Cork Institute Of Technology

Masters Report

COAcHMAN - Enabling Context Aware Home Automation

Author:
Aidan O Mahony

Supervisor:
Dr. John Creagh

A report submitted in partial fulfilment of the requirements for the degree of Master of Science

in the

Department of Computing

August 2015

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering & Science
Department of Computing

Master of Science

COAcHMAN - Enabling Context Aware Home Automation

by Aidan O Mahony

Home Automation and Ambient Assisted Living technologies are becoming ubiquitous in our society. Technologies such as Google's NEST smart thermostat and Apples HomeKit are gradually appearing in our homes. However, these technologies are mostly incompatible with each other and require a good deal of user interaction.

The aim of this research project is to simplify users interactions with their smart homes. Subsequent to this, linking online user profiles to smart home environments is investigated. The primary objective is achieved by conducting background research into context awareness and home automation technologies and, as a result of this research, a software solution (COAcHMAN - COntext Aware HoMe AutomatioN) is proposed. Using COAcHMAN, users can enable their homes to react based on the users situation (or context) rather than requiring user interaction. Furthermore, as COAcHMAN makes use of online profiles there exists a familiar interface for users of the system.

Contents

List of Figures

List of Tables

Abbreviations

COAcHMAN	**CO**ntext **A**ware **H**o**M**e **A**utomatio**N**
IoT	**I**nternet **o**f **T**hings
AAL	**A**mbient **A**ssisted **L**iving
OSGi	**O**pen **S**ervices **G**ateway **i**nitiative
HAN	**H**ome **A**rea **N**etwork
OWL	**W**eb **O**ntology **L**anguage
BSN	**B**ody **S**ensor **N**etwork
openHAB	**open** **H**ome **A**utomation Bus
DAO	**D**ata **A**ccess **O**bject
FIFO	**F**irst **I**n **F**irst **O**ut
SHE	**S**mart **H**ome **E**nvironment
GPS	**G**lobal **P**ositioning **S**ystem
CBR	**C**ase **B**ases **R**easoning
RFID	**R**adio **F**requency **ID**entification
SOAP	**S**imple **O**bject **A**ccess **P**rotocol

Dedicated to my wife Aileen and my children Ashleigh, Alex, and AJ. Without their sacrifices I would not have been able to achieve my goals.

Chapter 1

Introduction

It is predicted that by the year 2020 there will be 26 billion interconnected devices in the world[4]. These devices will range from personal devices (e.g. smart watches, smart phones, fitness bands) to automotive devices (e.g. self-driving cars, in-vehicle infotainment systems, on-board diagnostic systems) to health devices and many more. What is becoming clear therefore is that the users of these devices are incapable of providing input and directions to each of the devices they interact with. As a consequence of this, the devices we will interact with will need to anticipate our instructions. Rather than wait for a user to provide input, the systems the users are interacting with will be able to infer the users direction. In simple terms these devices will need to be "smart".

The purpose of this research project is to provide a real world "smart" home software implementation (named COAcHMAN - COntext Aware HoMe AutomatioN). In this report the research carried out, the conclusions deduced, and the reasoning behind the COAcHMAN software system are documented. The report will also provide details on a practical deployment of the COAcHMAN system and how it can be used to simplify interactions with a number of these "smart devices".

1.1 Smart Environments and Smart Living

At this point a reader will have deduced that this project is concerned with devices which play a part in smart environments. Cook and Das (2004)[5] define a smart environment as one *that is able to acquire and apply knowledge about an environment and also to adapt to its inhabitants in order to improve their experience in that environment.* To provide background for the user this section will introduce the concept of a smart city, a smart grid, and a smart home and also will provide some information on how they are intertwined.

1.1.1 Smart Cities

Nam and Pardo (2011)[6] provide a number of definitions of smart cities. One definition that complements this project is - *A city that monitors and integrates conditions of all of its critical infrastructures, including roads, bridges, tunnels, rail/subways, airports, seaports, communications, water, power, even major buildings, can better optimise its resources, plan its preventive maintenance activities, and monitor security aspects while maximising services to its citizens.*

The EU is funding three Smart Cities projects under the umbrella of the *Horizon 2020*[7] research and innovation framework programme. These are

- GROWSMARTER - this project covers the areas of low energy districts, integrated infrastructures and sustainable urban mobility, and includes deep retrofitting approaches for buildings from the 60s and 70s. This project is also interested in vehicles fuelled from alternative sources and smart traffic management.
- TRIANGULUM - this project is structured around zero / low energy districts, integrated infrastructures, and sustainable urban mobility.
- REMOURBAN - the emphasis of this project is on reaching a holistic approach where energy production, distribution and use, mobility and transport, and ICT come to form a continuum.

1.1.2 Smart grid

In *The path of the smart grid*[8], Farhangi presents the existing electricity grid as unidirectional in nature, wasteful, and suffering from domino effect failures. The solution to these problems is the next generation electricity grid - also known as the smart grid.

