For the scope of this thesis we will focus on the tablets as a user interface, whereby the reader will be first introduced into current existing solutions based on PC-type tablets in section 2.1 and later on a new software implementation on a mobile tablet like the iPad will be shown.

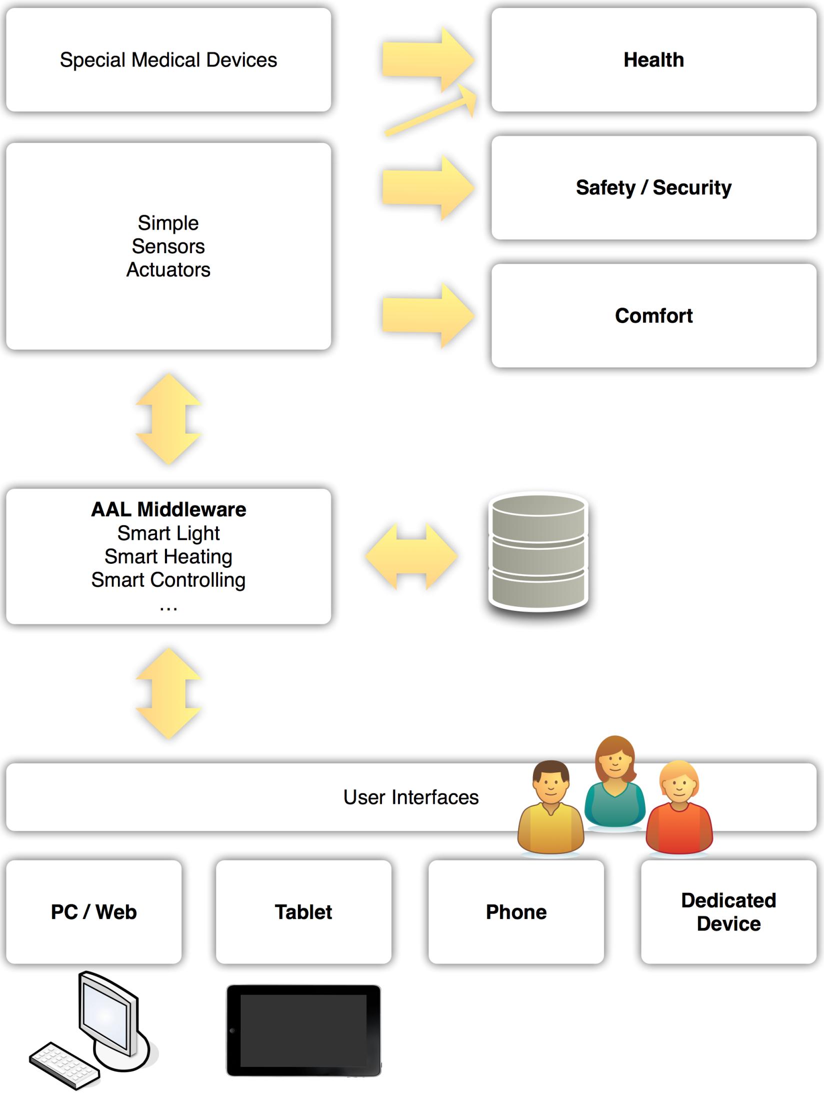


Figure 1.2.: Structure of a typical AAL Setup

Figure 1.2 sketches the structure of AAL solutions in context with these goals. By the nature of complexity of the human beings, AAL applications tend to be cost expensive and cheaper generic solutions have been rare.

1.2. This Work

1.2.1. The Practical Aspect of Tablet Supported AAL

Evidently, various mobile tablet devices gained high acceptance nowadays. The development of those is backed by a wide range of already available and upcoming applications, whereby Ambient Assisted Living is already covered in a more consumer driven niche, like controlling and monitoring the home entertainment system, temperature etc.[11]. However, the development of more clinical orientated applications, gained a great spark of interest lately. Modern tablet devices feature high computational power on even more than one core, a very fast flash storage and an easy to use software development environment which is very similar to personal computers, due to its mostly analogous architecture, driven by the same coding languages.

1.2.2. Tablet Platform Decision

Currently there are two major options to choose from when it comes to tablet device development. Either the Android platform, based on Java audited by Google or Apple with their UNIX[1] based iOS operating system. Both platforms have many things in common, but there are also some major differences. Unlike Android is the Apple App ecosystem a closed one, which means Apps need to be in line with the guidelines and to be approved one by one by Apple. This process is described in short in 4.1.2. For this thesis the author has decided to stick with Apple iOS but for no particular reason as the decision for a specific platform is not tailored by the functionality or any other means of both platforms. The same interface may be implemented on an Android based tablet device.

[1]Multitasking / Multiuser computer operating system, existing in many derivates

This thesis will highlight existing tablet solutions and the development of a new Apple iOS based user interface for an existing AAL middleware H.O.M.E.R., however, most conceptional parts are also true for Android based tablets and smart phones.

1.2.3. Use-Cases

Governed by the high diversity of AAL, many possible use-cases are imaginable in a tablet context, which would exceed the scope of this work. Most of the use-cases may require a basic user interface and communication to an AAL middleware, which will be covered in the next chapters. The following two use-cases are providing the foundation for the control of the smart home as well as a medical application.

1.2.3.1. Visualisation of Sensor Data and Control

Visualisation of all sensor data gives an overview of what is happening in the smart home environment. This allows to examine device power usage versus generated power or to inspect temperature over time, air quality etc.
Controlling the actuators, like switches and dimmers delivers a simple remote control use-case.

1.2.3.2. Tablet supported activity monitoring

A more challenging task is to provide activity monitoring. Activity monitoring allows to track the inhabitants movement profile and to alarm in case of possible emergency. For the scope of this thesis the reader will be introduced into activity recognition patterns in a theoretical way.

Part II.

Background

2. State of the Art

The healthcare segment of smartphones and tablets called m-health, being a subset of e-health gained recently a great spike of interest. By 2011 the total number of health care apps available on the Apple AppStore had been 9.000 already, by the summer 2012 the count is predicted to rise to 13.000 showing a steady increase. The Apple AppStore claims even higher numbers, but many of those are actually not health related[12][13].

2.1. Ambient Assisted Living

2010 the European Commission started the project *UniversAAL open platform and reference Specification for Ambient Assisted Living* which tries to develop a united AAL ecosystem and an open platform by evaluating current existing and developing AAL solutions. The main objectives of *UniversAAL* are the following:

- To make it technically feasible and economically viable to conceive, design and deploy innovative new AAL services.[14]
- Produce a platform providing the necessary technical support, and acting as an open, common basis for both developers and end-users.[14]
- Carry out support activities promoting widespread acceptance and adoption of the platform. These activities form an integral part of the project and will start at an early stage.[14]

Their vision is that someday consuming AAL services will be as simple for the end user as downloading and installing software applications nowadays.[14]

There are several background projects involved in the consolidation process. See figure 2.1.

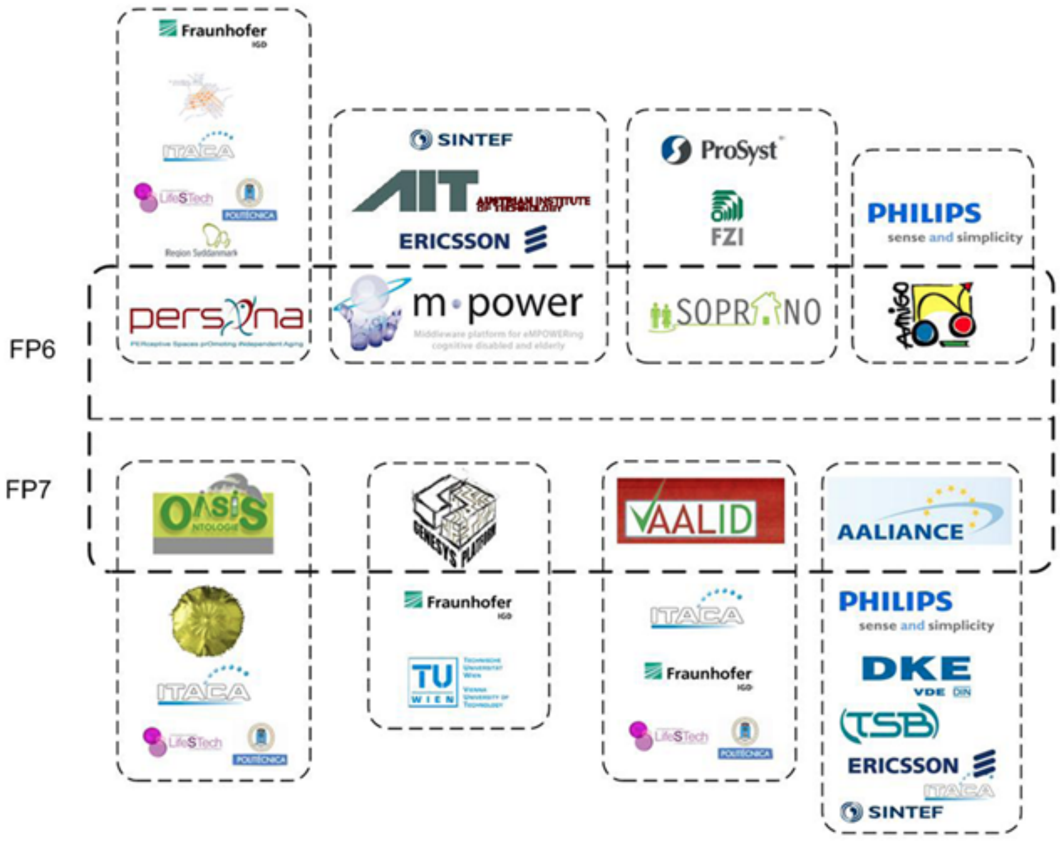


Figure 2.1.: Participation of UniversAAL partners in reference initiatives in AAL Reference Architectures[14] (FP6, FP7 Sixth and Seventh Framework Program)

The UniversAAL platform will contain ideas and code of the current state-of-the-art results and new development based upon experience of the already build AAL applications. This platform shall be pillared by a vital community. The platform will actually implement the reference architecture to provide runtime support on various execution environments[14].

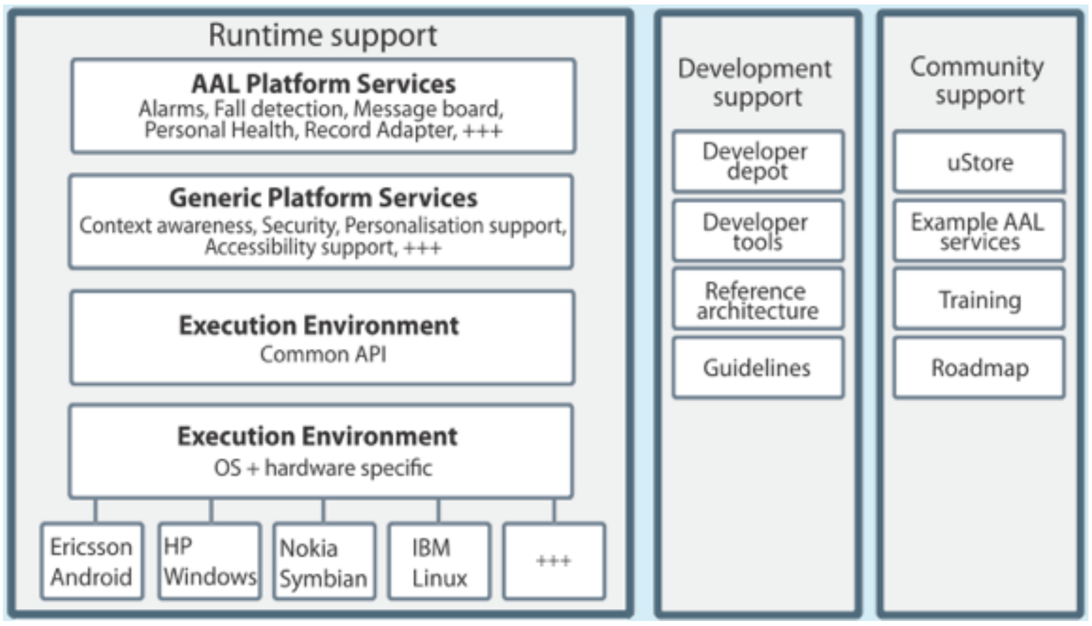


Figure 2.2.: UniversAAL Platform Schematic Structure[14]

The UniversAAL project is member and founder of the *AALOA AAL Open Association* project. Their mission is to join the people, resources and tools involved in AAL in a single forum, to achieve AAL progress[15]. They released a manifesto which highlights the current issues of AAL, like fragmentation of development efforts and why there has been no truly AAL breakthrough yet. The interested reader may find the manifesto at `http://aaloa.org/manifesto`.

2.2. AAL Middleware

There are many different approaches regarding Ambient Assisted Living integration, justified by a vast number of possible use cases, which range from a simple drug taking reminder to a highly sensor equipped dwelling, monitoring people's activity and alarming any dangerous situation even to a remote caregiver. A complete listing would be a quite fanciful task and is not subject of this thesis, but still the table 2.1 depicts an overview of various AAL middleware platforms, mostly funded by the European Commission over the last years.

Name	**Focus**	**Ref**
H.O.M.E.R HOMe Event Recognition System	OSGi based Sensor data mapping Finite State Machine	[16]
MPower project	based on Service Oriented Architecture	[17]
CompanionAble	Blackboard via Simple Object Access Protocol graceful, scalable, cost-effective integration	[18]
Soprano	OSGi based sensor / actuator ontology context ontology	[19]
OASIS	OSGi based use of ontology at all levels	[20]
PERSONA	OSGi based Service/Platform based on ontology	[21]
UniversAAL	European Union Project intents to consolidate current results promote open platforms	[21]

Table 2.1.: Not exhausting overview of AAL middleware platforms

This list is not exhausting. However, there is lately a great effort made by the UniversAAL project[14], to standardise the AAL middleware by making it technical feasible and economically viable to conceive and deploy innovative new AAL services.

2.3. Challenges

The reader was previously introduced into existing state of the art solutions to fulfil the needs of the elderly people sustaining a safe and independent lifestyle. Every project tries to tackle the following challenges in their own way, according to their focus. Due to the high diverse environment AAL is applied to, different challenges arise depending on the field of application. The following issues give a non exhaustive view of possible challenges in the AAL field of work.[22]

2.3.1. Development and Distribution

Albeit the UniversAAL project has some progress in distributing AAL services to the people, distribution is still an important challenge to tackle for the next generation of AAL systems. The research and developments efforts are fragmented. The efforts of unifying are slow but steady. The market suffers from a lack of technical convergence, due to different technical implementations which are not easily inter-comparable.[23][15]

2.3.2. Social Independence

The AAL country report of Finland comes to the conclusion

> "that assistive devices are not useful if not combined with services and formal or informal support and help."[24]

The *Promises and Challenges of Ambient Assisted Living Systems*[25] and the author of this thesis share this opinion. Whilst AAL devices are able to sustain the independence, they tend also to lower the rate of social connection to other people. This may lead to a stage where the elderlies in fact are surviving without actually living anymore. Surely there are counter measures like the project COPLINTHO [26], which tries to establish a network between the patient and family, friends and any other caregiver. However, this project is tailored for the recovery process of a patient, and not generally of use. It is essential for any AAL project to cover both aspects, the technical and the social aspect.[25]

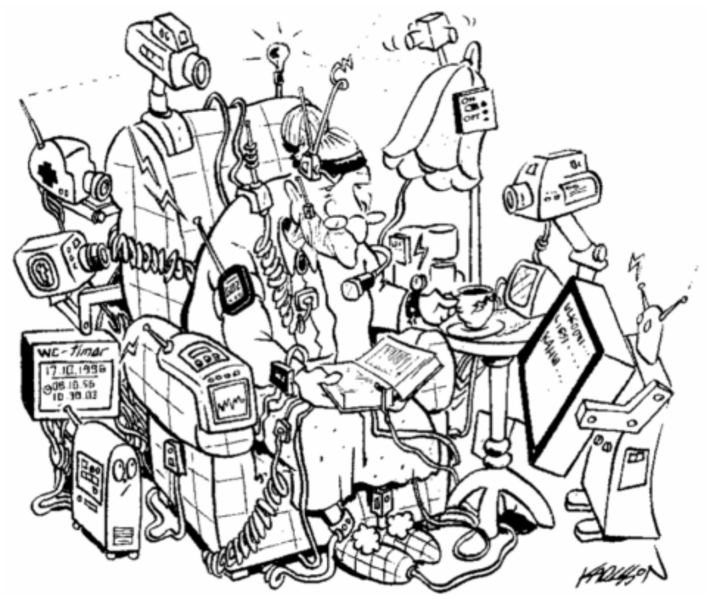


Figure 2.3.: Overusing Assistive Devices[27]

2.3.3. Adaption - People's Willingness

The *Promises and Challenges of Ambient Assisted Living Systems* by Sun et al.[25] analyses the people's unwillingness adapting to AAL systems, which is reasoned by two main issues, the psychological and technological frustration.

Psychological Frustration

Due to the advancing age, some people tend to become frustrated because of their loss of physical strength. Many AAL systems see their main user as weak and passively assisted by their people around, but this has also some psychological side effect of losing their self-esteem.[25]

Technological Frustration

Most older people are often scared by the fact of adapting to new technology, due to various reasons, like the fear of breaking things or the feeling of using it wrong – and other reasons. This leads to a major hurdle in developing and designing a user interface experience, which integrates smoothly into this environment.[25]
Another important issue is being discussed in the work *Attitudes and Require-*

ments of Elderly People Towards Assisted Living Solutions[28] J. Grauel, A. Spellerberg. They questioned 383 over 60-year-old for their attitude towards technology and home automation devices, as well as their financial willingness paying for such an installation. In contrary to the Sun et al. report, the questioned people are surprisingly open minded towards new technology but this may be the result of an "AAL solutions are good for the others, but not for my home" mentality. The report yields that about one third of all participating seniors are willing and able to pay an amount of only 30 € per month for the AAL installation, which on the other hand means that the costs for installation and operation of an AAL system must be in line with the finical capabilities of the elderly.[28]

2.4. Related Work

Most iOS smart home applications are tailored for home entertaining use, like LOXONE[1], iHome SYSTEMS[2], SAVANT[3] and many more. Many project groups discover the recent smartphones and tablets as a fitting user interface device like the *Real-time Energy Monitoring and Controlling System based on ZigBee Sensor Networks* by W.H. Kim et al.[29]. They build a network of ZigBee[4] linked sensors which is to be controlled and the accumulated data to be viewed by a web application and a native Apple iPad App.
Another M. Brinkmann et al. published *Concept and Design of an AAL home Monitoring System Based on a Personal Computerized Assistive Unit* [30]. In their paper they present an ambient assisted living approach based on a standard automation hardware, whilst – a PC tablet based – **P**ersonal **A**ssistant **U**nit for **L**iving "**PAUL**" takes a key role in the project. It provides a human computer interface designed for intuitive interaction and ease of use for the elderly.

[1]`http://www.loxone.com`
[2]`http://www.home.asia`
[3]`http://www.savantsystems.com`
[4]Specification by the ZigBee Alliance (`http://www.zigbee.org/`) for a suite of high level communication protocols, using small footprint devices.

3. Requirements

There are requirements on both the software and hardware side. This chapter will first evaluate the software requirements for the back-end and front-end side and elaborates the choice of hardware. For this project the overall requirements are a modular plugin for an existing Ambient Assisted Living framework, which implements a server accessed over a defined API[1] to communicate with a tablet driven user interface in terms of visualisation of a flat, including sensors, actuators, energy data, and allows a simple control of such actuators, mainly not for the inhabitants of the smart home, but for relatives and caregiver as a tool to check on their caretaker.

3.1. Software Requirements

The IEEE Standard Glossary of Software Engineering Terminology (1990)[31, p.62] defines a requirement as

1. A condition or capability needed by a user to solve a problem or achieve an objective.

2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

3. A documented representation of a condition or capability as in (1) or (2).

[1]Application Programming Interface

Given the state of the art projects, the author deducted the following functional and nonfunctional requirements for a tablet driven user interface for the H.O.M.E.R. project[2].

3.1.1. Functional Requirements

Functional requirements are also called behavioural requirements, often defined as “shall – statements”, these specify the software functionality that the final product must deliver.[32]

3.1.1.1. Integration into an existing AAL Framework

The user interface on the tablet device shall be, as easy as possible, portable to any other AAL framework by a developer. This shall be achieved by a decoupled back/front-end design, utilising a defined and simple API in between. Separating the back-end and front-end component by this API will also introduce the possibility of using other user interfaces natures as long as they are capable of serving the defined API.

3.1.1.2. Management of more than one Flat

Especially for the use case of an elderly care giving institution in the context of ambient assisted living, the caregivers shall be able to track more than one flat by the app on the tablet device. This will allow the connection to distinct remote smart homes within one app. The result of the current activity state for each flat shall be shown in an overview to provide early access to any fine distinction of the caretakers activity of his daily activity profile.

3.1.1.3. Visualisation of the Flat

To achieve a low initial setup burden, the flat shall be drawn, on the tablet, out of the geometric data information stored in the back-end. Sensors and actuators

[2]Home Event Recognition System is developed by AIT Austrian Institute of Technology GmbH. `http://homer.aaloa.org`

shall be visible and filterable by a favourite mark. This will ease the real estate in a smart home with a very high sensor density. The flat map shall also be multitouch accessible, which means, it is able to zoom in and out by the use of the pinch and pan gesture, this grants a very natural user interface interaction for the end user.

3.1.1.4. Visualisation of Sensor Values / Energy Balance

The Visualisation of the historic sensor values shall be implemented using a simple interactive graph representation. The current energy balance shall also be visible on the top to keep track of the energy consumption / generation for the smart home appliance.

3.1.1.5. Control of Actuators

It shall be possible to actively control the given actuators by a simple touch on the actuator to dim the light or to close the shutter.

3.1.1.6. Activity Recognition

The activity recognition shall compute an activity profile by – user triggered – sensor events over time. This data shall further be used to estimate a normal day activity profile, which is then used to track any differences in daily life to alarm in case of significant deviation of the normal day profile. This shall be achieved without any major configuration setup process or annotated data efforts.

3.1.1.7. Alarm Notification

Given the activity recognition data, in case of abnormal activity profile, a visual alarm shall be triggered to the caregivers by a notification shown on the iOS device.

3.1.2. Nonfunctional Requirements

Nonfunctional requirements are also referred to as constraints[33], goals[34] and quality attributes[35] in the literature, likewise operational costs, reliability, maintainability, performance, etc. are also mentioned as nonfunctional requirements[36]. There is, as time of writing, no formal definition or any complete list of nonfunctional requirements, as every project may endorse a different set of nonfunctional items. However, it is essential to identify these, as they play an important role for every ongoing design decision. John Mylopoulos et al. [36] developed a process oriented approach to represent and use nonfunctional requirements by an example of an expense management system.

3.1.2.1. Modularity

As this should bring the foundation to an iOS driven AAL framework user interface, which shall be easy to adopt to any AAL middleware, modularity becomes of high importance. Every middleware is likely to provide a slight different domain model scheme, so the interface to the given middleware must be highly adoptive.

3.1.2.2. Performance

Performance is a key factor in the mobile device sphere. The user is used to any instant response or at least valuable visual feedback of any progress. However, most actions should feel instant and non blocking.

3.1.2.3. Power Consumption

Analog to performance is the battery-life of high importance to every mobile application due to limited power resources. The application should not draw unnecessary power of the tablet by performing avoidable computational tasks. Any high computational task should be translocated to the server back-end where power is less an issue than mobile.

3.2. Hardware Requirements

To achieve the previous stated software requirements, complimentary hardware parts are necessary. The back-end side requires an AAL middleware capable PC or alike. Various AAL middleware platforms are previously discussed in chapter 4.2. Most AAL middleware softwares run on a off-the-shelf PC, so the hardware requirements will focus merely on the tablet aspect. The following sections will focus on state of the art techniques and defaults for tablets and tries to encipher the minimum requirements for a tablet.