Some of the characteristics of the smart grid are as follows

- Two way communication - communication between smart meters in homes and the utility providers. This will facilitate optimum power usage.
- Remote check and testing.

- Self monitoring and self healing.
- Distributed generation.
- Sensors throughout.

Also touched on by Farhangi is the Home Area Network (HAN) and it's role within the smart grid.

1.1.3 Smart Homes

In the context of smart cities and the smart grid, the smart home is the part the inhabitant of the smart environment has direct interaction with. In *Smart home gateway for smart grid*[9] Al-Ali, El-Hag, Dhaouadi and Zainaldain present a smart home computing platform that integrates a smart home appliance remote monitoring and control system with an automatic meter reading system to make the home ready for smart grid integration. Figure A.1 in Appendix A illustrates the relationships between the smart home and the smart grid.

1.2 Home Automation

As a consequence to the link between smart homes, the smart grid, and smart cities, the need for home automation arises. The purpose of a smart home is to *provide an awareness of the users' needs, providing them with a better home life experience without overpowering them with complex technologies*[10]. In *Home Automation in the Wild: Challenges and Opportunities*[11] Bernheim et al present a number of homes which have adopted home automation technologies for a variety of purposes including security, convenience, and task consolidation. They also presented barriers; high cost of ownership, inflexibility, and poor manageability. While COAcHMAN aims to address some of the issues around poor manageability, it is worthwhile spending some time examining the benefits of home automation before addressing the difficulties. In the next two sections we look at two of the main benefits of smart homes and home automation.

1.2.1 Ambient Assisted Living

As is recognised by many, our populations in the western world are generally ageing. As a result, the concept of Ambient Assisted Living (AAL) is gaining interest in many circles (in fact, the European Union is sponsoring 6 AAL projects in Ireland alone). AAL can be described as *a concept focused on the use of technology to improve the independence and well being of aged people in the environment where they live or work*[12].

Eichelberg, Rolker-Denker and Helmer[13] present a number of use cases for AAL. A selection below illustrate the connection between AAL and home automation.

- Behaviour Monitoring - the technology recognises dangerous events and changes of behaviour patterns and, depending on the type of event either notifies the user or calls for help.
- Calendar Service - The Calendar Service system acts as an "intelligent" calendar that reminds the user of appointments, notifies the user if certain activities of daily living are forgotten, and reminds the user of the medicine that is to be taken.
- Social Interaction with Smart TV - Technology can help older adults to stay in contact with friends, family and carers despite limited mobility by offering advanced communication functions such as user friendly video communication.

1.2.2 Energy Management

22% of all energy consumption is due to energy consumption from residential and commercial buildings[14]. One avenue for energy savings is to make use of home energy management systems. An example of a Smart Home Energy Management system is provided by the Authentic[15] project. This project aims to manage energy within a home. It achieves this goal via

- Appliance Scheduling - scheduling appliances to run based on time preferences, energy costs at the time, and to balance the overall load of the grid.
- Decision making - Controlling devices within the home based on external stimuli (e.g. temperature).
- Interfacing with the smart grid.

1.3 Internet of Things

The phrase "Internet of Things" (IoT) was first coined by Kevin Ashton in 1999[16] (as a title of a presentation at Proctor & Gamble). In simple terms it refers to the idea that a variety of *things* around us (i.e. sensors, phones, appliances) are able to interact with each other and cooperate with their neighbours to reach common goals[17]. While this project does not explicitly concern itself with IoT matters it is appropriate to bear in mind that this project does fall under this category. Figure A.2 in Appendix A shows an infograph of how the IoT effects the Smart Home.

1.4 Context Awareness

The primary area of research for this project is in the area of context awareness. Dey[18] proposes that when humans talk with other humans they are able to use implicit information - context - to understand each other more effectively. This does not transfer well to computer interactions. By increasing the ability of computers to understand context it is expected that the richness of communication between humans and computers will also increase.

Chapter 2 provides background research carried out in this area for this project however it worth providing some history and applications of context awareness to understand how it is useful in day to day activities.

1.4.1 Background

In the first work that introduces the term “context-aware”[19], Schilit and Theimer provide a basic yet still relevant definition of context as *the ability of a mobile user’s application to discover and react to changes in the environment they are situated in.* Initially, most context-aware research was in the areas of ubiquitous computing (e.g. a users profile following them so they can access their own data at any terminal within a building) and mobile commerce (e.g. targeted advertising)[20]. However as the IoT concept gathered momentum further applications of context-aware research was found.

1.4.2 Applications

Table 1.1 illustrates a number of context-aware applications that are currently available.

Category	**Application**	**Implementation**
Pervasive Gaming	Using the sensed human contexts to adapt game system behaviours	Google Ingress
Asset management	Used to locate rogue devices,interferers, Wi-Fi clients, tagged assets, and wired devices	Cisco Context-Aware Software
Activity Launching	Allow a user to setup rules for quickly launching apps based on their proximity to any given iBeacon they have placed in their home	iBeacon

Table 1.1: Context Aware Applications

1.5 Purpose of the project

One of the barriers to smart home adoption identified by Bernheim et al[11] is poor manageability. Some of the issues around poor manageability were

- Complex User Interface - one subject of the paper claimed the “Spouse Acceptance Factor” is not high enough.
- Lack of structure in subjects activities - rules based home automation systems did not handle change in routine as well as might be expected

The COAcHMAN system aims to address these adoption issues.

Chapter 2

Context Awareness Background Research

"Context is the new black" claims Aparna Chennapragada, head of product for Google Now [21]. What Aparna Chennapragada was referring to was the realisation by Google that users have different needs at different times and at different locations. The article also touches on the fact that Google recognises that users location data is likely extremely useful information to advertisers.

This chapter presents the results of background research into the area of Context Awareness, architectures of context aware implementations, and methods and algorithms for deducing users context.

2.1 Context Awareness

As briefly touched upon in the last chapter, context awareness refers to *the ability of a mobile user's application to discover and react to changes in the environment they are situated in.* In their paper "Context-Aware Computing Applications" [22], Schilit, Adams, and Want describe context-aware systems as systems *that adapt according to the location of use, the collection of nearby people, hosts, accessible devices, as well as to changes to such things over time.* What they are describing is a system which **reacts** to context and context changes. To understand context and context-awareness in greater detail the next number of sections provide background on context awareness fundamentals, the context life-cycle, and summarises a literature review into the area.

Authors	**Characteristics of context**
Schilit & Theimer[19]	location, identities of nearby people and objects, and changes to those objects
Schilit et al[22]	where you are, who you are with, what resources are nearby
Brown et al[23]	location, identities of people around the user, time of day, season, etc
Ryan et al[23]	user location, environment, identity, and time
Dey & Abowd[23]	location, identity, time, and activity

TABLE 2.1: Characteristics of context

2.1.1 Context Awareness Fundamentals

This section provides fundamental definitions, explanations, features, types and design principles which will be required later when discussing the COAcHMAN software.

2.1.1.1 Context

As is common with broad areas such as context there are many different definitions available for context. The two main accepted definitions are from Schilit et al[22] - *where you are, who you are with and what resources are nearby* and Dey et al[18] - *Context is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

As well as varying definitions of context there are also different definitions of the Characteristics of context. Table 2.1 present a number of Characteristics of context as presented in a number of seminal papers in the area. For the purpose of this project the definition of context and characteristics defined by Dey & Abowd are used. Perera, Zaslavsky, Christen, and Georgakopoulus in "Context Aware Computing for The Internet of Things : A Survey"[3] summarise the difference between raw data and context information

- **Raw (sensor) data**: is unprocessed and retrieved directly from the data source
- **Context Information**: is generated by processing raw sensor data

Also, Perera et al summarise the difference between the two different categories of events

- **Discrete events** : is where an event that occurs and time t and time $t+p$ (where p is the sampling rate) are considered to have been two separate events (e.g. turning on a light).

- **Continuous events** : is where an event lasting for time p and an event occurs and time t and one at time $t + p$ cannot be considered as two separate events (e.g. travelling in a car).

2.1.1.2 Context-Awareness

The two primary definitions of context awareness come from Schilit et al[22] - *the ability of a mobile user's application to discover and react to changes in the environment they are situated in*, and Dey[18] - *A system is context-aware if it used context to provide relevant information and/or services to the user, where relevancy depends on the user's task*. The latter definition is a more general one and of more use to this project as is Dey's definition of context (discussed in section 2.1.1.1).

2.1.1.3 Context-Aware features

Perera et al[3] provides a summary of the desired features of a context-aware system derived from previous research. The useful features of a context aware system are:

- **Presentation** : using context to decide what information and services to present to a user. Consider the example of a smart fridge which has a front display which can alert the user to produce approaching is use by dates.
- **Execution** : automatic execution of services. Consider the case where a user starts driving home from work, the smart home system can switch on the air conditioning and coffee machine to be ready when the user arrives home.
- **Tagging** : in a context-aware system there will be a large number of sensors and devices generating huge amounts of data. This data will need to be inspected and analysed. Therefore context needs to be tagged together with sensor data to be processed and understood later.

2.1.1.4 Context types

Perera et al[3] depict two types of context

- **Primary Context:** This is defined as any information retrieved without using existing context (e.g. GPS coordinates of a user)

- **Secondary Context:** This is defined as information which is computed from primary context (e.g. the distance a user is from home can be computed from two pieces of primary context).

Figure B.1 in Appendix B shows various types of context and their differences depending on your perspective.