3.2.1. Mobile Connectivity Standards

There are 3 major wireless interfaces available for mobile devices today. The well known Wi-Fi[3] and the GRPS / EDGE / UMTS / LTE mobile telecommunications technology standard, as well as Bluetooth 2.1 and Bluetooth 4.0 technology. Bluetooth however, is more targeted to the audio or sensor data transmission and is not of high importance for this project at this stage. However, the tablet may connect to various Bluetooth sensors acting as sensor gateway in the future. The Wi-Fi name was introduced to have a marketable brand instead of the originated IEEE standard: 802.11b Direct Sequence. The 802.11 technology has its roots in 1985 as the US Federal Communication Commission released the ISM band[4] for unlicensed use. Since then several 802.11 protocols had been invented named: 'a,b,g,n,ac'. The 'a' protocol used the 5 GHz and 3.7 GHz band, 'b' and 'g' operate in the 2.4 GHz frequency band, the 'n' in both the 2.4 GHz and in the 5 GHz band. The latest 'ac', which is by the time of writing still a DRAFT protocol operates only in the 5 GHz band. Data rate per stream has dramatically increased over time, ranging from theoretical 1 Mbit/s to 866.7 MBit/s. The maximum transmission range depends on local structures e.g. walls as well on the devices used. The common values are in the range of 35-70 meters indoor and 140 to 250 meters outdoor and free sight.[37][38]

[3]**Wi**reless **Fi**delity - wireless network connectivity
[4]industrial, scientific and medical radio bands

While Wi-Fi finds its application more in the home, office and particular public area, the mobile telecommunications technologies are used to bridge the link on much further distances. The first mobile telecommunication generation introduced 1980 based solely on analog telecommunications standards, whilst the later on the second generation utilises also digital standards, whereby the second generation focus lies on voice transmission, but it is also capable of slow data transmissions. The data performance is being improved over time in the so called 2.5G generation, *General Packet Radio Services* abbreviated GPRS and the 2.75G generation called *Enhanced Data rates for GSM Evolution* abbreviated EDGE. But both the GPRS and EDGE data transfer rates (< 0.22 Mbit/s) are considerable slower compared to local area networks which have a theoretical bandwidth of 100 MBit/s. The third generation offering higher data speeds, paved the way for a bag of various mobile applications. The standard however, does not specify a minimum or average data speed, but rates of 42 Mbit/s are possible. The latest member of mobile telecommunication standards is Long Term Evolution LTE which delivers higher theoretical download 299.6 Mbit/s, 75.4 Mbit/s upload rates. The standard is being developed by the 3rd Generation Partnership Project 3GPP and is about to be deployed in several countries all over the world. However, the frequencies used for LTE are often different per country, which leads to the fact that a LTE devices will probably not work in every LTE network worldwide, if the used frequency band is not supported by the device.[39]

3.2.2. Tablet Requirements Summary

The tablet needs to have WiFi capabilities, programmable network utilisation, an easy to use operating system and a high battery capacity is anticipated. Fortunately most modern operating system based tablets / smartphones satisfy these requirements in their basic configuration but they differ on the detail level. A 3G[5] feature plus data plan of a local mobile network provider allows the

[5] Third generation GSM data connectivity

application to be used in areas without WiFi coverage, for example on the road, but obviously mobile network coverage is required.
Another key requirement is the processing power, this dictates the smoothness of the user experience. Most computational expensive tasks shall be done by the middleware implementation to minimise the impact on the device battery life.

3.2.3. Choice of Hardware

Most of the current available smartphones and tablets fulfil the previously stated requirements. The choice of hardware for this thesis is narrowed to the group of iOS devices like the iPad or the iPhone, due their popularity and easy access for the author to an iOS device and developer license. It is important to highlight that devices of other vendors are in line with the iOS devices and it is a vital part of the implementation phase to keep the communication between the device and the middleware as interchangeable as possible to pave the way for non iOS implementations.

4. Environment

In the previous chapters the reader was introduced into the social background and Ambient Assisted Living in general and why this kind of technology is able to sustain the independence of the people and prolong vitality as well as the requirements for a tablet supported AAL application are defined. The following chapter will introduce the reader into the environment under a technical point of view by first giving an overview of all technical stakeholders and then iOS as well as the chosen AAL middleware and the hardware and connectivity between tablet and middleware.

Figure 4.1 sketches the technical overview, whereby the smart environment is a sensor and actuator equipped dwelling. These are linked by a variety of different bus implementations to the AAL middleware by one or more USB gateways to (i.e.) H.O.M.E.R. running on a PC hardware. The tablet interfaces with the AAL middleware over a classical TCP/IP network, either a local ethernet and/or the internet, see section 4.4 for details on the choice of protocol / technology.

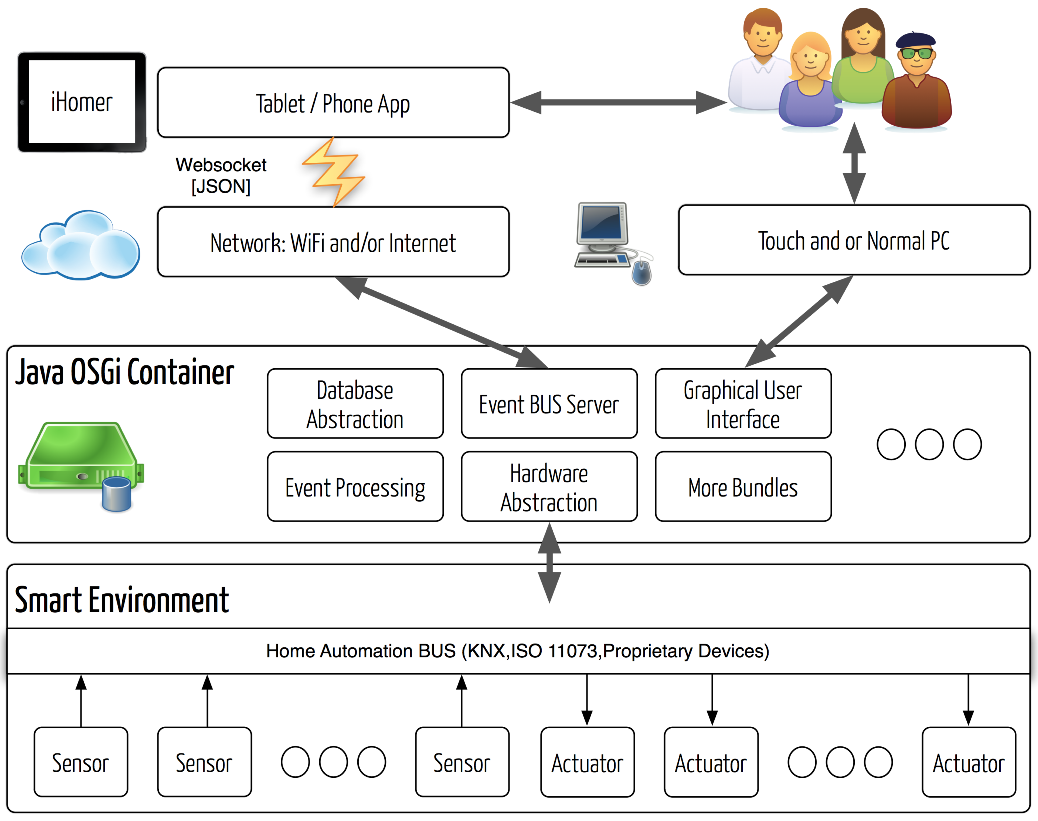


Figure 4.1.: Block Diagram Smart Home (Back-End, Sensors, Interfaces)

4.1. iOS Apple Mobile Operating System

The iOS-Device enabled user interface widens the range of possible Ambient Assisted Living applications tremendously. Utilising a mobile device for monitoring and controlling one or more Smart-Homes has many advantages over statically installed control terminals, in terms of costs and flexibility.

4.1.1. A brief history of iPhone OS / iOS

On the 9. of January 2007 Apple released iPhone OS with the introduction of the first iPhone. By the times of the initial release no public software development

kit was planed, only the ability to place a short cut to a web application to the home screen was officially supported. A web application is very different to a native deployed app, as they have some restrictions like not allowed to control the fullscreen mode or very limited 3D capabilities. By that time there is also no central App Store to search and find a web application. But the community discovered the potential of a highly powered mobile device and, despite the lack of a official method, first third party apps got developed, after the availability of a ARM/mach-o toolchain kit[1], by the growing community.[40][41]
The fact that it is a Mac OS X derivate, which was adapted for mobile use, gave a low entry barrier for developers. Apple recognized the high commercial potential of third party apps and opened the platform on the 11 of July 2008 with the release of version 2.0 to enrolled developers and released the App Store, a new closed ecosystem which acts as a one stop shop to buy apps from the developers. [42]
In June 2010, along with the introduction of the first generation iPad, Apple rebranded iPhone OS to simply iOS, not to mistake with the router operating system of Cisco, from which Apple licensed the name to avoid any potential lawsuit. The Apple App Store contained by February 10, 2012 over 650,000+ (225,000+ for iPad optimized) third party iOS applications, which have been downloaded about more than 30 billion times.[43]

4.1.2. iOS Development

Applications for iOS are written in Objective-C, an object-oriented programming language, and also a superset of the C programming language, which supports the same basic syntax as C. Similar to C code, header- and source-files are used to separate public declarations from the implementation details.[44]

.h Header-file, contains class, type, function and constant declarations

.m Source-file, implementing source file, may contain Objective-C and C code

[1] A set of various tools and compiler to cross compile binaries for the iPhone, developed of a hacker called "Nightwatch"

.mm	Source-File, implementing source file, same as *.m* but may contain C++ code in addition to Objective-C and C code

The application environment for iOS applications is provided by Cocoa, which is a set of object-oriented software libraries, a runtime system and the integrated development environment *Xcode* (see 4.1.2).

Architecture of iOS

There are four basic abstraction layers:[45]

Core OS	Contains the modified MACH kernel, file system, hardware interaction, accessory connection, security framework, threads, power management, networking infrastructure, system-level API
Core Services	provides core services like string manipulation, collection management, networking, URL utilities, contact management, preferences, location based services, services for hardware features like GPS, compass, accelerometer, gyroscope.
Media	Audio/Video framework, OpenGL ES framework, CoreGraphics, Quartz Core
Cocoa Touch	Build upon the Cocoa API of Mac OS X, delivers controls, touch interaction, inter process communication, written mostly in Objective-C

Xcode

Xcode is a free to download integrated development environment delivered by Apple for the creation of Mac OS X and iOS applications. Therefore its primarily focused on C, C++ and Objective-C, but because of its modular nature it supports also coding in other languages like Java, Ruby, Pascal and many others. The IDE itself runs only on Intel-processor powered Macintosh. This is an important fact, as any iOS development requires a Mac OS X able hardware for building/testing an app. It bundles also an so called Interface Builder to design user interfaces graphically and Instruments which features profiling and debugging tools, tailored for Objective-c, which allow a convenient way to directly debug in a simulated iOS environment or even on the plugged device itself.[46]

App Store

Back on July 10, 2008 the App Store for iOS apps, opened their virtually gates to the public, allowing registered third party developers selling their apps in a one stop shop. Developers need a paid subscription, in order to be allowed uploading any app to the App Store, even free ones. Apple keeps 30% of the revenue for operating the App Store, credit-card fees and other expenses. The developer point of view of the App Store is provided by iTunes Connect, which holds financial contracts, payments and financial reports, sales numbers, app administration. Before an app is going to be submitted its basic information needs to be setup within the iTunes Connect Application Manager. This step entails any marketing relevant information like App Store description, artwork and pricing details.[47]

Device Deployment

There are several requirements a developer has to meet, when it comes to app deployment. A developer needs to enrol in one of the three iOS Developer programs, depending on how the developer plans to distribute the apps. The standard program allows the distribution on the App Store, whereby the enterprise program is used for internal distribution only. Educational institutions may apply for a free university program if they plan to adopt iOS development in their curriculum.[48] After enrolling it is mandatory to register every testing iOS device in the provisioning portal. In return a developer certificate and provisioning profile will be generated and need to be installed on the developing machine and on each testing device. This mechanism is used to allow only code-signed software to be run on the iOS devices. Without enrolling it is still possible to develop apps and test the apps in the provided iOS simulator, but not on a physical mobile device. The simulator however, comes with some restrictions, and therefore it is not guaranteed that any app which is only tested within the means of the simulator will successfully run on a real iOS device.[49]

4.2. AAL Middleware

The State of the Art section 2 introduced the careful reader to current developed and currently in development AAL middleware solutions. The modular and versatile H.O.M.E.R platform will be used for the implementation environment of this thesis. However, most implementation details are AAL middleware agnostic designed, so the user interface can be easily adopted to most other middleware providers by implementing the matching interface.

4.2.1. H.O.M.E.R

H.O.M.E.R (**HOM**e **E**vent **R**ecognition System), developed by the Austrian Institute of Technology[16], is an open and flexible OSGi-based software platform, which aims at the integration of various automation systems and protocols, ISO/IEEE 11073[2] compatible devices and proprietary (if drivers are adapted to the generic model), wireless and wired sensors as well as actuators and consequential event and situation recognition for smart home, addressing comfort, energy efficiency and various Ambient Assisted Living applications like safety, autonomy, self-confidence, etc. There are currently two demonstration installations of the H.O.M.E.R platform available, the first one in an energy efficient passive house at the “Blaue Lagune” in Vösendorf demonstrating the energy balance and other features to visitors. The other one is an scientific demonstration flat consisting of two rooms called “2-Raumwohnung” at the vicinity of the Austrian Institute of Technology.[16]

[2]ISO/IEEE 11073 standard: `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=36347`

4.2.2. Platform / OSGi

The OSGi Alliance[3] is a worldwide consortium of technology innovators to create an open specification that enables a modular software around the Java technology, which in return reduces software complexity.
In OSGi terms a software module is called bundle and consists usually of a single JAR file, whereby this file is just a ZIP compressed folder structure containing the implementation classes alongside the mandatory `MANIFEST.MF` file within the `META-INF` folder. This file describes some bundle meta information details like bundle version, description and most important any dependent Java package imports and or services used by the bundle, as well as offered services and packages.[50]
H.O.M.E.R. is divided into the following OSGi bundles, whereby each bundle may provide and/or consume certain services:

Homer Core OSGi Bundles

Common	Database abstraction, domain model
Database	Database access through different option of datasources
JDBC	JDBC driver for SQLite,MySQL and PostgreSQL
GUI	Graphical user interface, foundation of dynamic window components, configuration and monitoring, editing state machine scenarios
IsoAgent	Implementation of the ISO 11073 standard and communication structures
Processing Event	Interface to UniMod state machine engine, loading and saving state machine configurations, sensor input simulator and playback of existing sensor data sets
Processing Position	Positioning data module

[3]OSGi: Open Service Gateway initiative. See `http://www.osgi.org`

UniMod	Encapsulates the UniMod state machine open source project (see 4.2.2)
EventBus Server	Websocket eventbus server allows state events, sensor events to be published
Shell Commands	deploys commands to be used within the OSGi Karaf shell

Standards and Technologies

The H.O.M.E.R. platform also endorses the following standards and technologies.

ISO/IEEE 11073-10471:2008	Health Informatics - Personal health device communication Part 10471: Device specialisation-Independent living activity hub
Apache Karaf	An OSGi framework[4]
Aries Blueprint	OSGi blueprint implementation[5]
Apache Maven	for building and modular configuration[6]
UniMod	Java Finite State Machine Framework for designing object-oriented event-driven applications published under LGPL v2.1[7]
Openhealth Project	mobile technologies research project for eHealth and well being[8]
BinaryNotes ASN.1	Abstract Syntax Notation One framework[9]

[4]Apache Karaf URL: `http://karaf.apache.org`
[5]Maven URL: `http://maven.apache.org`
[6]Apache Maven URL: `http://maven.apache.org`
[7]Unimod URL: `http://unimod.sourceforge.net`
[8]Openhealth URL: `http://openhealth.libresoft.es`
[9]BinaryNotes URL: `http://bnotes.sourceforge.net`

Databases	SQLite[10], MySQL[11], PostgreSQL[12]

4.2.3. Sensors

The H.O.M.E.R. platform unites all different kind of sensors, by mapping sensor input to the standardised H.O.M.E.R. data model, to be able to process any sensor input even further without the hassle of different vendor protocols. The table 4.1 gives a brief overview of all currently available sensor types.

Detection Sensor	**Unit**	**Description**
Contact	Boolean	Impulse on contact. e.g. button,drawer
Switch	Boolean	
Dosage	Boolean	
Enuresis	Boolean	Waterproof mat to detect bed wetting/moisture
Fall	Boolean	
Gas	Boolean	Gas concentration
Smoke	Boolean	
Motion	Boolean	Passive infrared sensor
Property exit	Boolean	
Window Handle	Boolean	
Usage	Boolean	
Water	Boolean	
Metering Sensors		
Metering Current	Ampere	
Metering Current Generation	Ampere	
Metering Energy	Joule	
Metering Energy Generation	Joule	
Metering Power	Watt	
Metering Power Generation	Watt	
Metering Voltage	Volt	
Metering Voltage Generation	Volt	
Brightness	Candela	
Temperature	Celsius	

Table 4.1.: Overview of various sensors

[10]SQLite URL: `http://sqlite.org`
[11]MySQL URL: `http://mysql.com`
[12]PostgreSQL URL: `http://www.postgresql.org`

4.2.4. Actuators

Actuators are devices who are able to perform some kind of action, like opening / closing shades, switches and dimmers. Actuators may be triggered by the framework or any other software component and by any hardware control.

4.2.5. Bus System

Those actuators and sensors are connected by usually a vendor specific bus system. H.O.M.E.R. translates the vendor specific sensor / actuator protocols into one generic protocol to avoid any vendor lock in, in the future.

4.3. Tablet

The choice of hardware has been narrowed down to the group of iOS enabled tablet devices. By the time of writing 3 of 5 members are on sale, compare with table 4.2. The Apple Inc. introduced the first generation iPad WiFi only on April 3, 2010, the WiFi + 3G model followed on April 30. Right from the start it gained a high spark of interest to open up new fields of application and in some areas it supersedes a conventional personal computer due to its form factor, ease of use, touch interface and performance. Steve Jobs former CEO of Apple Inc. said after unveiling the iPad 2. Generation:

> “The post-PC world would be dominated by such devices as smartphones and tablets. Some other vendors view tablet as something new in the PC market, but that is not the right approach to this [. . .] These are post-PC devices that need to be easier to use than a PC, more intuitive. The hardware and software need to intertwine more than they do on a PC” [51]

Spec	**iPad Gen2**	**iPad Gen4**	**iPad Mini**	iPad Gen1*	iPad Gen3*
Display [Pixel]	9.7-inch 1024x768	9.7-inch 2048x1536	7.9-inch 1024x768	9.7-inch 1024x768	9.7-inch 2048x1536
iOS	7.0	7.0	7.0	5.1.1	7.0
Capacity [GB]	16,(32,64)*	16,32,64,128	16,32,64	16,32,64	16,32,64
Dimension [mm]	240x186x8.8	240x186x9.4	200x130x7.1	243x190x13.4	240x186x9.4
Weight [g]	601	650	310	680	650
Battery [h]	9-10	9-10	9-10	9-10	9-10
Chip	Dual-core A5	Dual-core A6X	Dual-core A5	A4	Dual-core A5X
Cameras	0.7 MP back 0.3 MP front	5 MP back 1.2 MP front	5 MP back 1.2 MP front	-	5 MP back 1.2 MP front
Sensors	Accelerometer, ambient light sensor, magnetometer, gyroscope (iPad Gen1 no gyroscope)				
Wireless	Wi-Fi 802.11a/b/g/n Bluetooth 2.1 + EDR	Wi-Fi 802.11a/b/g/n Bluetooth 4.0	Wi-Fi 802.11a/b/g/n Bluetooth 4.0	Wi-Fi 802.11a/b/g/n Bluetooth 2.1 + EDR	Wi-Fi 802.11a/b/g/n Bluetooth 4.0
3G/LTE	3G optional	3G,LTE optional	3G,LTE optional	3G optional	3G,LTE optional

Table 4.2.: Overview of – as time of writing – *available* and **discontinued* Apple iPads running iOS.
Source: `http://store.apple.com/us/ipad/compare`[52]

4.4. Connectivity

The tablet device needs some sort of network communication to the given AAL framework. There are, as time of writing, three key communication technologies commonly used nowadays. The classical approach are web services using the WSDL standard, REST calls and some varieties, or as a bus approach web sockets. The following subsections state the basic idea, as well as advantages and disadvantages of each technology.

4.4.1. Web Services

There are many ambiguous definitions of a web service, but for the given context the following definition of w3.org shall be used:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using

> HTTP with an XML serialisation in conjunction with other Web-related standards. [53]

While the definition mentions the WSDL as machine-processable format, three different key implementation techniques has been emerged over the years. These are discussed in short by the next subsections.[54]

SOAP

SOAP - originally short for Simple Object Access Protocol - is a protocol specification used to interface and exchange structured information for web services. Its XML based and contains the following elements:

Envelope*	Identifies the XML as SOAP
Header	Header informations
Body*	Call and response information
Fault	Information about errors during the processing of the message
*	The asterisk determines a mandatory element.

The interface definition is provided by a WSDL file whereby WSDL is the acronym for *Web Services Description Language* which is a platform and protocol independent XML language to describe web services. It contains primarily the interface location, access protocol and any information regarding the access to the service like object definition, method names and return values. The current version of WSDL is 2.0 but its still poor supported by various software development kits, which still endorse the old WSDL 1.1. The greatest strength of SOAP/WSDL, given the clear enforced interface declaration, is also the weak spot of this technology definition. Some implementations on different platforms have or had issues with complex data types which lead to major – often time consuming – complications. Another disadvantage of having a data message wrapped in a quite verbose XML stream is the vast overhead on the line, which on both ends translates into more computational effort in creating and parsing SOAP data. For this project its mandatory to have a small computational overhead and a low bandwidth utilisation due to the mobile data nature of this project, so SOAP stands in this case for a not optimal solution.[54]

REST

REST is short for *Representational State Transfer* and describes the idea of interfacing on an architectural level. While its not exactly the opposite of SOAP its often referred as such when it comes to the debate between both. Remember SOAP is still a protocol which may be used within a REST architecture, however, this is unlikely the case. REST architecture is often based on the very basic JSON notation as described in the next paragraph. REST dictates a stateless client - server architecture, accessed by a defined URL per each service – also called resource in the REST context – and is accessed by a resource identifier, usually an URL. A representation is then transmitted to this resource identifier triggering the particular action the REST component is designed for. On the server side implementation there are usually various entry points, which are managed by a router component. This various entry points coincide into a API which is able to serve numerous clients.[54][55]
JSON stands for *Java Simple Object Notation* and denotes a very simple way to structure data messages for transportation. It has a very low notation overhead and thus perfectly serves a data exchange protocol in a mobile and web environment. JSON gained a spark of interest over the last years promoting it to a highly accepted standard for message exchange. Software development kits supporting REST / JSON applications are available for all major coding languages. In the iOS universe a advanced implementation is provided by RestKit, which automates REST calls and message building as well as provides a CoreData integration for caching purposes.[56][57]
Table 4.3 gives an overview of all datatypes available in JSON.
A simple pretty formatted JSON example featuring an object (a flat) containing the flatId, name, as well as 2 rooms, each as an object encapsulating roomId and

Type	Example
String	`"Hello"`
Boolean	`true|false`
Integer	`123`
Number	`0.1234`
Array	`ordered set of elements between square brackets [...,...]`
Object	`unordered set of objects between curly brackets \{key:value,...:...}`
null	`-empty-`

Table 4.3.: Compiled overview of JSON datatypes.
Source: [56]

name.

Listing 4.1: Simple flat representation in JSON

```
{
  flatId: 1,
  name: 'Alexanders Flat',
  rooms: [
    {
      roomId: 1,
      name: 'Living Room'
    },
    {
      roomId: 2,
      name: 'Bath'
    }
  ]
}
```

Both approaches are limited regarding live update capabilities, this means there must be some sort of polling for information, due to the fact that running a two way communication on the mobile end device is not recommended as most mobile network operator restrict accessibility by filtering incoming traffic, jeopardising any possibility running a simple server instance on the mobile device. A possible workaround to this limitation is to keep the sending port always open like a stream of data thats not going to terminate after each message. This web application model is called among others:Comet, Ajax Push, Two-way-web.[58]

4.4.2. WebSocket

Web sockets are a protocol based on a single TCP connection which allows bidirectional (full-duplex) communication between server and client. It is been standardised by the Internet Engineering Task Force (IETF) as `RFC 6455`[13] in 2011. In contrast to HTTP where each answer of the server requires a request of the client, like polling for information, is a web socket connection, once opened ready for any data transmission. This is similar to the previous discussed comet workaround.
WebSockets are agnostic to the data encoding and depending on the usage scenario JSON may be used, among others.[59]

4.4.3. Comparisons

The following table 4.4 tries to visualise the different approaches. The thoughtful reader will easily spot the overhead of protocol for SOAP messages.