2.1.1.5 Levels of context awareness

From a users perspective Dey and Barkhuus[24] identity three levels of context awareness. These are

- **Personalisation:** This is the idea that users can set their own preferences, likes, and expectations manually.
- **Passive context-awareness:** The system constantly monitors the environment and offers the appropriate options to users.
- **Active context-awareness:** The system continuously and autonomously monitors the situation and acts autonomously.

2.1.1.6 Context-Awareness Management Design Principles

There are a number of common design principles used when designing a context-aware middleware application. Perera et al[3] summarise these from a number of sources. A number of these design principles are used when designing the COAcHMAN software. They are

- **Architecture layers and components:** The functionalities need to be divided into layers and components in a meaningful manner. Each component should perform a limited amount of the task
- **Scalability:** Components should be able to be added or removed dynamically.
- **Automatic context life cycle management:** Context-aware software should be able to understand available context sources, their data structures, and automatically build internal models to facilitate them.
- **Mobility Support:** Many devices are mobile therefore context-aware middleware must be able to support multiple software and hardware configurations

- **Monitor and detect events:** Detecting an event triggers an action autonomously in the IoT paradigm. Doing this in real time is a challenge for context aware middleware.

2.1.2 Context Life Cycle

Perera et al[3] provide a summary of many different context life cycles however the one they derive themselves proves to be quite useful for this project. This life cycle is illustrated in figure 2.1 below and figure B.2 in appendix B and and is discussed below in more detail.

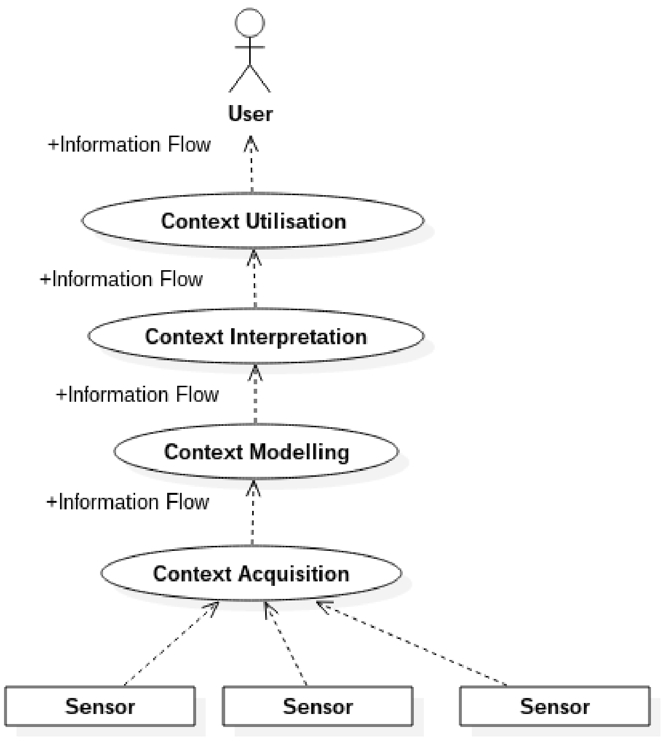


FIGURE 2.1: Context Information Flow

2.1.2.1 Context Acquisition

- **Based on responsibility:** context acquisition is either pushed or pulled from sensors. In the case of pull, a software component is responsible for requesting periodic data updates. In the case of push, sensors are responsible for sending data to the software component.

- **Based on frequency:** this can be either instant (e.g. a door opens) or interval based where data is gathered periodically (e.g. raining).
- **Based on source:** context acquisition can be divided into three categories - data is acquired directly from sensor hardware, data is acquired through a middleware infrastructure, or data is acquired from context servers.
- **Based on sensor types:** this refers to the different sensor types - physical sensors (e.g. temperature sensor), virtual sensors which gather data from many sources (e.g. calendar), logical sensors which combine physical and virtual sensors to produce more meaningful data (e.g. weather information).
- **Based on acquisition process:** there are three ways to acquire context; sense (through sensors and stored in databases), derive (perform computations on sensor data), and manually provided (from users).

2.1.2.2 Context Modeling

Context modeling (also known as context representation) is the method of presenting context in a useful and understood fashion. It can be compared to presenting web site data and structure in HTML where developers create web pages based on a common model. Context models can be either static (the models have predefined fields which need to be populated) or dynamic. This section presents a summary of the different types of context models.

- **Key-Value Modeling:** this model represents context data as simple key-value pairs in various formats (e.g. text files). This is easy to manage when there exists only small amounts of data in the system.
- **Markup scheme modeling:** this models context using tags. This approach brings with it many of the benefits of using a markup language (existing tools, range checking of values).
- **Graphical modeling:** this models context using relationships (e.g. SQL relational database or UML).
- **Object based modeling:** this approach models context using class hierarchies and relationships.
- **Logic based modeling:** this approach uses logical expressions to model context. This has some advantages for high-level context extraction however lack of standardisations means this approach lacks re-usability.

- **Ontology based modelling:** an ontology is defined as *a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents.* [25]. One of the more popular languages for authoring ontologies is OWL (Web Ontology Language - so named by the creators because "Why not be inconsistent in at least one aspect of a language which is all about consistency?").

2.1.2.3 Context Interpretation

Context interpretation is the process of inferring or deducing high level context from low level context data. When you have chosen the appropriate context model there are a number of further steps required to complete the interpretation process. These are

- **Context pre-processing:** this step cleans the collected sensor data. For example there may be duplicate data which is of no value or even missing data.
- **Sensor data fusion:** in this step the data from multiple sensors are combined into a more useful form of information.
- **Context inference:** this step deduces high level context from low level context data.

In the next number of paragraphs we examine a number of approaches to context reasoning

Supervised Learning This approach firstly collects sample context information and from this data derives a function for generating expected contexts from the collected data. A form of supervised learning used by COAcHMAN (which will be discussed in more detail later) is that of a decision tree.

Unsupervised Learning Using this method the system has no training data available and makes use of clustering techniques to organise data into context.

Rules This is a simple method of deriving context from data. The rules are normally structured in a familiar IF-THEN-ELSE format.

Fuzzy Logic This is a form of probabilistic reasoning which incorporates imprecise notions (e.g. fairly hot or somewhat tall). This approach is normally used in combination with other types of context reasoning.

Ontology Based This is based on some semantic web language (we briefly touched on OWL earlier). In their paper Wang et al[26] propose a context ontology (CONON) which allows developers to model context using and ontology. Figure B.3 in Appendix B shows an example of a serialised ontology.

Probabilistic Logic This approach allows decisions to be made based on probabilities attached to the data related to the problem. An example of this might be bridging the gap between raw GPS data and destination using previous calendar data to determine destination. The system might assign a high level of probability to a destination if it had been in a users calendar previously and closely aligns with the raw GPS data.

2.1.2.4 Context Utilisation

Context utilisation (or context distribution) is the task of delivering context to the user. This might be in the form of facilitating queries to the system of by providing a method to which a user can subscribe to context data or by providing actions the user depends on.

2.1.3 Literature Review

The majority of this section is taken from the paper "Context-aware systems: A literature review and classification" by Hong, Suh, and Kim[27] where they survey the available research on the topic of context awareness for the years 2000-2007. While the paper may appear out of date the findings they make appear (at least anecdotally) to be still relevant.

The main findings made in this literature review are as follows

- There is a common abstract architecture of context-aware systems. This is illustrated in figure B.4 in Appendix B.
- The authors constructed a classification framework of context-aware systems (illustrated in figure B.5 in Appendix B). This proves useful in understanding where various research fits into.

- The article classification highlights the main applications of context-aware systems. These are, in order of number of articles

  1. Smart space (including smart homes, smart hospitals, and class rooms)
  2. Tour guide systems
  3. Information systems
  4. Communication systems
  5. Mobile commerce
  6. Web services

  One may speculate this order might have changed in recent years due to the surge in mobile commerce and web service however there is not a more recent literature review to reference.

- What questions remain unanswered? These include what design patterns are useful for developing context-aware systems, what is the best algorithm for inferring user context, how to solve the problem of conflicting context and how to evaluate performance of context-aware systems. These questions would provide potential research topics in the future.

2.2 Context Aware Architectures

In this section we will discuss a number of existing context-aware systems with the intention of identifying how they applied the theory that has been covered in previous sections and what issues the authors discovered and overcame. This architecture analysis proved useful for development of COAcHMAN.

2.2.1 OSGi-based context-aware architecture

In the paper “Enabling context-aware Smart Home with Semantic Web Technologies” [28] Zhang, Gu, and Wang propose an OSGi-based context-aware service architecture for smart homes that make use of logic reasoning to enable context-aware services to deduce high-level implicit context from low-level, explicit context. To achieve this they propose a generic layered model called the Context Stack. This layered model combines context acquisition, model and representation, aggregation, interpretation, and utilization into a generic architecture. The five layers of the context stack are as follows

- **Context Acquisition Layer:** this is the lower level in which context is in a raw format and is acquired from a wide variety of ubiquitous context sensors.
- **Context Representation Layer:** the layers above this layer rely on a common context model. Therefore the purpose of this layer is represent raw sensor data as a machine readable model. The authors use a general object-oriented context model for this purpose.
- **Context Aggregation Layer:** this layer aggregates and relates contexts from distributed context sensors to form a centralised context database.
- **Context Interpretation Layer:** this layer aims to make use of reasoning to deduce high level context needed by upper layer services from lower level context data.
- **Context Utilisation Layer:** At this layer users devices will be directly affected by the context gathered by this system.