[13]`http://tools.ietf.org/html/rfc6455`

Technology	**API Example**

SOAP / WSDL
getFlat(1)
Request
	`POST /1.0/flat HTTP/1.1`
	`Host: api.example.org`
	`Content-Type: application/soap+xml; charset=utf-8`
	`Content-Length: nnn`
	
	`<?xml version="1.0"?>`
	`<soap:Envelope`
	`xmlns:soap="http://www.w3.org/2001/12/soap-envelope"`
	`soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">`
	
	`<soap:Body xmlns:m="http://api.example.org/1.0/flat">`
	`<m:GetFlat>`
	`<m:flatId>1</m:flatId>`
	`</m:GetFlat>`
	`</soap:Body>`
	
	`</soap:Envelope>`

REST / JSON
getFlat(1)
Request
	`GET /1.0/flat/1 HTTP/1.1`
	`Host: www.example.org`
	`Content-Type: application/soap+xml; charset=utf-8`
	`Content-Length: nnn`

SOAP / WSDL
getFlat(1)
Response
	`HTTP/1.1 200 OK`
	`Content-Type: application/soap+xml; charset=utf-8`
	`Content-Length: nnn`
	
	`<?xml version="1.0"?>`
	`<soap:Envelope`
	`xmlns:soap="http://www.w3.org/2001/12/soap-envelope"`
	`soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">`
	
	`<soap:Body xmlns:m="http://api.example.org/1.0/flat">`
	`<m:GetFlatResponse>`
	`<m:FlatId>1</m:FlatId>`
	`<m:FlatName>Alexanders Flat</m:FlatName>`
	`...<more flat specific values>`
	`</m:GetFlatResponse>`
	`</soap:Body>`
	
	`</soap:Envelope>`

REST / JSON
getFlat(1)
Response
	`HTTP/1.1 200 OK`
	`Content-Type: application/json; charset=utf-8`
	`Content-Length: nnn`
	
	`flatId:1,name:'Alexanders Flat',rooms:[roomId:1,name:'Living Room',roomId:2,name:'Bath'],`
	`...<more flat specific values>`

Table 4.4.: Simple connectivity technologies comparison.
Source: [56][60]

5. Activity Recognition

Activity recognition is the task of monitoring an actors behaviour by the use of all different kind of sensors and analysing the generated data to infer the actors activities. This covers many different tasks, like activity modelling, behaviour and environment monitoring, data processing and pattern recognition. The following steps give a brief overview of the activity recognition process, defined by Liming Chen and Ismail Khalil (2011)[61].

1. Create computational activity models in a way that allows software systems/agents to conduct reasoning and manipulation.

2. Monitor and capture a user's behaviour along with the state change of the environment.

3. Process perceived information through aggregation and fusion to generate a high-level abstraction of context or situation.

4. Decide which activity recognition algorithm to use.

5. Carry out pattern recognition to determine the performed activity.

In this chapter we are going to give a brief overview of current monitoring approaches which gather data for our activity model. Later on we are introducing different approaches for the processing of the activity data, considering various methods. Ultimately a particular method will be implemented and results using the activity benchmark data of Kasteren et al.[62] will be shown.

5.1. Monitoring Approaches

There are many different approaches for activity recognition, namely vision-based activity recognition and sensor based recognition, whereby the former uses surveillance systems and the later needs a network of different sensor technologies.

The vision-based approach has a strong history due to efforts made in surveillance research to detect and track humans, which works very well under a specified environment, however, there are disadvantages when it comes to reusability due to complexity of real world scenarios and the invasive perception of being monitored in the own home environment.

The sensor-based approach can be broadly divided into unobtrusive and obtrusive sensor technologies. Hereby obtrusive sensors, usually wearable sensors, can be attached to a person, like a heart rate sensor, temperature sensor, inertial sensor. In research, wearable sensors are widely used in the recognition of physical movements, like walking, sitting up/down, physical exercises, as those are generally characterised by a distinguishable periodic pattern. It is also an effective and relatively financially inexpensive method for data gaining, however, it disperses some drawbacks. As with all obtrusive technologies, human acceptance is generally very low, due to the foreign matter, an actor has to deal with in combination with size, ease of use, and battery life, but may be neglected in some cases. In some cases, ambiguous data may be gained of such wearable inertial sensors, for example there is, under some circumstances, no difference between making coffee or making tee.[61]

The unobtrusive object-based activity recognition approach has gained a great spark of interest as low-cost-low-power intelligent sensors with wireless networks and pervasive computing infrastructure becomes financially affordable and technically mature. Simple binary sensors are widely used for equipping dwellings in an ambient assisted living context to monitor the actors movement and inferring the conducted activities. [63][64][65]

5.2. Activity Recognition Algorithms

There are two different approaches when it comes to activity recognition algorithms, one is based on logical modelling and reasoning, which will not be discussed in this thesis, but interested readers may be pointed to [61, p. 7-8]. The other technique is based on machine learning, which covers supervised and unsupervised learning methods.

5.2.1. Supervised Learning

Supervised learning requires the use of annotated data to train the algorithm, afterwards the trained algorithm is used to classify unannotated data. As pointed out by Liming Chen and Ismail Khalil (2011) [61, p. 5-6], the following steps are sketching the general procedure:

1. to acquire sensor data representative of activities, including labelled annotations of what an actor does and when,
2. to determine the input data features and its representation,
3. to aggregate data from multiple data sources and transform them into the application- dependent features, e.g., through data fusion, noise elimination, dimension reduction and data normalisation,
4. to divide the data into a training set and a test set,
5. to train the recognition algorithm on the training set,
6. to test the classification performance of the trained algorithm on the test set, and finally
7. to apply the algorithm in the context of activity recognition.

Examples of widely used supervised learning algorithms in the context of activity recognition are Hidden Markov Models (HMM)[66][67], conditional random fields (CRF)[66][68], dynamic and naive Bayes networks[62], nearest neighbour, decision trees[69] and support vector machines (SVM)[70].

5.2.2. Unsupervised Learning

Unsupervised learning, in contrary to supervised learning, tries to classify patterns from unannotated data, by manually assigning a probability to each observable activity, which gets updated by a pre-defined stochastic model according to new observations. Liming Chen and Ismail Khalil (2011) [61, p. 6] summarised the needed steps:

1. to acquire unlabelled sensor data,
2. to aggregate and transforming the sensor data into features, and
3. to model the data using either density estimation or clustering methods.

Examples of used unsupervised learning algorithms for activity recognition are graphical models[71] and multiple eigenspaces[72] and some are also like supervised learning algorithms based on variants of HMMs and Bayes networks.

5.3. Hidden Markov Model

There are a myriad models based upon the Markov assumption, that a new value depends on the previous values somehow. Often this assumption is tightened to just the very last value, which is then called a first order relationship. The Hidden Markov Model is often used when it is not possible to directly observe model sequences, but rather the source for the changes in observations.[73]
Hidden Markov Models are based upon the theory of Markov chains, which has been known by engineers and mathematicians for over hundred years by now, however, the method for optimising the parameters of a Markov model to match observed signal patterns was developed in the late 1960's.[74] Hidden Markov Models have a great spark in speech recognition since then, but it is now used in a broad spectrum of areas, like finance stock prediction, written text recognition, gesture recognition and many more. In the year of 1989 Rabiner et al. published a read worthy introduction into Hidden Markov Models[74].

5.3.1. Markov Chain

A detailed explanation of the Markov Chain are found in the papers [74][75]. As already mentioned in the introduction, we are relying on the discrete time Markov chain of first order:

$$P(Q_{t+1} = q_{t+1} | Q_t = q_t, Q_{t-1} = q_{t-1}, \ldots, Q_0 = q_0) = P(Q_{t+1} = q_{t+1} | Q_t = q_t) \tag{5.1}$$

Q_t states the random variable at time t and q_t represents the variable for some state at time t, which concludes to our assumption that being in some state at a particular time t is only dependent on the previous state.

5.3.2. Notation and Definition of the Hidden Markov Model

We use the following notation for our variables:

T = length of the observation sequence

N = number of states in the model

M = number of observations symbols

Q = $\{q_0, q_1, \ldots, q_{N-1}\}$ distinct non observable (hidden) states

V = $\{0, 1, \ldots, M-1\}$ set of possible observations

A = $\{p_{ij}\}$ state transition probability distribution matrix

p_{ij} = $P\{Q_{t+1} = j | Q_t = i\}$
for $0 \leq p_{ij} \leq 1$ and $\sum_{j=1}^{N} p_{ij} = 1, 1 \leq i, j \leq N$

B = $\{b_{ik}\}$ state emission transition probability distribution matrix

b_{ik} = $P(O_t = k | Q_t = i)$
for $1 \leq i \leq N$ and $0 \leq k \leq M$

π	=	initial state distribution, all elements have to be positive and sum to unity $\pi_i = p\{Q_0 = 1\}, 1 \leq i \leq N$
O	=	observation sequence
λ	=	(A, B, π) HMM model

5.3.3. Three Problems solved by Hidden Markov Models

Hidden Markov Models provide the foundation to solve the following problems:

Evaluation Inferring the probability $P(O|\lambda)$ of an observation sequence O given the fully characterised model $\lambda = (A, B, \pi)$

Decoding Finding the path of hidden states that most probably generated the observed output sequence O.

Learning Generating a Hidden Markov Model given sequences of observations

Please note, it is important to differentiate between *structure learning*, which tries to find the optimal number of states and the paths between and *parameter estimation*, fitting the transition and emission probabilities.

Forward Algorithm

The evaluation problem is solved by the forward algorithm also called $\alpha - pass$. For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$ define

$$\alpha_t(i) = P(O_0, O_1, \ldots, O_t, x_t = q_i|\lambda) \tag{5.2}$$

$\alpha_t(i)$ depicts the probability of the partial observation sequence up to the time t, where the Markov process is in state q_i at time t. $\alpha_t(i)$ can be computed recursively:

1. Let $\alpha_0(i) = \pi_i b_i(O_0)$, for $i = 0, 1, \ldots, N-1$

2. For $t = 1, 2, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, compute
$$\alpha_t(i) = \left[\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji}\right] b_i(O_t) \tag{5.3}$$

3. From 5.2 we see that
$$P(O|\lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i) \tag{5.4}$$

Viterbi Algorithm

The decoding problem is solved by the Viterbi algorithm. It defines a path probability, which gives the probability of reaching a particular intermediate state following the most likely path. Thus, it gives us the probability for the best path by keeping a backlog of each predecessor of every state, which allows us to follow this specific path. It computes the maximum value for each step as follows:

1. For $t = 1$ compute
$$a(1) = argmax_j(\pi(j) b_{jt}) \tag{5.5}$$

2. For $t \geq 2$ compute
$$a(t) = argmax_j(a_{t-1} b_{jt}) \tag{5.6}$$

5.4. Build Model Structure

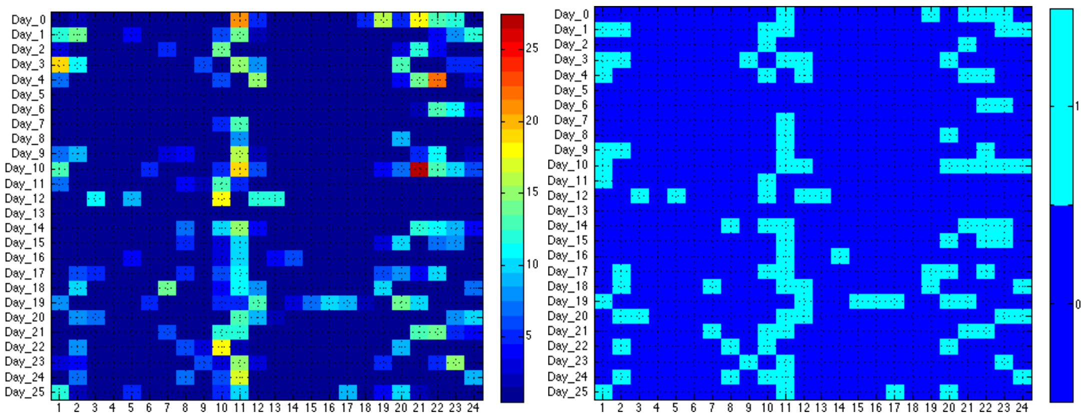
For the iOS application we are going to implement a system based upon a Hidden Markov Model, which learns semantic information of various binary user induced motion events. We proceed for the structural learning like Bruckner et al. introduced in their paper [76] and [73].

1. Filtering motion induced sensor data
2. Comparison of the chain's beginning / end
3. Merging of identical states
4. Merging of consecutive states
5. Merging of non-relevant states

We assume $T_{interval} = 1$ hour, so we get 24 values of $V = \{0, 1\}$ and thus, if we treat each activity interval as a state, we get $N = 24$ states for a day. This steps produce a Hidden Markov Model with an overseeable count of states due merging. Bruckner et al. state that the number of states is a compromise between generalisation, which means a low number of states, thus the model is applicable for a broader range of different scenarios and specialisation, whereby a high number of states depicts not every possible scenario and similar scenarios may have different paths. Also Stolcke et al.[77] showed that model merging of rather special models always conlude in better or equal results than specialising in very general models.

5.4.1. Sensor Data

The following figure 5.1 shows an example of all user induced binary sensor reading activations accumulated and sliced into 24 bins per day. The data was taken from the benchmark data, whereby (a) depicts the raw values and (b) the filtered readings by a threshold of at least 5 activations within an hour. This threshold is a trade-off between being to specific about the gathered readings and not to lose too much useful input. One might consider a different filtering approach. This figure shows for example that there is usually an activity around 9am until 10am, and the inhabitant returns around 6pm and gets into bed between 0pm and 1am. There are also seldomly some night activities, possibly going to the toilet or maybe having a restless night. This data was recorded and annotated by Kasteren et al., whereby the data shown is *Flat A*. For further details see the article [62] and download the data from this URL[1].



(a) Raw data (b) Filtered by at least 5 readings

Figure 5.1.: User induced sensor reading count grouped into 24 bins per day

This particular data will be used to build our model in the following sections.

5.4.2. Chain Building

Each day activity values are translated into states. In our case, we decided to make $T_{interval} = 1$ hour, which concludes into 24 states for each day. We are going to extend the structure of our model by extending the 24 states of the first iteration by a particular count of days to cover all important activity aspects of our inhabitant. In figure 5.2, we see that day 1 begins with 0, then switches to active and inactive again. Day 2 begins like day 1, but at 4th hour both days begin to differ. To tackle this, we are going to make a branch and adjust our transition probability accordingly. If there would be a third day also differentiating at this particular step, we have to adjust the transition

[1]used dataset download url: `https://sites.google.com/site/tim0306/codeFramework.zip?attredirects=0`

probability again matching the observed emission. This process may also be done in backward direction, see [78] for further details.

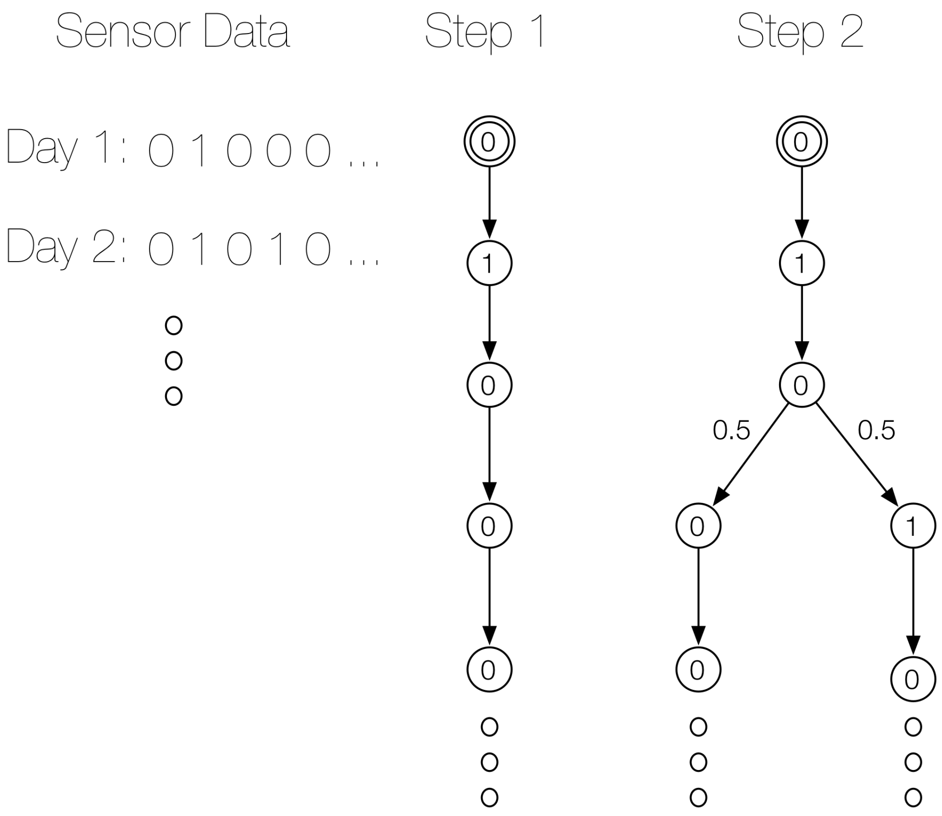


Figure 5.2.: Chain building procedure

5.4.3. Merging of States

Bruckner et al. present[73] a heuristic for merging identical states as follows. If the sensor readings are active over our defined $T_{interval} = 1$ hour, we can assume, the behaviour of the inhabitant continues. A similar assumption is made for the emission of prolonged inactive sensor readings. In conclusion, states with a 100% transition probability and identical emissions $Q = \{\ldots, 1, 1, 1, \ldots\}$ are merged into a single one. The parameters $P(Q_{ii})$ self transition probability and $P(Q_{ij})$

transition probability of the state are calculated as follows:

$$P(Q_{ii}) = \frac{N_{merged}}{N_{merged} + 1} \tag{5.7}$$

$$P(Q_{ij}) = \frac{1}{N_{merged} + 1} \tag{5.8}$$

$$P(Q_{ij}) + P(Q_{ii}) = 1 \text{ and } b_{ik} = 1 \tag{5.9}$$

After merging identical states, we deal with chains of states having a 100% transition probability into one state. This applies to state chains with a 100% transition probability and alternating emissions like $Q = \{\ldots, 0, 1, 0, 1, 0, \ldots\}$. The parameters $P(Q_{ii})$ and $P(Q_{ij})$ are computed as in the previous case, but the emission probability distribution is different.

$$b_{i0} = \frac{N_{count0}}{N_{merged}} \tag{5.10}$$

$$b_{i1} = \frac{N_{count1}}{N_{merged}} \tag{5.11}$$

$$N_{merged} = N_{count0} + N_{count1} \text{ and } b_{i0} + b_{i1} = 1 \tag{5.12}$$

5.5. Results

We have now seen how to model a daily routine using the Hidden Markov Model approach. Furthermore, we were able to reduce the state count to a manageable number. There are even further possibilities mentioned by Bruckner et al. [73], like the merging of non-relevant states.

The sensor data of all motion induced readings is shown in section 5.4.1. We used this data to build our Hidden Markov Model, but first, we sort the days from left to right according to their activity state in order to avoid unnecessary state branching. Figure 5.3 shows the HMM after the chain building step from left to right. The start state is on the left side and each grey circle represents a state with an emission probability of either 1 (active) or 0 (inactive) for every hour. Different branches may represent a different behaviour of the observed person.

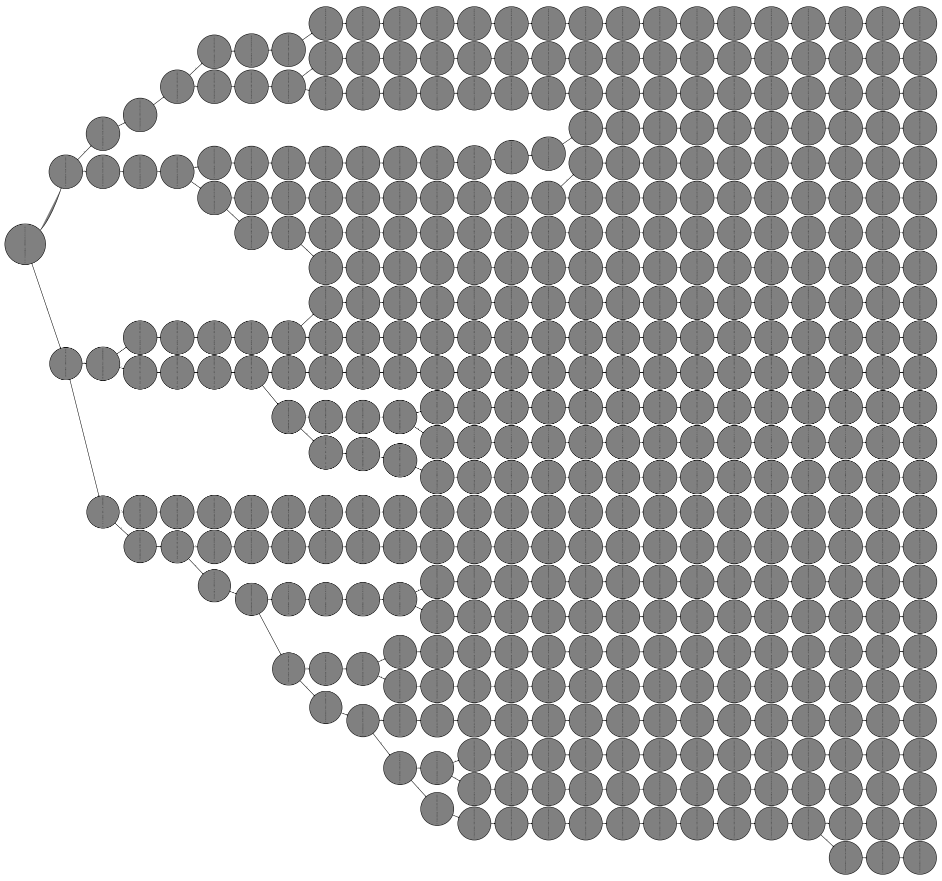


Figure 5.3.: Chain building of sensor data resulted HMM

Next we apply the state merging procedures of section 5.4.3, which boils down to the model shown in figure 5.4. Comparing the previous figure 5.3 with 5.4, we clearly notice the reduced state count. Also note that some states do have a self transition probability marked by a backward arrow on the top of a state[2].

[2]If you are reading the PDF version, you should be able to zoom in for more details.

We now have an HMM with all parameters $\lambda = (A, B, \pi)$ set. We are now able to execute the Forward- and Viterbi algorithm to calculate the probability of an observed sequence against the model, and to follow the most likely path of a given sequence.

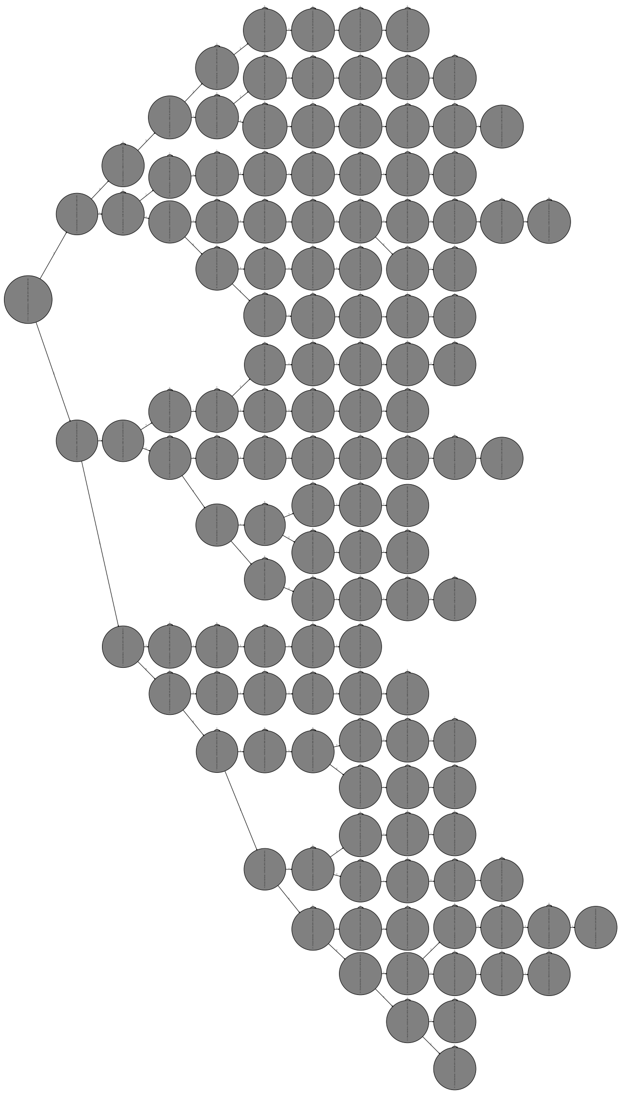


Figure 5.4.: State merging resulted HMM

If we calculate each daily observed sequence probability, drawn as the neperian logarithm in figure 5.5, we see that there are several best adapting days until 9am, afterwards the top line represents day 5 and 13, which both are days without any activity, scoring the best adaption with a logarithm probability of −6.22. The best active day is 7 and the worst adapting sequence, day 18, has a logarithm probability of −20.05.

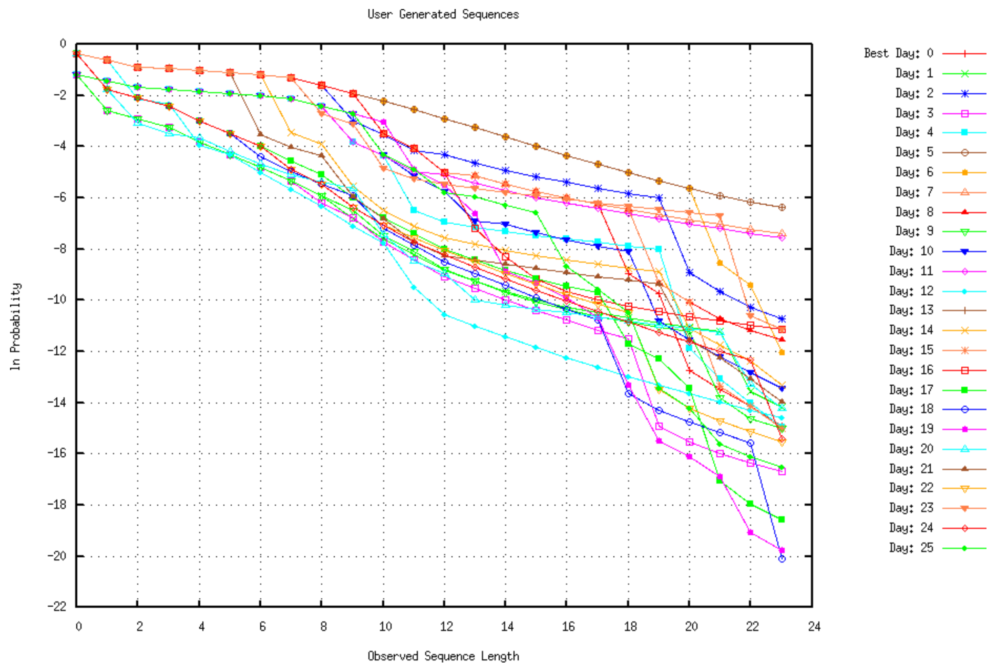


Figure 5.5.: Each observed sequence compared against the merged HMM model

There are cases where the state value is not matching the observed value and is thus not be able to adapt to the observation sequence, however, Bruckner et al. are proposing to send a 1% value to the transition matrix A in this particular step, allowing the comparison to continue. This cases are visualised by steep drops in the logarithm value of figure 5.6.

The calculated neperian logarithm of random generated observation sequences with the HMM are shown in figure 5.6. The attentive reader will notice the steeper graph character. Also the best adapting sequence, drawn as a a blue line, comes from several observation sequences, however, the best adapting sequence, random day 19, is depicted by the purple line with square.

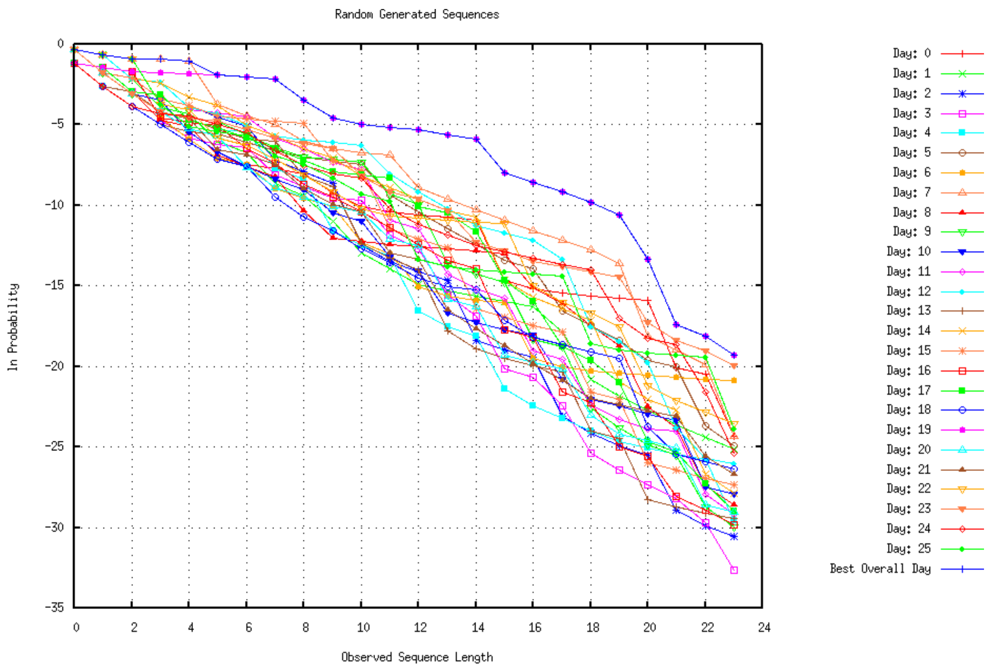


Figure 5.6.: Random generated sequences compared against the merged HMM model

5.5.1. Find The Best Day

Bruckner et al. describe a method[73] to find the best day, or in other words the best daily sequence, which represents a standard day of the inhabitant, out of all, in our case 26 days. They first again sorted all days by their activity value from left to right and compared the best state sequence of each sorted day index with the state probability of each unsorted day index.

original Index $\updownarrow$ sorted Index	0 $\updownarrow$ 6	1 $\updownarrow$ 24	2 $\updownarrow$ 8	3 $\updownarrow$ 25	4 $\updownarrow$ 22	5 $\updownarrow$ 0	6 $\updownarrow$ 2	7 $\updownarrow$ 3	8 $\updownarrow$ 4	9 $\updownarrow$ 23	10 $\updownarrow$ 20	11 $\updownarrow$ 21	12 $\updownarrow$ 12
original Index $\updownarrow$ sorted Index	13 $\updownarrow$ 1	14 $\updownarrow$ 10	15 $\updownarrow$ 5	16 $\updownarrow$ 7	17 $\updownarrow$ 13	18 $\updownarrow$ 16	19 $\updownarrow$ 18	20 $\updownarrow$ 17	21 $\updownarrow$ 11	22 $\updownarrow$ 14	23 $\updownarrow$ 9	24 $\updownarrow$ 15	25 $\updownarrow$ 19

Table 5.2.: Mapping original index vs sorted, by activity level, index

The result is shown in figure 5.7. Obviously each day matches itself best, so the worst day depicts the worst adapting day. We see that the days with sorted (original[3]) index 18 (16), 20 (17) and 12 (12) are likely to adapt worst. Considering this information a rough judgement, we are continuing by comparing further parameters.[73]

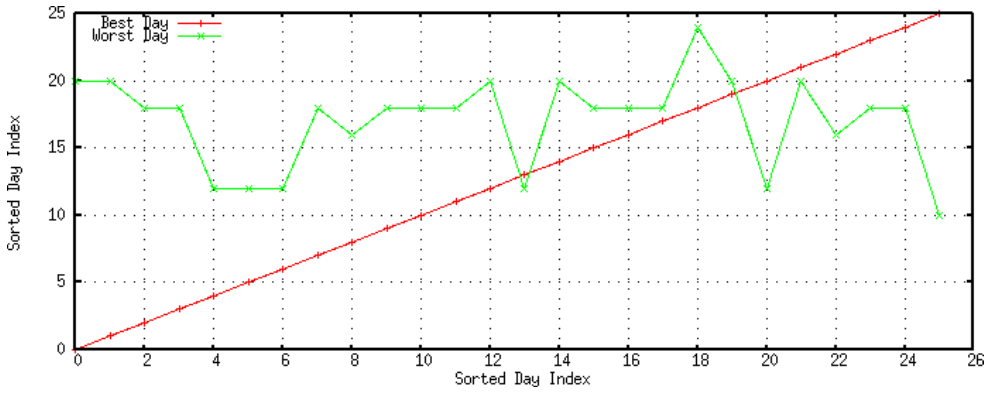


Figure 5.7.: Sorted by activity day index vs original day index

Comparing each day with all other days and noting the neperian probability of the best and worst adapting day gives us the figure 5.8. The days with sorted (original) index 0 (5), 1 (13) score the best values of −8, followed by day 3 (7) and 21 (11). Their worst adaption probability values are between −40 and −45. The overall range is −8 to −21 for the best days and −40 to −70 for the worst days.

[3]The mapping between the sorted index versus the original index is shown in table 5.2.

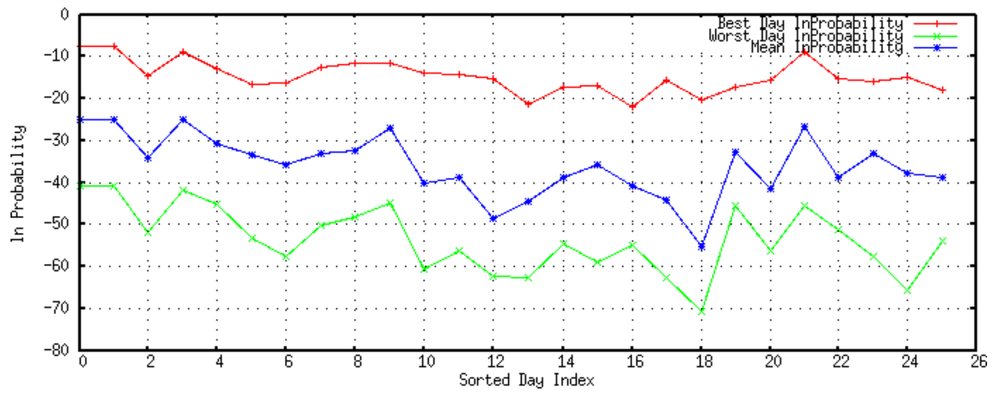


Figure 5.8.: Best day versus worst day probability for sorted day index

The blue line shows the mean value which, reinforces the assumption that the days with sorted (original) index 0 (5), 1 (13) have more similarity to all other days than the day with index 18 (16). However, sorted index days 0 and 1 are both without any activity, so the best first active day matches are day 3 and 21.

All logarithm values summed up from comparing each day against all other days gives us an indication of the deviation between one particular day and all others days, which is shown in figure 5.9. We see that similar to the previous figure 5.8, days with sorted (original) index 0 (6), 1 (24) and 21 (11) have the best logarithm probability value around -650. Day 18 (16) has the worst logarithm value of -1440.

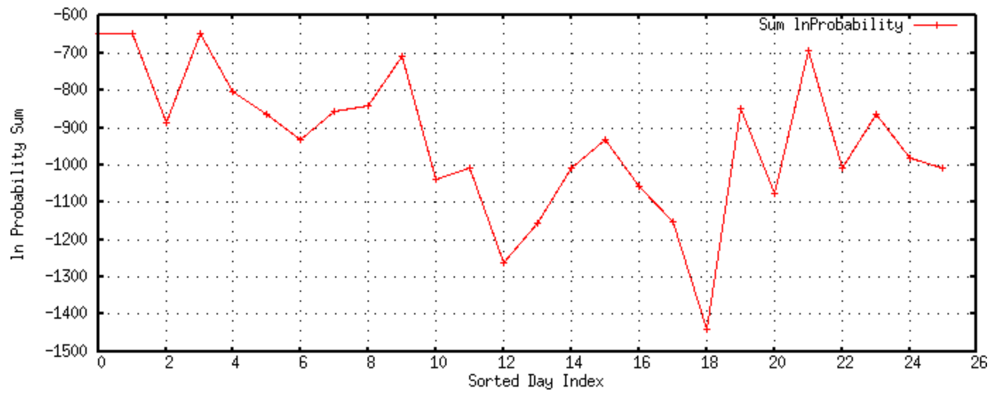


Figure 5.9.: Summed logarithm probabilities for sorted day index

To evaluate, if an observation sequence is considered normal or abnormal, we look at the standard deviation of the logarithm value between the best sequence and an observed sequence. In figure 5.10, we have an overview of the standard deviation, the mean logarithm probability and the activity level in percent.

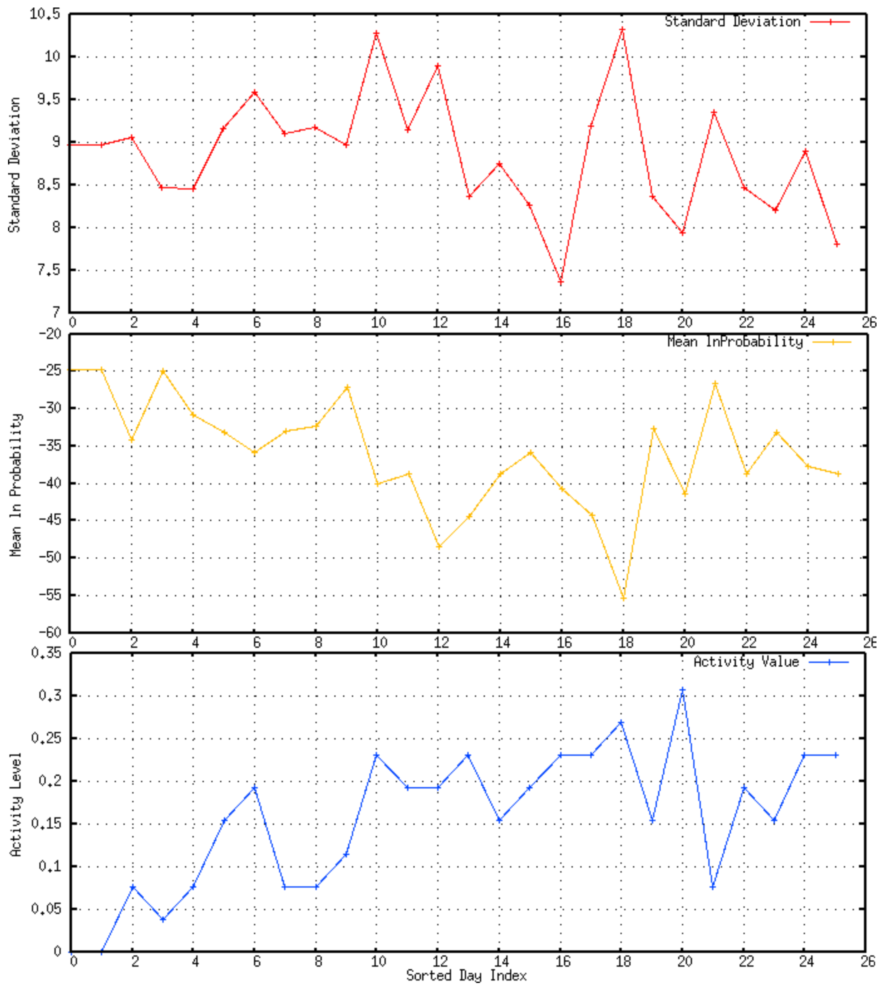


Figure 5.10.: Standard deviation (top), mean (center) of logarithm probability comparing days and activity level[%] (bottom) for sorted day index

Day 0 and 1 are both zero activity days, so we focus on the best active result day 3, which has the best logarithm mean value of −25 and a standard deviation of 8.49 on an 0.045% activity level. This accounts for a logarithm value range of −16.51 to −33.49. Bruckner et al. consider observation sequences, which fall into the stated logarithm probability range normal sequences, otherwise they are abnormal sequences, which shall indicate abnormal daily activity.[73]

5.6. Discussion

In the previous sections, we discussed the theory of the Hidden Markov Models including the Forward and Viterbi algorithm. We used sensor data recorded by Kasteren et al.[67], whereby we did not concentrate on one particular sensor, but rather combining all person induced readings into one to build our HMM. Then we reduced the state count and found the best standard day of the inhabitant according to the procedure introduced by Bruckner et al.[73].
After skipping empty activity days, we interpreted the sorted (original) index day 3 (7) as our standard default living day. Figure 5.11 shows how this translates into acceptance or rejection of the observation sequence for each day of our set.

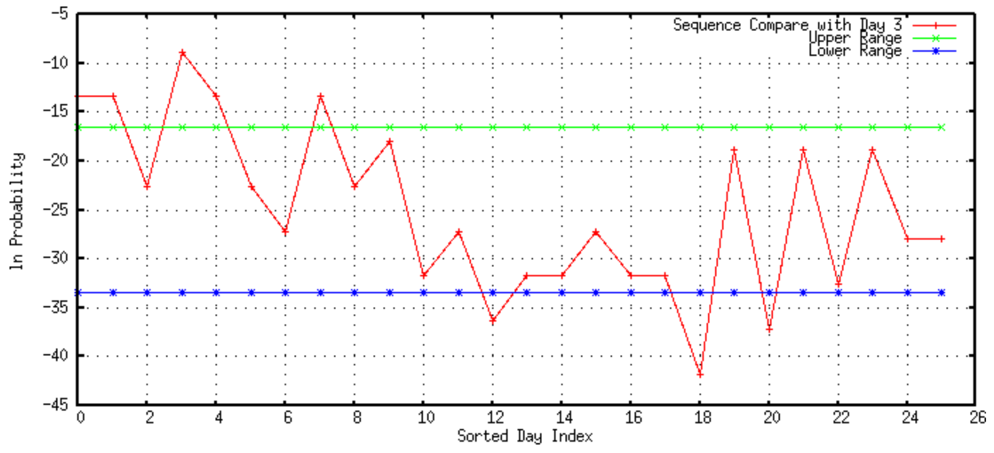


Figure 5.11.: Comparing all other days observed sequences with sorted index day 3 most likely state sequence

If we naively have a look at the rejected observation sequences, we will see some interesting details. Firstly our days without any activity are below the threshold, also the days 12 (12), 18 (19) and 20 (10) do have obvious different

activation patterns. Day 12 got some unusual readings in the morning hours, day 18 afternoon hours are also quite unique and day 20 is overly active between 20pm and midnight. Our training day 3 will also be rejected due to the fact that its logarithm probability value is too good to fall into the range. Day 4 and 16 both share the fate of the inactive days as these feature little motion after all.
Despite the apparently not too bad result of this rather simple approach, there is plenty of room left for improvements. Some ideas might be to consider external and environmental influences, like the weather, the weekday or some other related events. Bruckner et al. improved the state merging by also taking the backward direction into account reducing the state count even more. They also try to decipher specific paths of the model to make more specific assumptions about the actions of the user.[73][78]

Part III.

Implementation

6. Software Design

The reader was introduced into the tablet supported AAL context, state-of-the-art technologies and the requirements are listed in section 3.1. This chapter describes the design process in line with the previously stated requirements build in the environment of chapter 4, to visualise and control sensors and actuators of a smart home on an Apple Inc. iPad as an iOS App remotely over internet connectivity, whereby the design shall pave the way to be AAL middleware agnostic.
There are many different approaches when it comes to software design. For this thesis the actual design has begun actually earlier than this chapter analysing the environment and their stakeholder (Section 1.1.1.4) like the caretaker and caregiver, as well as the AAL system and finding the requirements (Chapter 3) based on the analysis.
This chapter will elaborate how to design the software to comply with this requirements.

6.1. AAL functionality as iOS Mobile Application

The high acceptance of tablet devices as introduced in section 1.2.1 and their capabilities supporting feature rich but still easy to use third party apps leads to a perfect match implementing AAL functionality. Apps are easy to deploy and update on a myriad of iOS devices supported by the AppStore. Other non iOS devices feature very similar mechanisms, even accompanied by lower entry barriers, see 4.1.2 for details on the hurdles.

6.2. Mobile Access Considerations

Mobile access is an important part of the software design, as it will e.g. influence the delay of messages and it also has a strong influence on the costs of the mobile data application in case of unnecessary protocol overhead. The delay topic is tackled by a clever message bus design and a lightweight protocol is used to minimise the data overhead on the line.

6.2.1. AAL Communication Bus Design - Live Updates

Home automatisation bus drives the communication between the AAL middleware and sensor/actuator devices. Messages of this myriad of different devices are unified and persisted, eventually further processed, within the middleware. A triggered sensor event passes so the bus, the middleware and shall be available on the mobile application, best without any major lag. The next section introduces a design decision based on the given nature of home bus middleware concept.

6.2.2. Connectivity Options

The environment connectivity section 4.4 elaborates all current available mobile access technologies. Its now up to decide which to use and why it is reasonable to do so.

Option 1 REST / Comet The first idea to design such a mobile access is implementing a simple REST capable server module which ties into the database service and messaging bus of the middleware. This is surely a viable approach, but comes with a technical major drawback. To mimic live updates either updates are received by polling of the mobile app against the REST server, or to keep the connection open for an extended period of time to allow some sort of push notification.

Option 2 Web Socket The second idea is more in line with the nature of a bus driven system. Web sockets[59] are a perfect match to any message bus driven architecture. A web socket server needs to be available on the middleware side, allowing forwarding of any live sensor bus messages to the web socket bus. Any registered end device connected to this web socket will be able to receive and process any message without significant lag, other than mobile networking induced delay.

6.2.3. Final Communication Design

Due to the advantages of option 2 the web socket approach is the more advanced than option 1, but still to be remarked option 1 is still a viable solution in some cases if the chosen middleware is not to be easy adapted to the web socket approach or an already existing REST server is in place.
The final communication line will look like the following. We have 4 main participants, first the AAL home bus, on the hardware level, which is tied together in the next one, the AAL middleware, which in turn manages and listens to the home bus, providing a unified event bus. Finally there needs to be a member of the event bus who manages our last participant(s), the mobile client(s). This shall be a web socket server which, if a client is connected and registered to the server, sends any matching event message to the WebSocket and thus to the mobile client.

6.3. Mobile User Interface Design

The mobile user interface design is quite different in contrast to the classical personal computer interface design. Due to the result of some studies which showed that understanding content on the mobile device is more challenging than on the classical desktop computer. The users focus lies on his or her fingers instead of the classical mouse / keyboard approach.[79]
Also the dynamic nature of an tablet needs to be considered as the tablet may

be physically hold in every direction urging the user interface to be adaptive. Svanæs at al. describes, after conducting several user centric low-fi prototyping workshops, that mockup prototyping does not require special competences, but stronger constraints on the technology are helpful and even necessary to design solutions for the given scenarios or requirements.[80] This method has been used to design all mockup designs for the mobile application using the very convenient balsamiq wireframe software[1], constrained to use most iOS provided user input elements. Like the use of so called *popover* elements, which are the mobile pedant to well known pop-ups, but usually implemented with a reference point and *table views*, which are as the name suggest tables consisting of rows and cells but often only one structural column. Using them consistently allows a unified user experience, and a fine grained level of information in conjunction with the drill down table view pattern. Drilling down is a simple navigation between one or more table views like sketched in figure 6.1, we dig one level deeper into the content by touching one table cell.[81]

[1]balsamiq URL: `http://balsamiq.com`

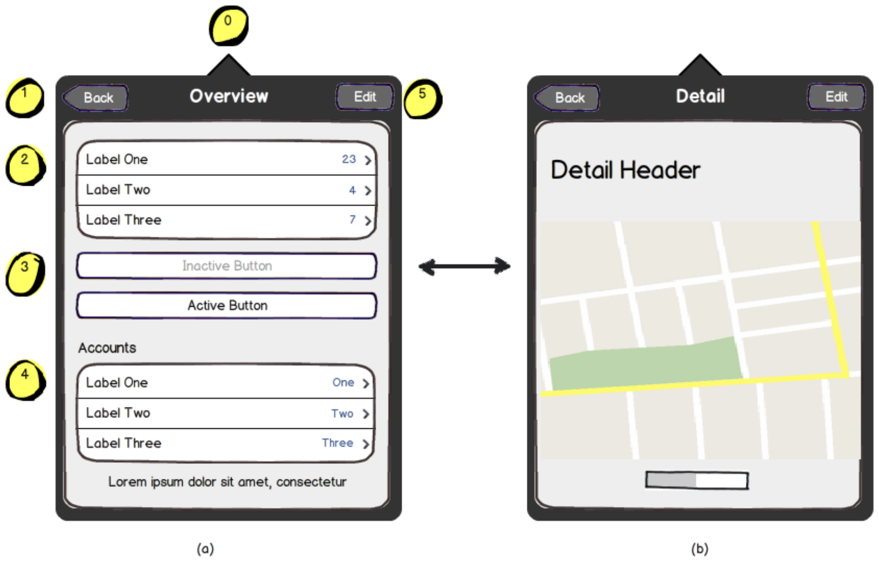


Figure 6.1.: *Mockup* Popover example:
(a) (0) Reference point, (1) Navigation, (2) TableView, (3) Buttons, (4) Grouped TableView, (5) Navigation Edit Mode Button
(b) detail view after touching one of the tableview elements.