As well as proposing the context-stack, the authors define a web ontology (CONON) to facilitate context representation and logic reasoning. As web ontologies are a detailed topic they are covered elsewhere. After discussing the ontology the authors discuss how they make use of the ontology for various forms of logic reasoning. Finally, they describe their implementation which is based on an OSGi framework (OSGi is discussed in greater detail in the next chapter). The framework implementation chosen is ProSyst's mBededServer. The low level context data is acquired from temperature sensors and light sensors and high level context data is provided by a users calendar. They propose a number of uses for their prototype.

2.2.2 SOCAM - A service-oriented middleware for building context-aware services

In the paper - "A service-oriented middleware for building context-aware services", Gu, Pung, and Zhang[29] propose an architecture which they claim provides for the building and rapid prototyping of context-aware services. Similar to other approaches they make use of OWL for context modelling and also use first-order predicate calculus to represent context. For example, *Location(Bob,work)* means Bob is at work or *Status(door,open)* means the door is open. The authors also classify context into two main categories - Direct Context and Indirect Context. This refers to the means by which the context is obtained i.e. Direct Context is directly acquired and Indirect Context is derived by interpreting direct context via context reasoning. They further break down Direct

Context into Sensed Context (e.g from a sensor) and Defined Context (e.g. specified by a user).

2.2.2.1 SOCAM architecture

The SOCAM architecture illustrated in figure B.6 in Appendix B is a reasonably familiar architecture at this stage. It is divided into the following layers

- **Context Sensing Layer:** This consists of virtual and physical sensors
- **Context Providers:** These extract useful contexts and convert them to OWL representations. A major enhancement suggested by this paper is to make use of video analysis to perform object tracking and behaviour analysis to derive high level descriptions of events within the home.
- **Context Interpreter:** This provides reasoning services which process context information. The authors propose two methods of reasoning; ontology reasoning and user defined rules based reasoning. Ontology reasoning is claimed to be useful due to the transitive properties available (e.g. John is in the living room and the living room is in the home therefore John is in the home). The User-defined rule-based reasoning proposed in this paper is based on a programming engine similar to Prolog.
- **Context Database:** Stores context ontologies and past contexts.
- **Context-aware Services:** Makes use of different levels of contexts and adapts the way the system behaves according to the current context.
- **Service Locating Service:** This provides a mechanism where context providers can advertise their presence and it also facilitates users locating these services.

The implementation is based on Java RMI running on Linux and supports various networking technologies (e.g. Ethernet, ADSL, Bluetooth).

2.2.3 CoCaMAAL - A cloud-oriented context-aware middleware

CoCaMAAL is described by the authors of "CoCaMAAL: A cloud-oriented context aware middleware in ambient assisted living" - Forkan, Khalil, and Tari[30] - as a model for addressing issues of management of sensor data, derivation of contextual information as well as monitoring user activities with an aim of identifying appropriate situational services. The focus of this paper is slightly different from the others in that it focuses on AAL while using context aware services to achieve this aim. Therefore Body Sensor Networks (BSNs) play a more important part in this architecture. As can be deduced by the title of the paper this architecture depends heavily on cloud services for provision of simplified user access and demand elasticity.

The CoCaMAAL system overview consists of the following components

- **AAL Systems:** These are a network of AAL sensors (e.g. heart monitor, temperature monitor) which are generating raw sensor data which is inputed to higher level context generation services.
- **Context Aggregator and Providers (CAP):** The CAP cloud contains computing logic and methods of converting low-level raw data to abstract context representation which is recognisable to all components of the architecture.
  - **Context Aggregator:** The context aggregator is the main collaborator in this model. It is a distributed cloud service responsible for collecting incoming sensor data from the various AAL systems, aggregating all the context information into a single context model, separating the data for sending it to various context providers and for tracking the identification of AAL systems.
  - **Context Providers:** These providers convert sensor data into context. Each of the context providers runs as a cloud service which collects raw sensor data as input and responds back with classification output using a SOAP-based web service. Various methods of classification methods can be used (e.g. Decision trees, unsupervised learning).
- **Service Providers (SP):** SPs are applications related to context awareness. For example, an SP can be an application running on a mobile device of an AAL system that reminds the user of appointments or it could be a caregiver application that monitors for emergency situations.
- **Context-aware middleware (CaM):** The CaM cloud has the infrastructure for processing context data, sorting and retrieving context, access control, and many more computational tasks.
- **Context data visualisation:** Context data contains valuable information of users. There exists a requirement for a GUI for examining these records.
- **Context Modelling:** The context model is used to represent the context data so that any concerned application can use it. It is formed by the aggregator cloud

after identification of key contexts. CoCaMAAL uses an ontology based context model using four major entities - Person, Place, Environment, and Device.

Also of note is some of the sample assistive services used as examples. These are relevant examples to the COAcHMAN software as well as providing sample rules. Some sample rules can be found in table B.1 in Appendix B.