Functionality wise it must cope with the following requirements stated in section 3.1.

6.3.1. Management of more than one Flat

The following mockup figure 6.2 shows the principle design idea of managing more than one flat by representing each flat as a tile ordered in an array. A flat representation consist of the common name of the flat and a scaled down floor-plan. This view may be extended to some sort of traffic light guided one stop information screen where warnings and alarms are being collected for birds eye overview experience.

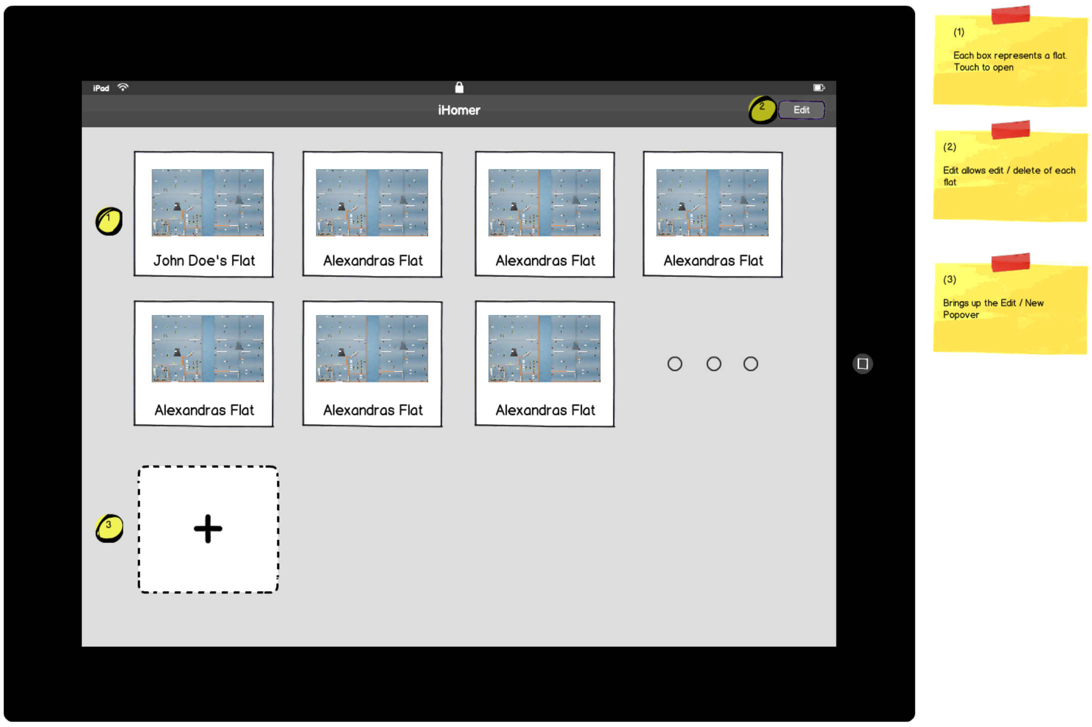


Figure 6.2.: *Mockup* Flat selection)

6.3.2. Visualisation of the Flat

The visualisation is designed to start from the natural floor-plan down to each room including sensor and actuator pictograms. Doors are visible as well for easier reading of the plan. Each element is designed to be touchable for further informations. Mockup figure 6.3 shows the flat view scaffold with one kitchen light context dialog open. The top bar is used for flat level navigation and view preferences and a collected device battery overview. The heart icon is designed to collect a list of favourite devices for fast access. A person icon (3) shows the latest occupancy spot of the person living in the flat, whereby the fainted version (4) marks the previous room of occupancy. This live occupancy tracking is linked to the live update sensor event feed and updated every time a sensor is triggered ultimately by the person in the flat.

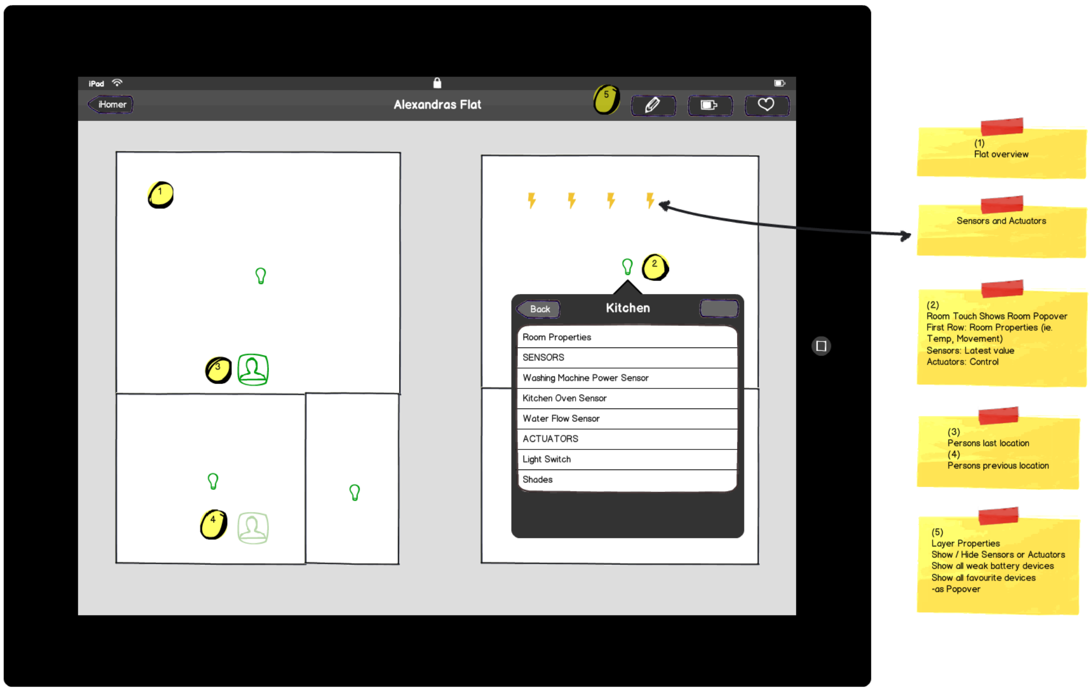
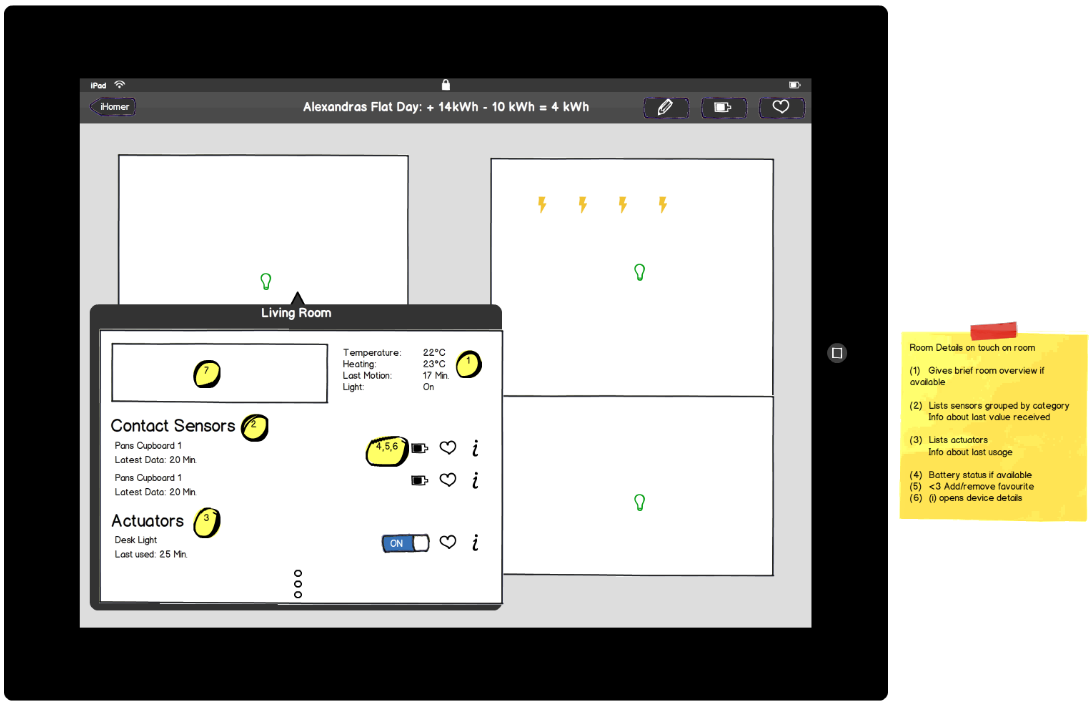


Figure 6.3.: *Mockup* Flat view)

Touching a room will open a so called popover which gives an overview list of all room specific sensors and actuators. Every sensor shows some extra functionality like latest activation and the battery state, favourite switch, and an info button for more details. Actuators are easily control-able by the according user interface element. An switch will be shown as ON/OFF element, a dimmer as a slider object.



Room Details on touch on room

(1) Gives brief room overview if available

(2) Lists sensors grouped by category
Info about last value received

(3) Lists actuators
Info about last usage

(4) Battery status if available
(5) <3 Add/remove favourite
(6) (i) opens device details

Figure 6.4.: *Mockup* Room details)

6.3.3. Visualisation of Sensor Values

The visualisation of sensor data, especially physical occupant induced events, but also other like temperature data, brightness progression are primary concern for the AAL function. There are plenty of different sensor types and techniques are to be supported. Different graphing representations are necessary, depending on kind of generated data. Unfortunately the iOS platform comes with very little build in charting capabilities, but some third party efforts are made to fill the gap, some iOS specific implementations and other web – JavaScript – based libraries. Open source native implementations are unfortunately not that feature complete and turnkey ready as their commercial pedants. However, with the advancing of web technologies over the last decades web-based libraries gained some traction and are also bearing an other big advantage of native libraries is their platform independence, which is in line with the idea to pave the way of easy porting of this project to other operating systems. We will use the *HighCharts* library for this project as it is simple and flexible to use, but still having the component approach in mind switching to another library should not break substantial parts of the application later on.

Native Libraries

Core Plot	Albeit the name, its an open source project (New BSD License)[2] and actively maintained. It provides simple 2D plotting and is integrated with Apple technologies like Core Animation, Core Data and Cococa Bindings.[3]
iOSPlot	Simple open source, unrestricted licensed, line charts and pie charts library, not actively maintained.[4]
iOS:CHART	Commercial charting library for iOS and OS X. 2D and 3D capabilities.[5]
KeepEdge Library	Commercial charting library for iOS. 2D and 3D capabilities.[6]
Shinobi Controls	Commercial charting library for iOS and Android OS. Most feature and usage complete implementation.[7]

Web-based Libraries

HighCharts	paid license for commercial project and Creative Common Attribution-NonCommercial 3.0 License[8] for personal/education and non profit use. Rich plotting and chart-

[2]New BSD License URL `http://opensource.org/licenses/BSD-3-Clause`
[3]CorePlot URL `https://code.google.com/p/core-plot/`
[4]iOSPlot URL `https://github.com/honcheng/iOSPlot`
[5]iOS:CHART URL `http://www.threedgraphics.com/tdg/products/tools/ioschart/`
[6]KeepEdge Library URL `http://www.keepedge.com/products/iphone_charting/`
[7]Shinobi Controls URL `http://www.shinobicontrols.com`
[8]Creative Common Attribution-NonCommercial 3.0 License details: URL `http://creativecommons.org/licenses/by-nc/3.0/`

	ing capabilities on cross-platform based on SVG graphic technology: iOS, Android, Windows, OS X, Linux.[9]
D3Js	Feature rich open source BSD licensed drawing library based on SVG graphic technology, does not only charts, but can visualise virtually anything. Cross-platform compatible: iOS, Android, Windows, OS X, Linux.[10]
Other	There are plenty of other libraries available as of today. The interested reader may find a comparison of the most popular at socialcompare.com[11]

We elaborated the different sensor types in the environmental section 4.2.3. Table 4.1 lists most common used sensors, whereby the first group of contact sensors are firing if the specific sensor had been triggered by whatever the sensor is watching for, like person presence, water, smoke and other. This data points are to be considered as events on an x axis in a time series view. The other group of metering sensors are usually reporting measured values either timely on regular basis, manual or triggered by any other means. The visualisation needs to be aware of the type of graph to be presented based on the sensor properties like y axis labelling.

6.4. System and User Notifications and Messaging

There are several different types of notifications in an AAL environment. These notifications are derived of various events or the presence of such in a certain order or constellation. Not every message is of the same importance to the user, in respect to that needs each message to be different presented, depending on the classification. The following table 6.1 gives a brief overview of each message class and their description and representation.

[9]HighCharts URL `http://www.highcharts.com`
[10]D3Js URL `http://d3js.org`
[11]Comparison of JavaScript graph and charting libraries URL `http://socialcompare.com/en/comparison/javascript-graphs-and-charts-libraries`

Message Class	**Description**	**Representation**
Information	Information styled messages are in a wide ranging field of applications, and getting notified if a certain event, i.e. the person is waking up in the morning, happened, or a particular sensor level value is reached.	Simple popup message, to be manually or automatically, after a certain time frame, dismissed
Warning	Warnings are similar to alarm typed notifications, but without the immediate need for action, i.e. a not switched of light after leaving the room for x minutes.	Yellow highlighted message style, visible until manual confirmation or automatic hidden after a certain time frame.
Alarm	An alarm is triggered if there is some event of immediate urgency registered, like fire detected, or a fall sensor is triggered	Red highlighted message style, needs to be manually confirmed
Silent	Silent messages are simple system events like the observation of temperature change or a room change event recognised by the sensors	Short highlighted simple message, dismisses automatically after a certain time frame.

Table 6.1.: Overview of various AAL system generated message classes

6.4.1. Usabilty

“The old computing was about what computers could do; the new computing is about what users can do.

Ben Shneiderman

As pointed out by Bernd Shackel et al. usability is a product of ease of use and efficacy in terms of (human) performance. They proposed a formal definition of usability of a system or equipment as the following:[82]

> “the capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support, to fulfil the specified range of tasks, within the specified range of environmental scenarios”.[82]

Usability is a quite extensive topic on its own, for the range of this thesis we try to provide a easy to use interface, but we are not focusing on assessing the performance of the usability. The usability design for this thesis is aligned to the *Apple Human Interface Guidelines* document[83] as this guide reflects all relevant do’s and don’t for developing a simple mobile user interface.
Beyond that several consideration are made to improve the user experience. Avoiding unnecessary outputs and focus on the viable parts are of great concern. Popovers explained in section 6.3 are a great way to hide away unnecessary clutter by providing context based information.
If the AAL environment got a high count of sensors and actuators the flat view will be very cluttered. But some sensors may be of less interest than others, therefore favourites are introduced. Each sensor may be easily flagged as favourite and the view can be adjusted to just display favourite devices. Besides that filtering by device groups like contact sensors, or lights and both are all intuitively

filterable in the flat view options. An extra battery popover collects all devices ordered by battery charge showing near empty ones at the top, this allows a battery management even if the devices is currently hidden in the flat view. In addition to that are soon or near empty devices are highlighted in the view to assist in locating the battery to be replaced.
Another factor of increasing the human performance is abstracting as much things as possible out of the sight of the user, this is achieved by handling any back-end connectivity automatically as required. There is no need to manually connect or disconnect to the server at all. Same applies to saving and persisting the user state, there is no need to manual hit save.

6.4.2. Control of the Smart Home

Actuators are devices which control home appliances like shades, lights and a myriad of other devices in a smart home. Actuators themselves are controlled by control messages on the home bus. But first we need to provide a way to find the actuator the user want to control. This can easily achieved by the floor plan and room plan where each actuator is being shown on their particular location in the flat. Touching a actuator will open a user interface tailored for the particular type of actuator, which may be a simple ON/OFF element, or a dimmer element controlled by a slider interface. Selecting a new value for the actuator will trigger a control event on the Web Socket bus, which is then on the back-end side translated into a home control command understood by the particular device.

7. Implementation

The reader was introduced into the environment of the AAL field and current tablet devices and technologies wiring them together. In the previous *Software Design* chapter 6 where several key design decisions taken as well as state of the art technologies elaborated.
This chapter will now explain the software implementation process for both, the OSGi modules on the server side and an iOS application good for the Apple AppStore. For this process the used software design patterns, as well as the implemented classes and the communication between back-end and mobile application are explained. This does not cover any deep language insights, but some pointers to the quite comprehensive free Objective-C documentation are provided in the bibliography and footnotes.

7.1. Software Design Patterns

Cristopher Alexander et al. says:

> "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".[84]

Whilst Alexander et al. where talking about patterns in buildings and towns, the – so called – "Gang Of Four" [1] used this very definition as foundation for

[1] Their book "*Design Patterns: Elements of Reusable Object-Oriented Software*" is to long to read so it got abbreviated to the book of the Gang of Four, namely: *Gamma, Erich*

their object oriented software pattern definition. They split a pattern in four essential elements:[85]

1) **Pattern Name**	A name to describe the problem the pattern solves
2) **Problem**	Describes when to apply the pattern and the context
3) **Solution**	Depicts the elements that make up the design and relationships, responsibilities and collaborations, but not a concrete design or implementation.
4) **Consequences**	Details any trade-offs like systems's flexibility, extensibility or portability and results of applying the pattern.

Based on this definition we are going to elaborate pattern used for this thesis.

7.1.1. Model View Controller

The Model View Controller abbreviated as MVC pattern was firstly used to build user interfaces in Smalltalk-80[2]. It consists of three parts, the model represents the data representation, the view as screen representation and the controller takes care of user or any other input by updating the model. Figure 7.1 gives an idea of the traditional approach to MVC. In this approach there is no direct connection from the view to the controller, but rather the view changes the model, and the controller gets notified by the change of the model in some way. [87][88]

and Richard, Helm and Ralph, Johnson and John, Vlissides.[85] See `http://c2.com/cgi/wiki?GangOfFour` for details.

[2]A dynamic object-oriented programming language; The interested reader may find the introduction of Goldberg et al. [86]

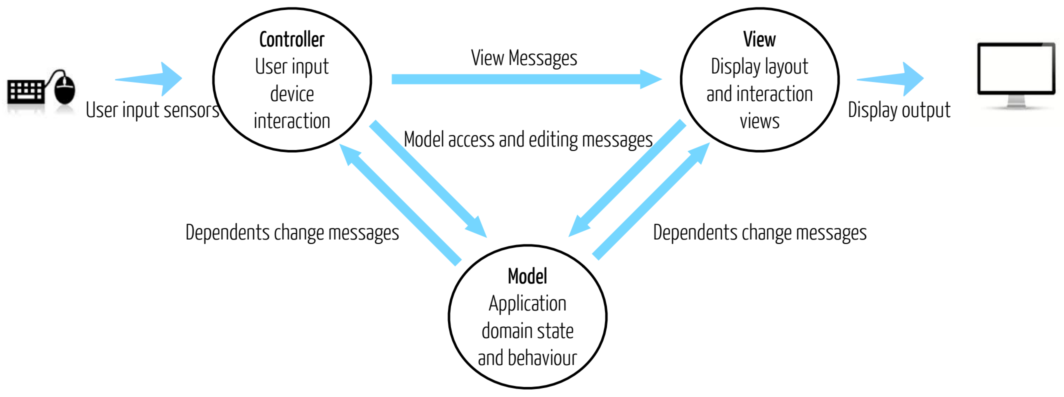


Figure 7.1.: Model View Controller State and Messages Sending
Author adapted of source: Goldberg et al. [88]

Lately this approach, separating the view and the controller, is softened by many developers and the controller mediates between the view and model alike. Apple Inc. uses notifications of state changes in model objects to communicate with view objects by the controller objects, which mediates the flow of data between the model and view bidirectional. A reason why the MVC pattern is so important is that it allows highly reusable model and view objects whilst application logic is concentrated at the controller modules.

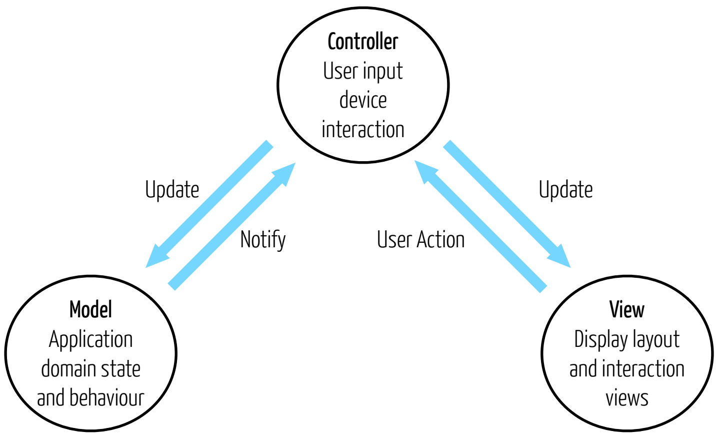


Figure 7.2.: Model View Controller Cocoa Version
Author adapted of source: [89]

In the iOS environment MVC is accompanied by the *Cocoa Bindings* technology. Which is a collection of technologies that allows a reduction of code dependencies between views, models and controllers. It automatically synchronises views when the bound model data changes and eliminates the otherwise necessary glue code by using the *Interface Builder*[3] to connect the the model – view – controller components graphically.[90] For a deeper dive into this subject be referred to the *Concepts of Objective-C Programming* guide: [89].

The mobile application uses the MVC pattern throughout the application for every view component. This application design is dictated by the Cocoa framework provided by Apple Inc. Most views therefore are build graphically using the *Interface Builder* and packed as `*.xib` files. These are XML files containing the representation of the graphical edited view. These `*.xib` files are accompanied by usually one controller class of the MVC pattern. Most controllers utilise the `-(void)viewDidLoad` method to initialise the view component and to make adjustments to the layout of the view which are not possible in *Interface Builder*.[89][90]

7.1.2. Singleton

The singleton pattern ensures that a class has one instance and is globally available, which is important for the logic in some cases.[85, p.144][91] We discussed in section 6.2.2 the connectivity options and decided us for the *WebSocket* approach. To manage the *WebSocket* communication we create a singleton instance of the `MGEventBusController` to take care of any communication between the back-end and mobile application. A singleton design is of advantage due to the natural fact of having exactly one bus data exchange. Source listing 7.1 shows the static method to retrieve a singleton instance in Objective-C which is then called like this: `[MGEventBusController sharedInstance]`.

[3]Interface Builder is a module of the Xcode integrated development environment software to design and link view elements with controlling code

Listing 7.1: Objective-C singleton retrieval method

```
 + (id)sharedInstance                          // '+' sign: static
{
    static dispatch_once_t pred = 0;           // predicate
    __strong static id _sharedObject = nil;    // declare static
    dispatch_once(&pred, ^{                    // only called once,
        _sharedObject = [[self alloc] init];// initialise object
    });
    return _sharedObject;                      // return singleton
}
```

7.1.3. Observer Pattern

The intent of the *Observer Pattern* is to define a one-to-many relationship between objects, if the one side object changes its state the many objects get notified and are able to take action on the change. This is a powerful way to maintain abstraction between two or more objects.[85, p.326]

Objective-C embraces two different build in ways to implement the *Observer Pattern*, the first Key Value Observing allows to add an observer to nearly every class for a specific key path, whereby the key path states the member attribute to be observed. A drawback of this implementation is that the broadcaster does not know who is listening, its also crucial that methods must follow the naming conventions otherwise the the observation messages will not work. For this thesis the KVO pattern is used for the `MGFlatViewController` to track configuration changes. Details about the KVO Objective-C implementation are found in the *Key Value Observing Guide*[4].[92]

The second build in approach is the `NSNotificationCenter`, a *Singleton* which is a even more decoupled way of the *Observer Pattern* than KVO. Any object can send a notification to the center, and any object may register

[4]KVO Guide: `https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/KeyValueObserving/Articles/KVOImplementation.html`

to the center to listen for one or more specific notifications. The first hand advantage is that the sender nor receiver needs to know about the other. Listing 7.2 shows the code to register to the `NSNotificationCenter` by calling the `-(void)methodCalledOnNotification:(NSNotification*)aNotification` method having the `NSNotificationObject` as parameter. The different messages are distinguished by the `name:` parameter which takes a simple string as input, however, its a good idea to use a constant string object for identification to avoid typos and cluttering of inline string object definitions.[93]

Listing 7.2: Objective-C `NSNotificationCenter` Observer a Notification

```
[[NSNotificationCenter defaultCenter]
    // observer class instance
    addObserver:self
    //method of observer to be called
    selector:@selector(methodCalledOnNotification:)
    //static notification id string
    name:MGMessageDidUpdateNotification
    //optional attached object
    object:nil];
```

Listing 7.3 depicts the source to post a specific notification to the center.

Listing 7.3: Objective-C `NSNotificationCenter` Post a Notification

```
[[NSNotificationCenter defaultCenter]
    postNotificationName:MGMessageDidUpdateNotification
    object:objectToAttach];
```

For this project notifications are used to broadcast any *WebSocket* related messages through the application. This allows future components easily to register for any AAL specific event independently of existing listeners. However, a word of warning at this place, do not try to stuff all inter object messaging into the `NSNotificationCenter`, there are other patterns which may suit this task even better like delegates.

Delegates are also used for the observer pattern. A delegate object is weakly[5] referenced by a host object and the hosts sends messages to the referenced delegate object. Protocols are often used in chorus with the *Delegate Pattern* to define the messages a delegate object needs to understand either mandatory und optional. In the Java programming language a protocol is the pedant to a Java Interface declaration. Code listing 7.4 depicts a sample protocol definition which must be implemented by the `delegate` object. [94]

Listing 7.4: Objective-C Delegate Protocol: `MGConfigureFlatViewController.h` File

```
...
//delegate property definition
@property (nonatomic, weak) id<ConfigurationAddDelegate>
    delegate;
...
@protocol ConfigurationAddDelegate <NSObject> //extends NSObject
- (void)configureFlatViewController: //mandatory method
   (MGConfigureFlatViewController *)configureFlatViewController
   didAddConfiguration:(Configuration *)configuration;
@optional //declares optional methods
- (void)optionalMethod;
@end
...
```

[5]weak means in this context its a simple pointer reference without retaining. See `https://developer.apple.com/library/ios/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/EncapsulatingData/EncapsulatingData.html` for details

7.2. Interfacing AAL Middleware

We evaluated current state of the art options of the underlying interfacing technology in section 6.2. Now we are going to explain the actual implementation based on the evaluation result *WebSockets*. The implementation is split in a server part living in an OSGi bundle on the server side and on the mobile application side in the `MGEventBusController`. The chosen AAL framework H.O.M.E.R. already provides a Homer Core Eventbus bundle which takes care of basic *WebSocket* support based on the Vert.x[6] server. Every *WebSocket* capable client can connect to this Eventbus bundle, using the correct address the server is listening on. The default address for the Homer Core Eventbus bundle is `http://ip.of.server:1234/eventbus`, please be aware this is subject to change and has to be verified with the back-end. `1234` is the port and `eventbus` the path the server component is listening on.

After a successful connection we need to register the client to the event bus in order to be able to receive messages targeted to this particular address. This is achieved by sending a JSON formatted message to the connected *WebSocket*:

Listing 7.5: JSON Message Register Eventbus

```
{
  "address":      "<generated UUID-1>",
  "mode":         "register",
  "replyAddress":"<generated UUID-1>",
  "senderId":     "<generated UUID-1>"
}
```

The address, replyAddress and senderId are all populated by a randomly generated identification string. To guarantee a unique identification a *Universally*

[6]Vert.x is a lightweight, high performance application platform having a Distributed Event Bus as backbone. See `http://vertx.io` for details.

Unique Identifier UUID[7] is used. This identification is then registered at the event bus as address, ready to receive messages.

There are two other services registered to the event bus, firstly the `HomeControlService.java` which is capable of rendering the complete flat data structure as JSON message and to control actuators, as well as posting actuator state change events to the web socket bus. This service is already provided by the chosen AAL middleware.

Secondly a new OSGi bundle, analogous the previous service, is needed to cope with several home bus event related commands:

REQUEST_LATEST_EVENT

Queries the logged event database for the latest recored sensor data. This data will be needed to initialise the mobile app view.

REQUEST_MESSAGES_FOR_SENSOR

Returns all recorded sensor values for the given time period and specific device identification.

REQUEST_SENSOR_DATA

Similar to the previous command, but different JSON formatting to optimise for the charting engine.

REQUEST_ENERGY_DATA

Same as the previous command, but energy consumers and generators data compiled for charting.

REQUEST_LATEST_MOTION

All recorded motion data generated by usage, motion and person activated devices compiled. This data is also needed for initialisation of the motion view.

[7]Standardised by the Open Software Foundation as part of the Distributed Computing Environment, to identify objects in a decentralised system uniquely without a central registration system. Details are found at `http://tools.ietf.org/html/rfc4122`

REQUEST_ACTUATOR_REQUEST_EVENT

Requests if possible a actuator device status message, which can contain information about the current state of the device, like if it is currently switched on or off. Used to initialise the user interface for actuators.

REQUEST_CURRENT_ACTIVITY_CONDITION

Request the current activity state of the user, which is calculated based on Hidden Markov Model described in section 5.5.1. This information is used for warnings on the screen if the activity is not confirming.

Each command may be accompanied by an `integer value`, an `integer deviceId`, milliseconds formatted `startDate` and `endDate` as `long` datatype. These commands are processed by `EventControlService` which is also capable to forward any home bus sensor and state event message to the web socket bus. This happens by publishing messages to the address defined in the `EventControlService.java` class. Every event bus client registered to this particular address will receive live updates of all home bus events and state changes.

7.2.1. The Mobile Device as WebSocket Client

We now have discussed the provided middleware web socket technology and implementation by listing all required services and commands to bring vital information on the WebSocket. iOS does not provide a WebSocket implementation out of the box, but there are two mayor third party WebSocket libraries available as of today. SocketRocket[8] provides an easy to use WebSocket client library conforming the RFC 6455[59] standard. The other option would be the new CoreWebSocket library[9]. Both of them are valid candidates but for this thesis we used the SocketRocket implementation due to its more mature nature.

The `MGEventBusController` acts as a singleton instance managing one or more WebSockets provided by the SocketRocket library. According to the middleware specification the controller registers itself to the event bus and then to the

[8]SocketRocket details see GitHub: `https://github.com/square/SocketRocket`

[9]CoreWebSocket details see GitHub: `https://github.com/mirek/CoreWebSocket`

`HomeControlService` and `EventControlService`.

7.3. Class Dependency Graph iOS

The following class dependency diagram depicts any inter class dependencies and an architectural overview of all classes. Only custom view implementations are shown, as most view objects are build by the *Interface Builder* in a graphical way. Custom view implementations grant a deeper level of control like the `MGFlatView` which is in charge of translating `Coordinate` objects into a graphical representation of the flat. The `main` class is the application entry point which instantiates the `MGAppDelegate` singleton. Details about the most important classes are to be found in section 7.6 later in this chapter. Classes like [`Flat,Room,Device,...`] are considered to be domain objects, whereby their dependencies are omitted in the graph in favour of a better architectural overview.

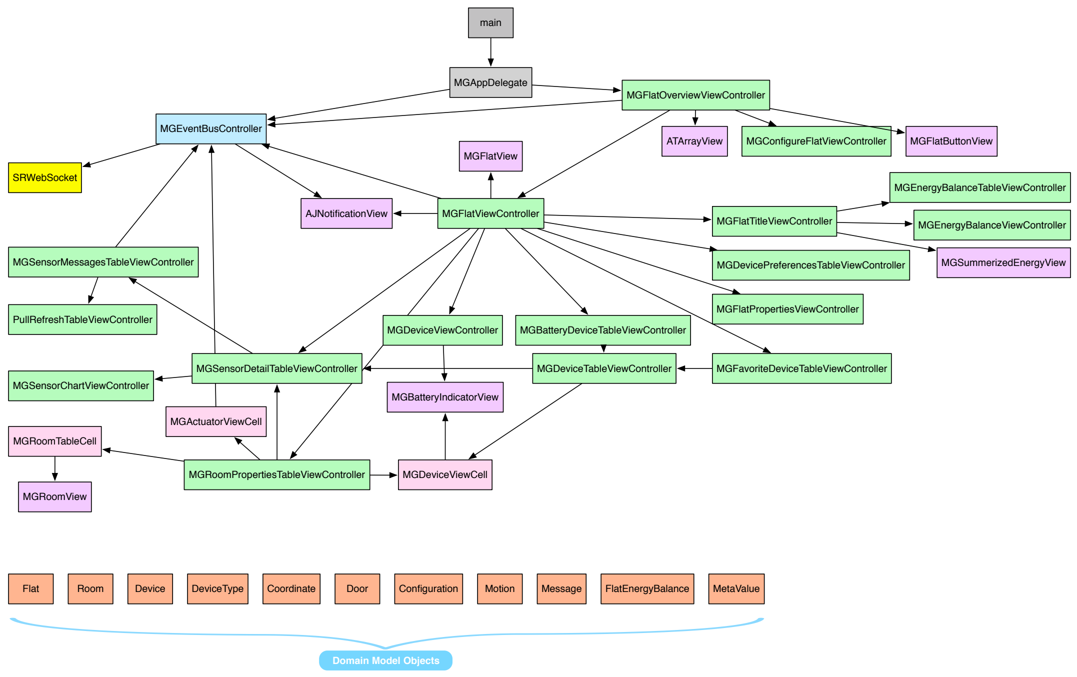

main
MGAppDelegate
MGFlatOverviewViewController
MGEventBusController
ATArrayView
MGConfigureFlatViewController
MGFlatButtonView
MGFlatView
SRWebSocket
AJNotificationView
MGFlatViewController
MGEnergyBalanceTableViewController
MGFlatTitleViewController
MGEnergyBalanceViewController
MGSensorMessagesTableViewController
MGDevicePreferencesTableViewController
MGSummerizedEnergyView
PullRefreshTableViewController
MGFlatPropertiesViewController
MGDeviceViewController
MGBatteryDeviceTableViewController
MGSensorChartViewController
MGSensorDetailTableViewController
MGDeviceTableViewController
MGFavoriteDeviceTableViewController
MGBatteryIndicatorView
MGActuatorViewCell
MGRoomTableCell
MGRoomPropertiesTableViewController
MGDeviceViewCell
MGRoomView
Flat
Room
Device
DeviceType
Coordinate
Door
Configuration
Motion
Message
FlatEnergyBalance
MetaValue
Domain Model Objects

Figure 7.3.: iOS Mobile Application Dependency Graph

7.4. Domain Objects in iOS

A domain object represents a digital version of a real world object, which are used as a data storage as well as live data cache to optimise startup performance. The domain model is implemented by the integrated Xcode data model editor, that being a user friendly graphical user interface to tinker with `*.xdatamodeld` folder type that contains a simple XML based file containing all data object relevant information like name, attributes, relations and restrictions. Its not recommended to edit this `*.xdatamodeld` structure manually. This definition is then used by the iOS CoreData framework to build the CoreData back-end providing a fast and convenient object store.

The following figure 7.4 shows the implemented domain objects, in *Xcode*, stating their name at the top of each block, their attributes within and their relations are defined by the lines between the boxes. A single arrow symbol depicts a to one relation, whereby a double arrow indicates a to many relation like for example the “Room $< - >>$ Device” relations reads as following: A room may have many devices, but a device belongs to one room. An “A $<< - >>>$ B” notation denotes a many to many relationship, saying one of A may have one or more of B and vice versa. Details about the CoreData domain modelling may be found in the *Core Data Programming Guide* [95].

`Flat`	Represents a real world flat and holds one or more rooms, whereby a room consists of one or more doors and zero or more devices. Other informations like the flats energy balance is also attributed to the flat object.
`Configuration`	Is tied to each flat and holds information for server connectivity and general user preferences for visualisation of the flat as well as the customised flat name.
`Room`	Represents a real world room, the coordinates are defining the two dimensional layout. Devices are either a sensor

	or actuator. It can have one or more doors. Room name and type are persisted in the MetaValue objects.
`Device`	Either a sensor or actuator device, assigned to one room at a certain location determined by the coordinate object.
`DeviceType`	Discerns between sensor and actuator with the category attribute and the general device type.
`Message`	A device message, usually sensor data. Also holds the information about the battery state. DayOfWeek is a helper attribute to easily accumulate weekday data. This saves time on processing.
`MetaValue`	Generic meta values like name and other for a room and device object.
`Coordinate`	2D coordinates. One for each device, many for each room, each coordinate depicts an edge of the room walls, whereby the order attribute is a simple numeric value to assess the right sequence of coordinates.
`Door`	Simple door object, just for visual guidance.
`Motion`	All motion relevant sensor events compiled per each room, to identify the occupants movements.
`FlatEnergyBalance`	Simple composite object to keep track of the overall energy balance of the flat.

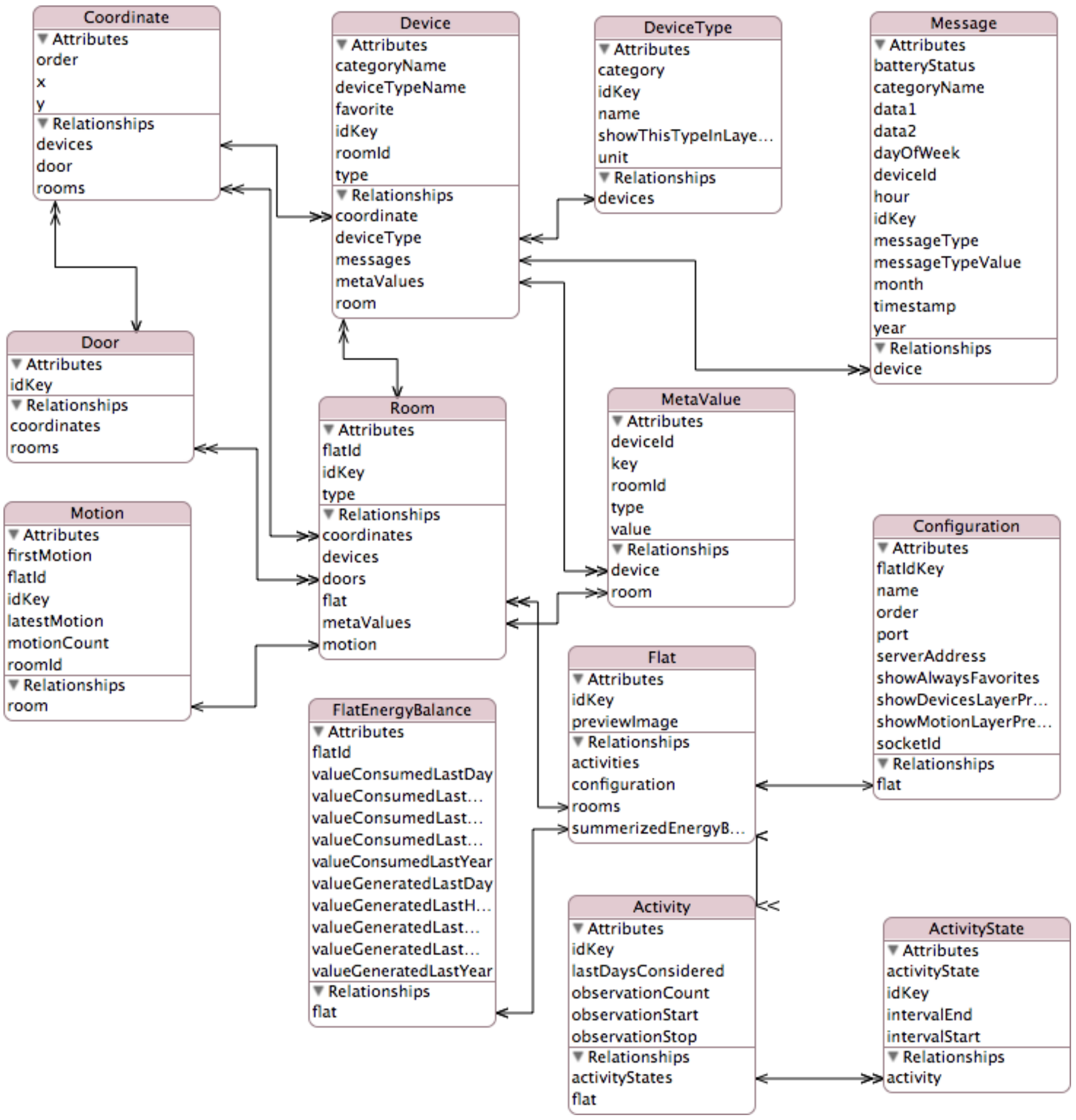


Figure 7.4.: iOS CoreData Domain Model

7.5. Domain Object Mapping

To be as AAL middleware agnostic as possible are all domain objects mapped between their back-end name and the CoreData objects. The third party library RestKit is actually, as the name suggests, for REST implementations developed.

It aims to make interacting with RESTful web services simple. For performance reasons it is also able to cache any data locally in the CoreData space.[57]
The first prototype used the full feature set RestKit provides, but as the WebSocket technology evolved over time we just use the CoreData mapping feature. Specifically this is done by accessing the `RKMapperOperation` class directly feeding in the parsed JSON feed string as a `NSDirectory` holding nested key value pairs. The `RKMapperOperation` works in line with the CoreData context adding or updating managed object instances.
This also needs to have any mappings in place, which are set in the initialisation process after the startup of the mobile application in the `MGAppDelegate`. The basic idea is to map any declaration and relation of the JSON provided data to their CoreData counterparts. This is an advantage if an existing AAL middleware provides either REST or WebSocket driven interfaces to easily adopt the mobile client by just changing the mapping details. However, if there are substantial differences between the domain model of the mobile application and the provided domain model interface of the middleware, more plumbing is likely to be required in form of adapting the interface compliance. Details on how to map objects are found in the RestKit mapping documentation[10].

7.6. Class Descriptions

The reader was introduced into the application architecture showing the dependencies of all involved classes. To get a deeper understanding of the implemented classes, we will give a brief introduction of the important ones.

MGAppDelegate

This is a singleton instance and found in every iOS application. For our application the `AppDelegate` takes care of initialising the CoreData back-end and the RestKit library including the data mappings between RestKit and CoreData.

[10]RestKit Mapping Documentation: `https://github.com/RestKit/RestKit/wiki/Object-mapping`

By default it also implements the `UIApplicationDelegate` protocol[11] which defines a myriad of functions targeting the application state, remote and local notifications and more. For this application we only need the obligatory app state changes event function to determine when we need to persist the current application model to the data store.

MGFlatOverviewController

Visual entry point of the application, it presents an array of already configured flats shown as preview images of each flat. Allows to add / update / remove new flats by calling the `MGFlatConfigurationController`. The matching views are defined in the `MainStoryBoard_iPad.storyboard`.

MGFlatConfigurationController

Called by the previous `MGFlatOverviewController` to present a user interface to edit the flat connection properties.

MGFlatViewController

The very heart of the application. After a configured flat is selected in the `MGFlatOverviewController` an the details passed on to this, it takes care of presenting a visual representation of the flat using the `MGFlatView` including all devices. This representation is the central user interface point, touching a room or a device will open the a popover managed by the controller in charge.

MGRoomPropertiesTableViewController

In a popover context called by the `MGFlatViewController` its subclassed the `UITableViewController` which in turn provides the basic behaviour of an often used TableView implementation. The main purpose of this controller is to manage the content of every cell and to dispatch any other view content if the user is digging deeper into the popover content. The top cell shows a summary of the room parameters like temperature, last movement and activity information.

[11]Details about UIApplicationDelegate `https://developer.apple.com/library/ios/documentation/uikit/reference/UIApplicationDelegate_Protocol/Reference/Reference.html`

MGSensorDetailTableViewController

Also a TableView based ViewController which is intended to present a view for a specific sensor device. Also gives access to sensor history data either as raw values or in a chart based on the HighCharts[12] library.

MGEventBusController

Central communication controller, which holds an array of WebSockets, for every flat one to allow concurrent connections to different AAL middleware instances. It also handles proper initialisation procedures and dispatches any live bus communication to the `NSNotificationCenter`.

MGFlatPropertiesViewController

Manages the flat properties, this changes are reported back by the delegate pattern using the `FlatPropertiesDelegate` protocol. So the `MGFlatViewController` is able to react on any property change.

MGFlatTitleViewController

The title allows a quick information about the energy stats of the flat. This controller manages the displayed values and intercepts touch events to rotate the energy balance values by day, month and yearly time spans.

MGSensorChartViewController

This controller is used to manage the `UIWebView` which loads a HTML file containing the HighCharts library. It also passes the current sensor id to a in JavaScript defined function which in turn loads the required sensor data of the WebSocket bus.

[12]See section 6.3.3 for details about HighCharts

7.7. Used Third Party Libraries and Applications

“The standard library saves programmers from having to reinvent the wheel.

Bjarne Stroustrup

Third party libraries are widely used and are an generally accepted way to avoid reimplementing the same functionality again thus they may save precious development time and allow a focus on the project centric issues. Steven Raemaekers et al. define a *third-party library* as

> “... a reusable software component developed to be used by an entity other than the original developer of the component ...” [96]

We previously introduced already some core third party libraries but for completeness all used libraries are referenced in this section. Depending on the development language and environment, third party libraries are often managed by some sort of management software. Objective-C does not have any dependency system natively available as of today, but the Eloy Durán et al. community developed the widely accepted *CocoaPods*[13] dependency framework. The mobile application also relies on CocoaPods for dependency resolution, which are stated in the `Podfile` as simple '`pod "<DependencyName>"`' line statements, `<DependencyName>` to be replaced with the actual name. CocoaPods takes care of downloading and linking the dependency against to build process.

OHAttributedLabel

Developed by O. Halligon[14] and MIT licensed[97] Provides simple classes to provide styled text in simple labels. Used for the energy balance overview in the

[13]CocoaPods details see: `http://cocoapods.org`
[14]`https://github.com/AliSoftware/OHAttributedLabel`

title to dynamically colorise red and green within one label depending on the positive or negative value.

MagicalRecord

MagicalRecord is an add on library for CoreData. It is developed by S. Mora et al. and allows easy one line data fetching statements and in return making the CoreData code more readable and easier to maintain. It is used for almost all CoreData fetch requests. The interested reader is directed to the comprehensive documentation of MagicalRecord for further details.[15]

RestKit

We introduced and explained the usage of RestKit[57] in the Object Mapping section 7.4. RestKit provides several key features missing in the native Objective-C iOS frameworks like a simple high level HTTP request / response implementation, CoreData support, object mapping, database seeding and more. For this thesis RestKit is mainly used for its CoreData and object mapping support but in case the AAL middleware of choice does offer a REST interface, RestKit will allow an easy switch of interfacing technology.

AJNotificationView

A notice component developed by Alberto et al.[16] which provides easy access to a slide in notification view including queuing.

HighCharts

We compared the HighCharts library already in the *Visualisation of Sensor Values* section 6.3.3, which is a JavaScript library for static and dynamic charting. It provides a cross platform implementation supporting many different kinds of chart types.

[15]MagicalRecord details see: `https://github.com/magicalpanda/MagicalRecord`
[16]AJNotificationView details: `https://github.com/ajerez/AJNotificationView`

Jahmm

Jahmm[17] is a Java implementation of Hidden Markov Model related algorithms. Albeit not the fastest implementation it is well documented and convenient to use.