2.2.4 Ambient Home Care System - Context-aware middleware for ambient assisted living

The next example of context-aware middleware - AHCS - is an ambient home care system designed to promote independent living in old age. The use cases presented by Hristova, Bernardos and Casar[31] are heart-rate monitoring, medication prompting, generation of agenda reminders, and emergency notifications both for the elderly and for their caregivers. The system monitors location of people within the home, environmental state of devices with in the home via sensors, calendar information, and the heart rate of the user.

To implement the AHCS the authors make use of a context-aware framework called the Context Toolkit developed by Salber, Dey and Abowd[32]. The main actors in the Context Toolkit are

- **Widget:** This is the software piece that encapsulates each sensors data and communicates when the data changes or satisfies a certain rule.
- **Aggregators:** These combine the context information from several widgets and infer higher level data.
- **Interpreters:** These translate certain information from low level to higher levels.
- **Discoverer:** This component provides a discovery mechanism in the system. It contains a registry of all components. Applications can query the discoverer for information on how to locate desired components.
- **Enactor:** This enables applications developers to define application logic.

2.2.4.1 Architecture of AHCS

The architecture of the AHCS is broadly similar to other context aware systems we have seen to this point. Low level context is gathered via a framework called CASanDRA which provides reusable acquisition services. The authors define three types of sensors used to gather context data, these are

- **Virtual Sensors:** These can be accessed via web services such as Google Calendar or Google Weather.
- **Physical Sensors:** These are hardware based sensors e.g. heart rate monitoring.
- **Logical:** These type of sensors include data inferred from log files

The data acquired using CASanDRA is used by the Context Toolkit (CTK). This toolkit provides most of the tools required to create context from data and distribute it however there still exists the need to create a reasoning engine. The reasoning engine implementation used by the authors is a rule based engine chosen in conjunction with key-value pair context models. It is based on "if-then-else" rule statements.

2.2.5 Analysis of architectures

The analysis of the architecture will be with respect to the context aware fundamentals. The primary aim of this analysis is to examine how the different context aware designers approach their implementations using the fundamental theory. This will provide some reference during the COAcHMAN design stage. The second aim of the analysis is to identify relative strengths and weaknesses of the implementations so as to avoid the weaknesses in the COAcHMAN system as well as being able to take advantage of the strengths. Therefore the analysis will be in the following areas: definitions of context, context data & information, system architectures, context models, context interpretation, and context utilisation.

2.2.5.1 Definitions of context

Table 2.2 shows the definitions of context used by the various systems. What is obvious is that all systems use different definitions of context depending on the application of the system. Context-aware systems targeted at AAL include user physiological data for context. Also worth mentioning is the fact that not all systems clearly define context.

System Name	Definition
OSGi-based	No explicit definition however refers to sensors, location, activity, and user status
SOCAM	Any information that can be used to characterise the situation of an entity, where an entity can be a person, place, or physical or computational object
CoCaMAAL	No explicit definition but references to ambient readings, physiological state, location
AHCS	No explicit definition but references location, environmental state, calendar, heart rate

Table 2.2: Definitions of context

2.2.5.2 Context Data & Information

This section refers to the types of sensors and data gathered by the systems. The data collected is presented in table 2.3. Once again we can see the definitions vary in quality. The authors of the AHCS system have provided good clear conceptual descriptions of the sensors they are using. Therefore it would appear this definition should be used when designing a context-aware system.

System Name	Definition
OSGi-based	Raw sensor data & calendar information
SOCAM	Sensed Context - Physical sensors & virtual sensors (e.g. web services) Defined Context - User preferences
CoCaMAAL	Physiological sensors, environmental sensors, cloud services
AHCS	Virtual sensors (cloud services), physical sensors (ambient data), and logical sensors (data inferred from log data)

Table 2.3: Context Data & Information

2.2.5.3 System Architectures

Table 2.4 gives a brief summary of the various architectures employed by the surveyed systems. There does appear to be a general pattern for developing context-aware systems, namely data acquisition, context representation, reasoning and interpretation, and finally utilisation or application. Therefore it would appear sensible to consider this pattern when developing a context-aware home automation middleware.

System Name	Definition
OSGi-based	Acquisition Layer, Representation Layer, Aggregation Layer, Interpretation Layer, and Utilisation Layer
SOCAM	Context Sensing Layer, Context Middleware Layer (reasoning, database and interpretation), and Context Application Layer
CoCaMAAL	Sensor data, Context aggregator and providers cloud, data visualisation and social network cloud, and service providers cloud
AHCS	Sensing (CASanDRA), Context Acquisition, Reasoning and decision

Table 2.4: System Architectures

2.2.5.4 Context Models

The popular method of representing context (illustrated in table 2.5 is clearly an ontology of some sort (most popular is some form of web ontology). Any context-aware system would have to consider using OWL or some other web ontology for representing context.