7.8. Source-Code and Installation

The requirements to deploy the mobile application on a iOS tablet are discussed in the *Device Deployment* section 4.1.2. Which in short requires an payed iOS developer membership. However, deployment and tryouts in the free Xcode accompanied iOS Simulator[98] are possible without any charge. The necessary setup for the mobile application requires an Intel processor based Apple Computer, an Apple ID, the free download of Xcode available in the OS-X AppStore and a clone of the the latest source code of the mobile application.
The AAL middleware used as an implementation example requires either a Linux, MacOS or Windows operated computer running a *Java Development Kit* of Orcale[18] (or compatible) and the build tool *Maven*[19]. Source-Code and further installation details are available on the project site `http://homer.aaloa.org`.

7.8.1. Source-Code

The source code of both, the mobile application *iHomer* and the OSGi bundles for the AAL middleware H.O.M.E.R., are available on GitHub[99] upon request under the following url `https://github.com/matthiasgasser`.

7.8.2. Installation on the iOS Tablet

After the requirements are met, and the latest source-code is cloned from the Git repository, then is the next step to open the `Mobile Homer.xcworkspace`

[17]Jahmm details: `https://code.google.com/p/jahmm/`
[18]Java Development Kit details and download `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[19]Apache Maven details and download `http://maven.apache.org`

file in Xcode. The project may now be build and run in the iOS Simulator or be deployed on a connected iOS tablet.

7.8.3. Installation of the AAL Middleware Addition

First it is required to download and build the H.O.M.E.R. AAL middleware according to the accompanied installation document. The same build tool called *Maven* is required to build the additional OSGi bundles. This is done by navigating into the root folder of the downloaded OSGi bundles and calling `mvn clean install`. This produces two JAR files ready to be moved into the deploy folder of the H.O.M.E.R. installation.

7.9. Tablet Application iHomer Demonstration

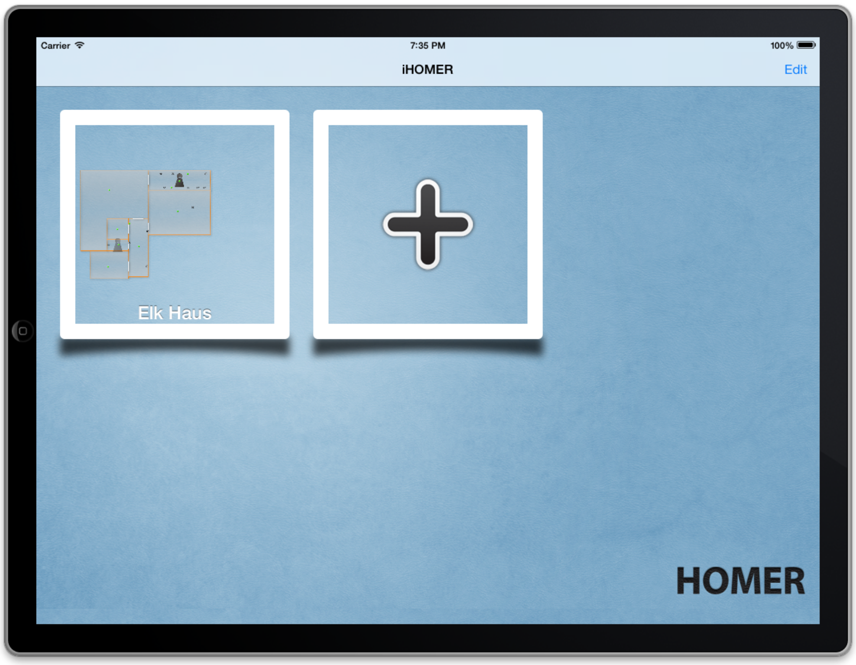
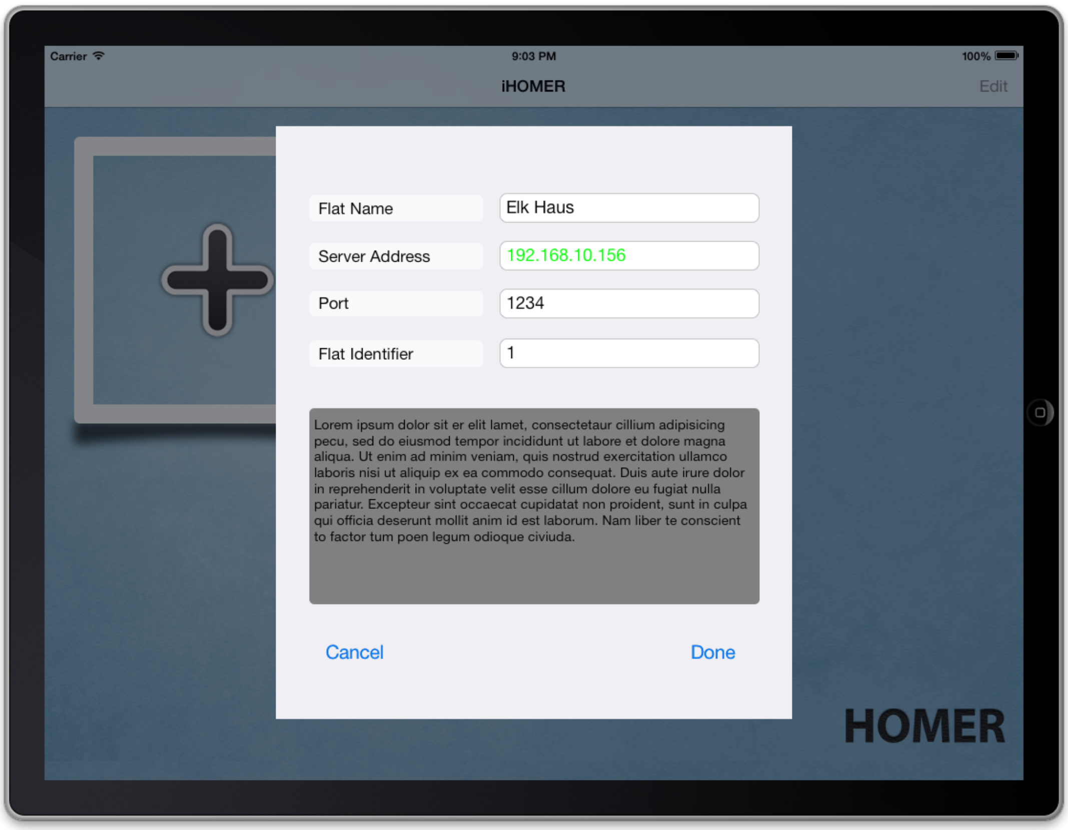


Figure 7.5.: iHomer Launch and Flat Overview Screenshot

Figure 7.5 represents the entry screen after application launch. Every white square marks an observed flat with the name and a minified representation of the flat layout, which allows to monitor and control more than one flat in the case of a sheltered housing scheme, the plus (+) sign gives access to the flat configuration form to add an additional flat. The edit button on the top right switches into the edit mode.



Flat Name	Elk Haus
Server Address	192.168.10.156
Port	1234
Flat Identifier	1

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

Figure 7.6.: iHomer Flat Configuration Screenshot

The flat configuration screen allows easy creating or updating a flat connection. The name is just a user readable record locator to identify this very flat. The server address can be either an IP[20]. or a domain name like `example.org`. Port

[20]Internet Protocol: an in IPV4 determined string of numbers like: `aaa.bbb.ccc.ddd` to identify internet participants. Details are found at:`http://www.iana.org/`

should correspond to the server port configured on the back-end side. The identifier represents the unique numeric identification of the flat, especially if the server hosts more than one flat.

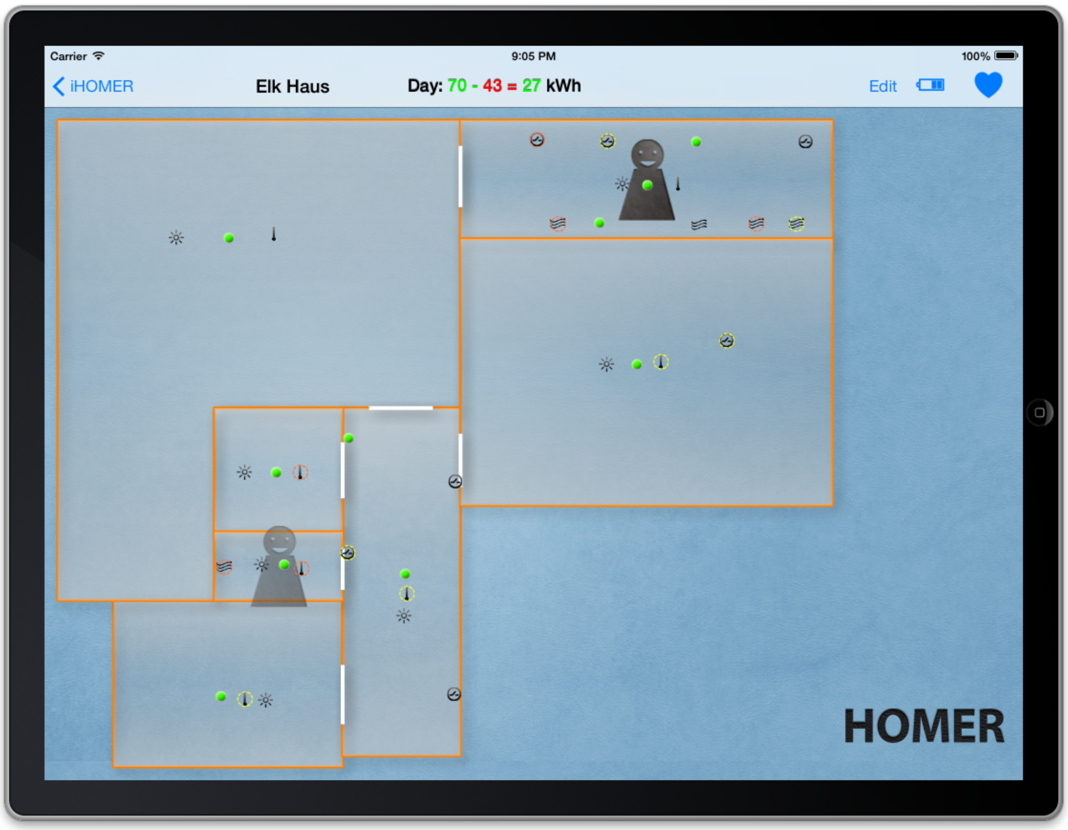
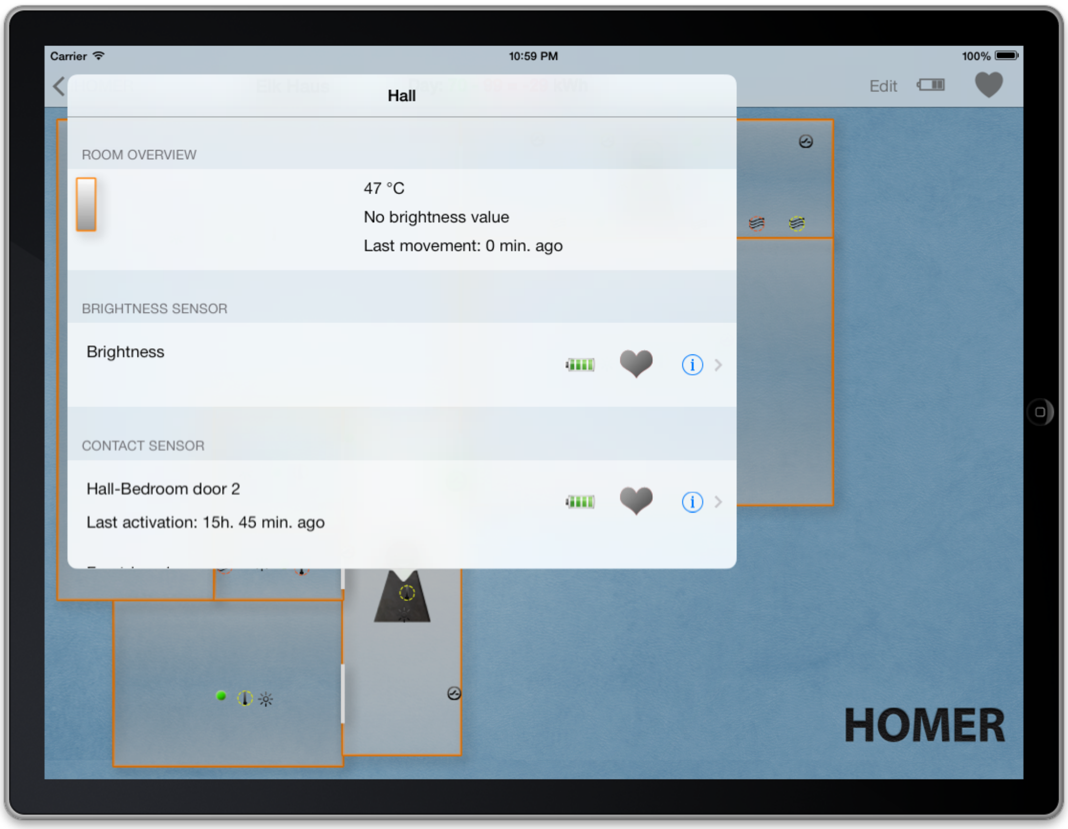


Figure 7.7.: iHomer Flat Screenshot

The main view of the flat which shows sensors and actuators posed by a matching pictogram, virtually on the corresponding locations for easy access. The visible devices types and favourites are customisable by the configuration popovers on the top right accessible by the `Edit` button. The floor plan also allows easy pinch to zoom gestures, for a detailed zoomed in view. Devices reporting weak (yellow) or empty (red) battery charges are shown in a colorised circle. There are two playing token figure look a likes which represent the last known (dark figure) and previous (fainted figure) room of occupancy. The top bar shows the user readable record locator defined in the flat configuration screen and the current energy balance in a day, week, monthly and yearly view alternating every few seconds. Touching this energy balance will bring up the full history in a convenient popover comparing generated and consumed energy, see figure 7.13.



ROOM OVERVIEW

47 °C
No brightness value
Last movement: 0 min. ago

BRIGHTNESS SENSOR

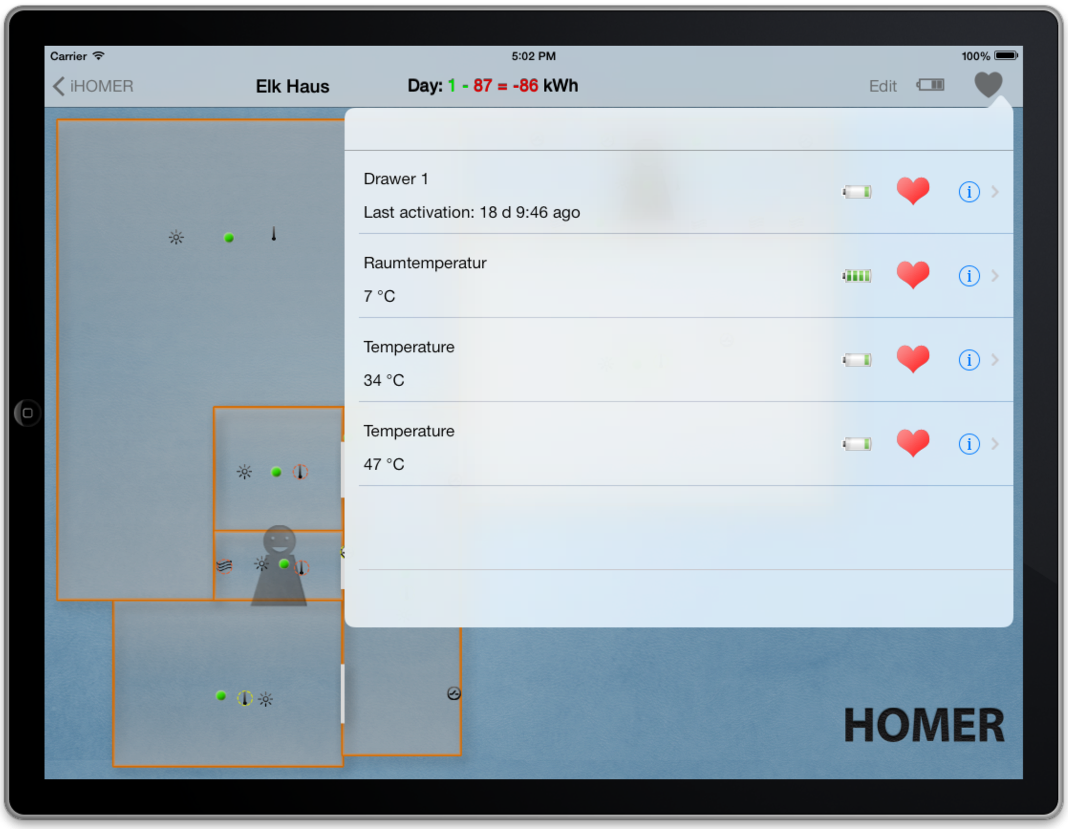
Brightness

CONTACT SENSOR

Hall-Bedroom door 2
Last activation: 15h. 45 min. ago

Figure 7.8.: iHomer Room Popover Screenshot

Touching a room in the main view opens the room popover, which compiles all sensors and actuators in the given room. Each sensor record shows the very last recorded value, battery status and allows access to even further details like history chart and values using the (i) button. The heart symbol adds the sensor to the favourite collection popover which is accessible by the heart symbol on the top right, see figure 7.9.



Drawer 1
Last activation: 18 d 9:46 ago

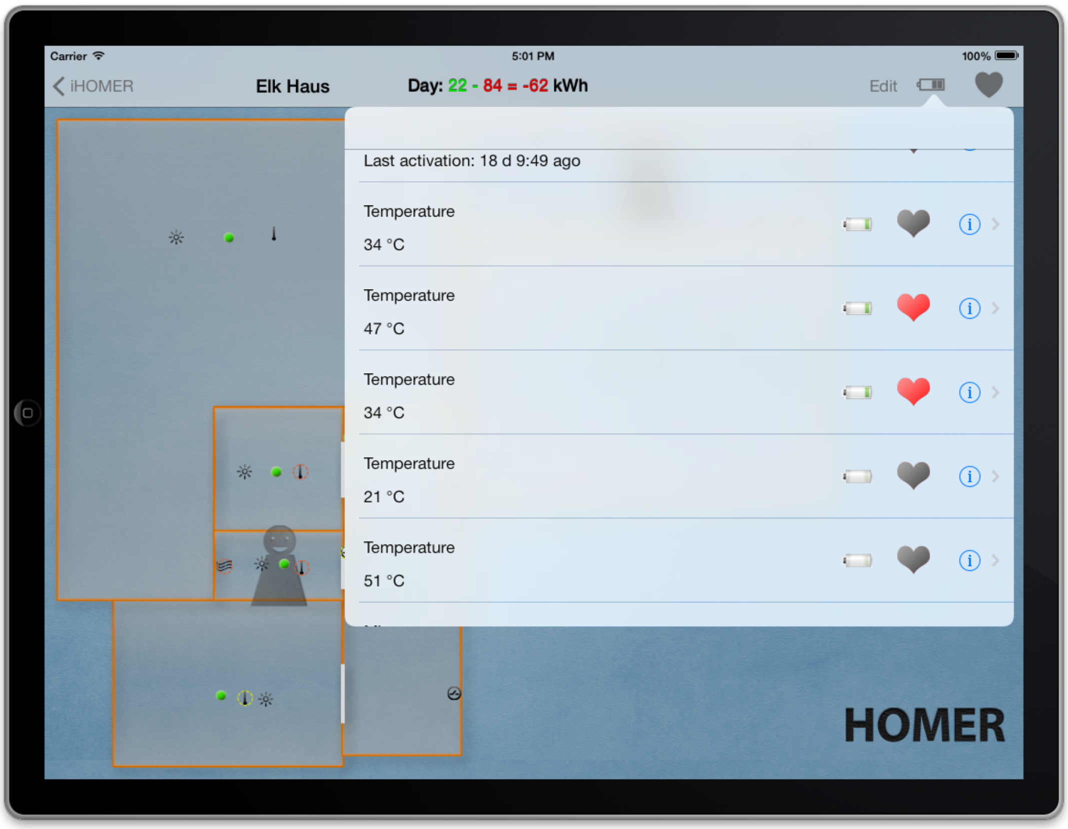
Raumtemperatur
7 °C

Temperature
34 °C

Temperature
47 °C

Figure 7.9.: iHomer Device Like Popover Screenshot

This popover gives fast and easy access to user selected devices. Each record shows similar to the room popover the last recorded value and the (i) button gives access to further details like historic values and charting.



Last activation: 18 d 9:49 ago

Temperature
34 °C

Temperature
47 °C

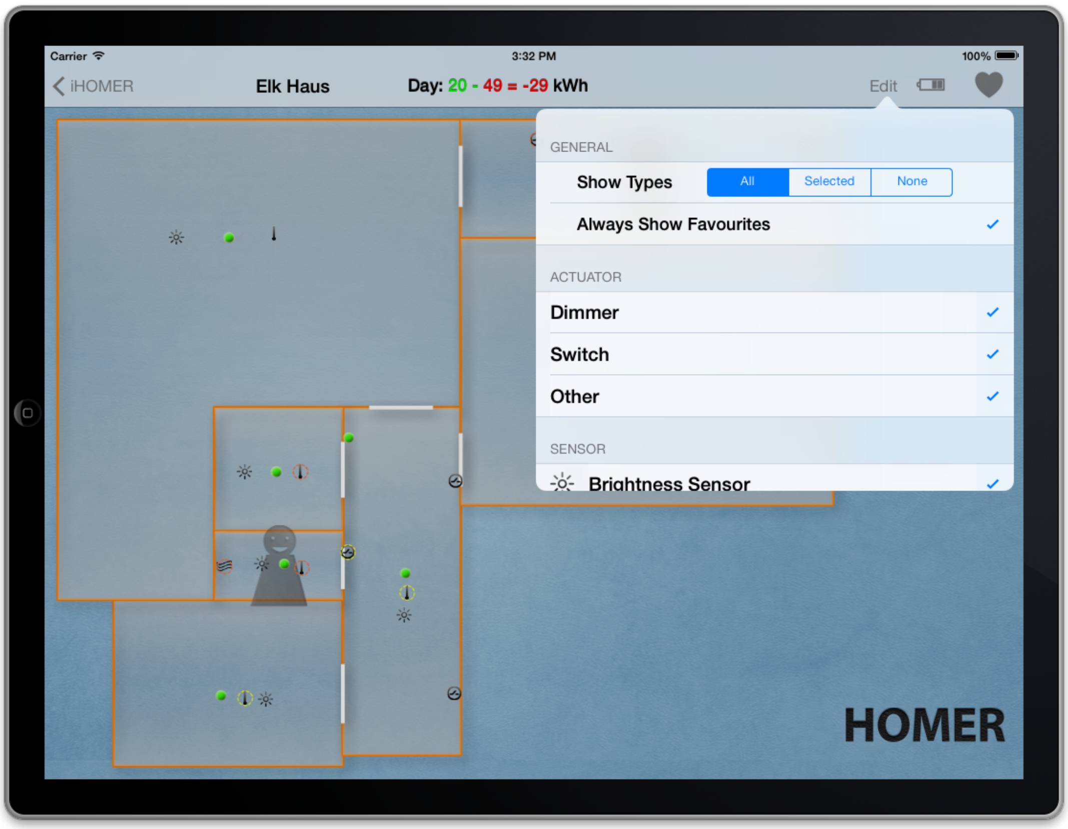
Temperature
34 °C

Temperature
21 °C

Temperature
51 °C

Figure 7.10.: iHomer Battery Popover Screenshot

Similar to the favourite popover presents the battery popover all devices reporting weak or near empty battery charges. The battery charge is symbolised by the battery icon.



GENERAL

Show Types [✓ All] Selected None

Always Show Favourites ✓

ACTUATOR

Dimmer ✓

Switch ✓

Other ✓

SENSOR

Brightness Sensor ✓

Figure 7.11.: iHomer Device View Popover Screenshot

The flat view configuration popover manages the visibility of devices in the flat view. Devices may be filtered by type like a dimmer,temperature sensor etc. and by favourite devices. This prevents the cluttering of the flat view in case of a myriad of devices.

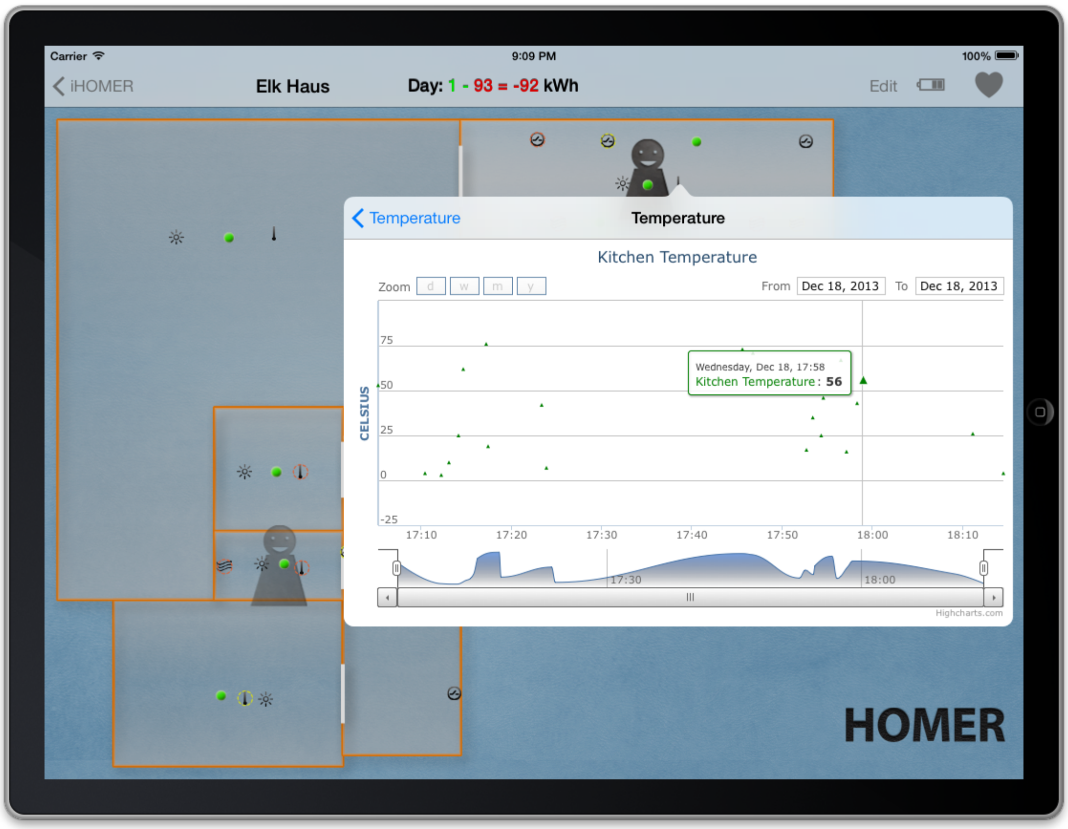


Figure 7.12.: iHomer Device Charting Screenshot

Selecting a sensor gives the option to show a chart of recorded data. This chart is interactive, whereby each datapoint may be touched for exact value. The date range is adjustable by either using the time view on the bottom or the d,w,m,y buttons on the top. Please note that the values in the screenshot are not representing an actual measurement, but are generated randomly by the given middleware.

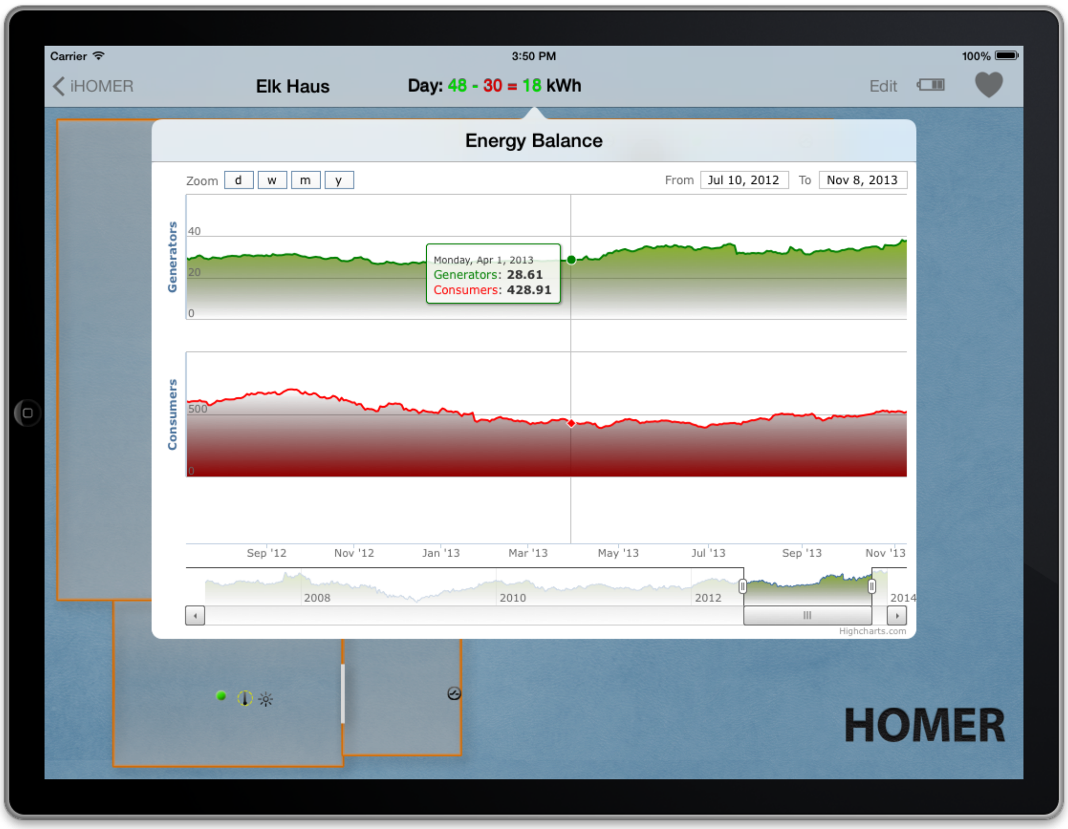


Figure 7.13.: iHomer Energy Balance Screenshot

The energy balance popover is identical to the sensor charting but comparing the generated energy versus the consumed energy. This values are also generated randomly.

8. Conclusion And Future Work

The goal of this thesis had been to develop an iOS application running on a tablet device which allows the visualisation and control of a sensor and actuator equipped smart home. As part of the visualisation a simple model was presented to detect if the users activity is within a to be normal considered state. This extends the field of application for previously only stationary control-able smart homes to the mobile field.
To understand the requirements and environment of the Ambient Assisted Living field was the social environment and life situation of the elderly and existing tablet supported AAL applications researched. Also the back-end side of currently existing AAL frameworks was evaluated and compared to each other. Most of them shared a similar architecture but aimed for distinct goals, but there is progress towards a unified standardised AAL platform observable but yet not finished.
This research led to the definition of the functional and nonfunctional requirements, to control and visualise the smart home. The development is geared towards a platform independence which is defined as nonfunctional requirement. The mobile connectivity standards are being evaluated and compared which supported the hardware requirements. The choice of hardware had been simple due to the limited choice of this monopolistic space.
The environment and technical details of the iOS and AAL middleware platforms are researched in respect to the requirements. This is particular important to be able to understand and to choose technologies for the design phase.
Given the requirements and we understood the technology environment we were able to actually start the software design and user interface mockups. This

design documents are then being implemented using well known software design patterns to be in line with modularity and reusability. The results of the this process are discussed after the implementation.
This thesis gives an overview of the Ambient Assisted Living field and their current ongoing trends and developing standards but still it is a very young and dynamic area of work and research. The booming trend of tablet devices and the need for an intelligent environment will lead the way into an still to be explored land. The introduced mobile application may be a step towards in this journey to explore methods and ways to actually support people in need without chaining them in an uncomfortable way to the new technologies yet to come.
There are many hurdles like the cost of sensors and infrastructure and installation costs on the way, but they are surely going to be tackled in the future as this broadens up to a wider customer target.
The mobile application is a small step into the future providing a head start in the tablet driven AAL field being publicly available for scientists and any other person. However there is still a lot of work todo like improving the algorithms and evaluating and performance testing existing ones.
And I am absolutely sure that if we respect the balance of technology and not over-assisting our families we are able to sustain people's happiness but also keep them save and socially integrated.

“ The traveler sees what he sees, the tourist sees what he has come to see.

G.K. Chesterton

List of Tables

List of Figures

Glossary

3GPP 3rd Generation Partnership Project. 24

AAL Ambient Assisted Living. 3, 7, 15, 37, 61, 63, 80

AALOA AAL Open Association. 14

Ambient Assisted Living Are housing facilities for people with disabilities. 3, 7, 8

API Application Programming Interface. 19, 20, 29

App Common name for a mobile application. 18

CRF Condition Random Field. 45

EDGE Enhanced Data rates for GSM Evolution. 23

GPS Global Positioning System. 29

GRPS General Packet Radio Service. 23

H.O.M.E.R Home Event Recognition System. 15

HMM Hidden Markov Model. 45

HTML Hyper Text Markup Langauge. 87

HTTP Hypertext Transfer Protocol. 37, 89

IDE Integrated Developing Environment. 30

IEEE Institute of Electrical and Electronics Engineers. 19, 23

IETF Internet Engineering Task Force. 40

ISM Industrial, Scientific and Medical radio bands. 23

JSON JavaScript Object Notation. 38, 39

KVO Key Value Observing. 77

LTE Long Term Evolution. 23

MVC Model View Controller Design Pattern. 74

OSGi Open Services Gateway initiative. 32, 73

REST Representational State Transfer. 37, 38, 62, 85, 89

SOAP originally defined as Simple Object Access Protocol. 37, 38

SVG Scalable Vector Graphics. 67

SVM Support Vector Machine. 45

UMTS Universal Mobile Telecommunications System. 23

UniversAAL UniversAAL open platform and reference Specification for Ambient Assisted Living. 13, 14

URL Uniform Resource Locator, identifies and localises a resource like e.g. a webpage including the access method. 29

Wi-Fi Wireless Fidelity. 23

WSDL Web Services Description Language. 37, 38

XML Extensible Markup Language. 37, 38, 75

Bibliography

[1] Steg, H., et al. Europe is facing a demopgraphic challeng - ambient assisted living offers solutions. *VDI/VDE/IT, Berlin, Germany*, 2006.

[2] Scoping Report. The Future of the Family to 2030 – A Scoping Report –. *OECD International Futures Programme*, December 2008.

[3] Conseil de l'Europe. `http://www.assembly.coe.int/Mainf.asp?link=/Documents/AdoptedText/ta12/ERES1864.htm`, 06 2012.

[4] DESTATIS Statistisches Bundesamt. `https://www.destatis.de/bevoelkerungspyramide/`, 03 2013.

[5] Bründel H. Hurrelmann K. *Einführung in die Kindheitsforschung*, page 82f. Beltz, Weinheim/Basel/Berlin, 2. edition, 2003.

[6] H.P. (Hrsg.) Naegele G., Tews. *Lebenslagen im Strukturwandel des Alterns.* Opladen: Westdeutscher Verlag, 1993.

[7] A Kruse. Alterspolitik und gesundheit. *Bundesgesundheitsblatt - Gesundheitsforschung – Gesundheitsschutz*, 6:513–522, 2006.

[8] Executive Summary. Executive summary. *New directions for youth development*, 2012(136):7–11, December 2012.

[9] Birgid Eberhardt Reiner Wichert. Ambient assisted living. In Springer, editor, *4. AAL - Kongress 2011*, 2011.

[10] Peter Georgieff. Ambient Assisted Living - Marktpotentiale IT-unterstützter Pflege für ein selbstbestimmtes Altern. *Integration The Vlsi Journal*, (Fazit), 2008.

[11] Cobago. `http://www.cobago.net`, 07 2012.

[12] MobileHealthNews.com. Report: 13k iphone consumer health apps in 2012. urlhttp://mobihealthnews.com/13368/report-13k-iphone-consumer-health-apps-in-2012/, 09 2011.

[13] Chang Liu, Qing Zhu, Kenneth a. Holroyd, and Elizabeth K. Seng. Status and trends of mobile-health applications for iOS devices: A developer's perspective. *Journal of Systems and Software*, 84(11):2022–2033, November 2011.

[14] universaal project. `http://www.universaal.org`, 06 2012.

[15] AAL Open Association AALOA. `http://aaloa.org`, 04 2013.

[16] AIT Austrian Institute of Technology GmbH. `http://homer.aaloa.org/redmine/projects/homer-core`, 02 2012.

[17] Mpower. `http://www.sintef.no/mpower`, 07 2012.

[18] Companionable. `http://www.companionable.net`, 07 2012.

[19] Soprano ip. `http://www.soprano-ip.org`, 07 2012.

[20] Frauenhofer allianz ambient assisted living oasis. `http://www.aal.fraunhofer.de/projects/oasis.html`, 07 2012.

[21] Persona project. `http://www.aal.fraunhofer.de/projects/persona.html`, 07 2012.

[22] Antonio Kung and Bruno Jean-bart. Making AAL Platforms a Reality. pages 187–196, 2013.

[23] Juan a. Botia, Ana Villa, and Jose Palma. Ambient Assisted Living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications*, 39(9):8136–8148, July 2012.

[24] Finland. Ambient assisted living. *country report*, 2005.

[25] Hong Sun, Vincenzo De Florio, Ning Gui, and Chris Blondia. Promises and Challenges of Ambient Assisted Living Systems. *2009 Sixth International Conference on Information Technology: New Generations*, pages 1201–1207, 2009.

[26] IBBT. Coplintho communication platform for interactive ehomecare. `https://projects.ibbt.be/coplintho/`, 04 2013.

[27] Risto Karlsson. *Helsingin Sanomat*, 18.10 1996.

[28] Jonas Grauel and Annette Spellerberg. Attitudes and requirements of elderly people towards assisted living solutions. In Max Mühlhäuser, Alois Ferscha, and Erwin Aitenbichler, editors, *Constructing Ambient Intelligence*, volume 11 of *Communications in Computer and Information Science*, pages 197–206. Springer Berlin Heidelberg, 2008.

[29] Woong Hee Kim, Sunyoung Lee, and Jongwoon Hwang. Real-time Energy Monitoring and Controlling System based on ZigBee Sensor Networks. *Procedia Computer Science*, 5:794–797, January 2011.

[30] Matthias Brinkmann, Martin Floeck, and Lothar Litz. Concept and design of an aal home monitoring system based on a personal computerized assistive unit. In Max Mühlhäuser, Alois Ferscha, and Erwin Aitenbichler, editors, *Constructing Ambient Intelligence*, volume 11 of *Communications in Computer and Information Science*, pages 218–227. Springer Berlin Heidelberg, 2008.

[31] IEEE Standards Board. IEEE Standard Glossary of Software Engineering Terminology. *The Institute of Electrical and Electronics Engineers*, 121990, 1990.

[32] Karl E. Wiegers. *Software Requirements*. Second edition edition, 2007.

[33] Gruia-catalin Roman and St Lolis. A Taxonomy of Current Issues in Requirements Engineering.

[34] Jack Mostow and Marina Rey. Toward Better Models Of The Design Process. 6(1):44–57, 1985.

[35] Merlin Dorfman. Requirements Engineering. pages 1–30, 1999.

[36] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, June 1992.

[37] Ieee 802.11 working group. `http://www.ieee802.org/11/`, 07 2013.

[38] The wi-fi alliance. `http://www.wi-fi.org`, 07 2013.

[39] The 3rd generation partnership project (3gpp). `http://www.3gpp.org`, 07 2013.

[40] Engadget.com. Engadget.com firsh third-party app appears for iphone. `http://www.engadget.com/2007/08/06/first-third-party-game-app-appears-for-iphone/`, 06 2012.

[41] TUAW.com. Hello world, says iphone. `http://www.tuaw.com/2007/07/20/hello-world-says-iphone/`, 06 2012.

[42] John Laugesen and Yufei Yuan. What Factors Contributed to the Success of Apple's iPhone? *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, pages 91–99, 2010.

[43] Apple Inc. Apple wwdc special event june 2012. `http://www.apple.com/apple-events/june-2012/`, 06 2012.

[44] Apple Inc. Objective-c-primer. https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/_index.html, 06 2012.

[45] Apple Inc. Cocoa fundamentals guide. https://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html#//apple_ref/doc/uid/TP40002974-CH3-SW16, 06 2012.

[46] Apple Inc. Xcode 4 user guide. https://developer.apple.com/library/ios/#documentation/ToolsLanguages/Conceptual/Xcode4UserGuide/000-About_Xcode/about.html#//apple_ref/doc/uid/TP40010215, 06 2012.

[47] Apple Inc. Developing for the app store. https://developer.apple.com/library/ios/#documentation/General/Conceptual/ApplicationDevelopmentOverview/DeliverYourAppontheAppStore/DeliverYourAppontheAppStore.html#//apple_ref/doc/uid/TP40011186-CH8-SW1, 06 2012.

[48] Apple Inc. ios developer program. https://developer.apple.com/programs/start/ios/, 06 2012.

[49] Apple Inc. ios provisioning portal. https://developer.apple.com/library/ios/#recipes/ProvisioningPortal_Recipes/_index.html, 06 2012.

[50] OSGi Alliance. Osgi. http://www.osgi.org, 05 2013.

[51] James A Sena, D Ph, and California Polytechnic. The PC Evolution and Diaspora. (April), 2012.

[52] Apple Inc. Apple inc. http://store.apple.com/us/ipad/compare, 07 2013.

[53] W3.org. Web services architecture working group. http://www.w3.org/TR/ws-gloss/, 06 2013.

[54] Michael zur Muehlen, Jeffrey V. Nickerson, and Keith D. Swenson. Developing web services choreography standards—the case of REST vs. SOAP. *Decision Support Systems*, 40(1):9–29, July 2005.

[55] Roy Thomas Fielding. Representational state transfer (rest). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 11 2013.

[56] Francis Galiegue. Json schema: core definitions and terminology. http://json-schema.org/latest/json-schema-core.html, 11 2013.

[57] The RestKit Project. Restkit: Objective-c framework for ios. http://restkit.org, 11 2013.

[58] Rob Gravelle. Comet programming: Using ajax to simulate server push. http://www.webreference.com/programming/javascript/rg28/index.html, 11 2013.

[59] Internet Engineering Task Force (IETF). The websocket protocol. http://tools.ietf.org/html/rfc6455, 11 2013.

[60] W3C. Soap specifications. http://www.w3.org/TR/soap/, 11 2013.

[61] Liming Chen and Ismail Khalil. Activity Recognition : Approaches , Practices. pages 1–31.

[62] T L M Van Kasteren, G Englebienne, and B J A Kröse. Human Activity Recognition from Wireless Sensor Network Data : Benchmark and Software. pages 165–186.

[63] Dan Ding, Rory a Cooper, Paul F Pasquina, and Lavinia Fici-Pasquina. Sensor technology for smart homes. *Maturitas*, 69(2):131–6, June 2011.

[64] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes- present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81, July 2008.

[65] Marjorie Skubic, Gregory Alexander, Mihail Popescu, Marilyn Rantz, and James Keller. A smart home application to eldercare: current status and lessons learned. *Technology and health care : official journal of the European Society for Engineering and Medicine*, 17(3):183–201, January 2009.

[66] Afsaneh Doryab and Julian Togelius. Concurrent Activity Recognition For Clinical Work.

[67] Tim Van Kasteren. Activity Recognition for Health Monitoring Elderly using Temporal Probabilistic Models.

[68] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, number July, pages 209–216, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[69] Emmanuel Munguia Tapia, Stephen S. Intille, William Haskell, Kent Larson, Julie Wright, Abby King, and Robert Friedman. Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor. *2007 11th IEEE International Symposium on Wearable Computers*, pages 1–4, October 2007.

[70] Rachid Kadouche, Hélène Pigot, Bessam Abdulrazak, and Sylvain Giroux. User ' s Behavior Classification Model for Smart. pages 149–164.

[71] L. Liao, D. Fox, and H. Kautz. Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. *The International Journal of Robotics Research*, 26(1):119–134, January 2007.

[72] Bernt Schiele. Unsupervised Discovery of Structure in Activity Data using Multiple Eigenspaces.

[73] Dietmar Bruckner, Brian Sallans, and Roland Lang. Behavior learning via state chains from motion detector sensors. *2007 2nd Bio-Inspired Models of Network, Information and Computing Systems*, pages 176–183, December 2007.

[74] Lawrence R. Rabiner B.H. Juang. An introduction to hidden Markov models. *Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [et al.]*, Appendix 3(January):Appendix 3A, June 2007.

[75] J. a. Bilmes. What HMMs Can Do. *IEICE Transactions on Information and Systems*, E89-D(3):869–891, March 2006.

[76] Guoqing Yin and Dietmar Bruckner. Data Analyzing and Daily Activity Learning with Hidden Markov Model GuoQing Yin and Dietmar Bruckner Institute of Computer Technology.

[77] Andreas Stolcke. Hidden Markov Model Induction by Bayesian. (Ml), 1993.

[78] Guo Qing Yin. Automatic Scenario Detection for Ambient Assisted Living of Elderly People. (March), 2012.

[79] You-Jia Liu Chih-Yu Hsiao and Mao-Jiun J. Wang. Usability evaluation of the touch screen user interface design. In Sakae Yamamoto, editor, *Human Interface and the Management of Information*, volume Part I, LNCS 8016 of *HIMI/HCII*, pages 48–50. Springer, 2013.

[80] Dag Svanæ s and Gry Seland. Putting the Users Center Stage : Role Playing and Low-fi Prototyping Enable End Users to Design Mobile Systems. 6(1):479–486, 2004.

[81] Apple Inc. Table view programming guide for ios. `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/`

TableView_iPhone/TableViewAndDataModel/TableViewAndDataModel.html, 11 2013.

[82] S. J. Richardson B. Shackel, editor. *Human Factors for Informatics Usability.* Cambridge University Press, http://books.google.at/books?hl=de&lr=&id=KSHrPgLlMJIC&oi=fnd&pg=PA21&dq=usability+definition&ots=IUVrIUZVFc&sig=hzmvbBiXvFfMML2CNaBDHdzGSEA#v=onepage&q=usability%20definition&f=false, February 1991.

[83] Apple Inc. ios human interface guidelines. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, 11 2013.

[84] Murray Silverstein Max Jacobson Ingrid Fiksdahl-King Shlomo Angel Christopher Alexander, Sara Ishikawa. A Pattern Language. *Oxford University Press, New York*, 1977.

[85] Erich Gamma, Helm Richard, Johnson Ralph, and Vlissides John. *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley, USA, 1994.

[86] Adele Goldberg and David Robson. *Smalltalk-80.* Addison-Wesley, Xerox Palo Alto Research Center, 1983.

[87] Horia Ciocarlie. Best practices in iPhone programming.

[88] Glenn E Krasner and Stephen T Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. 1988.

[89] Apple Inc. Concepts in objective-c programming. https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html#//apple_ref/doc/uid/TP40010810-CH14, 11 2013.

[90] Apple Inc. Cocoa bindings programming topcis: Introduction to cocoa bindings. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CocoaBindings/CocoaBindings.html, 11 2013.

[91] Apple Inc. Cocoa core competencies: Singleton. https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Singleton.html, 11 2013.

[92] Apple Inc. Key-value observing programming guide. http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/KeyValueObserving/, 11 2013.

[93] Apple Inc. Notification programming topics: Notification center. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Notifications/Articles/NotificationCenters.html, 11 2013.

[94] Apple Inc. Cocoa fundamentals guide: Cocoa design patterns. https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/ObjectiveC/Chapters/ocProtocols.html#//apple_ref/doc/uid/TP30001163-CH15, 11 2013.

[95] Apple Inc. Core data programming guide. https://developer.apple.com/library/mac/documentation/cocoa/conceptual/coredata/articles/cdMOM.html, 11 2013.

[96] Steven Raemaekers, Arie Van Deursen, and Joost Visser. An Analysis of Dependence on Third-party Libraries in Open Source and Proprietary Systems.

[97] Massachusetts Institute of Technology. The mit license. http://opensource.org/licenses/MIT, 11 2013.

[98] Apple Inc. ios simulator user guide. https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/, 11 2013.

[99] GitHub. Github platform. `http://github.com`, 11 2013.