System Name	Definition
OSGi-based	Based on Web ontology (CONON)
SOCAM	Based on Web Ontology Language (OWL)
CoCaMAAL	Based on Web Ontology Language (OWL)
AHCS	Context Toolkit Widgets

Table 2.5: Context Models

2.2.5.5 Context Reasoning

As demonstrated in table 2.6, a popular method of context reasoning is by making use of rules e.g. *If condition X then Y else Z*. The main advantage of this approach is simplicity.

System Name	Definition
OSGi-based	First order logic (if-then-else)
SOCAM	Ontology reasoning & rule based
CoCaMAAL	Rule based
AHCS	Rule based

Table 2.6: Context Reasoning

2.2.6 Analysis Conclusion

Certainly a number of clear trends are found in analysing the OSGi-implementation, SOCAM, CoCaMAAL, and AHCS. These are

- They do not always have a clear definition of what they consider context to be.
- Context data is comprised of data from various types of sensors, not all are physical sensors.
- A similar architecture is employed by all
- The majority use some form of web ontology to model context
- The majority use some form of rule based reasoning system

These findings will prove useful in the design phase of COAcHMAN however, as is demonstrated by the systems that were analysed, the application area and other factors play a role in the design decisions also.

Chapter 3

Home Automation Technologies and Protocols

We briefly introduced smart homes and home automation in Chapter 1, however in this chapter we are going to explore what protocols, technologies, and middleware are available. At the end of this background research a designer will be able to identify any protocols or technologies that will aid in the development of a context aware home automation system.

3.1 Home Automation Protocols and Technologies

The term "smart home" is first used in 1984[33], however examples of home automation go back as far as the world fairs of the 1930s. While Bernheim et al[11] have already discussed why the technologies have yet to be widely adopted, recent discussions on ageing populations suggest that this may change[34].

3.1.1 Overview of available technologies and protocols

As a general rule the technologies and protocols go hand in hand, i.e. if we refer to the protocol TCP then the transmission technology could be wifi or ethernet however if we refer to the protocol zigbee then the transmission technology will be a zigbee device. It appears that generally smart home technologies rely on dedicated protocols, although as we will see in the next section there exists a large number of them to choose from.

3.1.2 Comparison of available technologies

To determine which technology is suitable for a project an analysis based on the following topics is helpful

- Is it wireless or wired technology? Or both?
- Is it based on an open or proprietary standard? If the standard is proprietary perhaps is is free to use?
- What type of sensors and switches are supported?
- Are software libraries readily available?

A good deal of the information is based on reading any documentation provided by the manufacturer or standards body.

BACnet BACnet is a "data communication protocol for building automation and control networks". It is based on a client-server model and is an ANSI & ISO standard. There does appear to be some open source libraries for using the BACnet protocol however a quick search of amazon.co.uk shows no available BACnet devices. It would appear that regardless of the benefits of this protocol there would be significant effort involved in integrating this into any project.

C-Bus C-Bus is a protocol for building automation that uses Cat-5 ethernet cable. It enjoys some popularity in Australia, Russia, the Middle-East and Asia. Similar to BACnet the overhead in integrating this protocol would likely to be too great for a medium size project.

CC-Link CC-Link is an open-architecture network. It operates primarily on ethernet. A brief investigation into the products available show that the target for these protocol is more industrial automation rather than home automation.

DALI DALI (Digital Addressable Lighting Interface) is targeted toward lighting automation. It requires additional wires for communication and can be arranged in a bus or star topology. Products, software, and documentation are scarce.

EnOcean EnOcean is an energy harvesting wireless technology for building automation. It is primarily a wireless technology and EnOcean sensors and light switches perform without batteries by gathering energy from sources such as solar power or electromagnetic energy. A group of companies have grouped together to form the EnOcean alliance which is dedicated to standardising the technology. It is reasonably easy to acquire and use software libraries. There are a number of wall switches, sensors, and alarms available for purchase.

KNX KNX is an OSI based standard for building automation communication protocol. The physical media can be wired (ethernet or power line) or wireless (radio or infrared). There are a large number of KNX compliant devices available for purchase. Software support for KNX development appears to be difficult to obtain.

Modbus Modbus is a popular communication protocol for connecting industrial electronic devices. Originally developed in 1979, it is designed to be royalty free, openly published and easy to deploy. A brief investigation of the products available that use this protocol reveal that it is very much industrial applications which make use of Modbus, there are few devices which would be suitable for home automation.

oBIX oBIX (Open Building Information Exchange) is a standard for creating web services based interfaces for building control systems. There is some interest in using this standard in conjunction with smart grids as it will facilitate communication between buildings and energy providers. As it is a standard for control systems it may not be applicable for a context aware system.

VSCP VCSP (Very Simple Control Protocol) is a free automation protocol for building automation. It is independent from the physical media which means that it can function with many different devices (e.g. CAN bus, ethernet, wifi devices). There exists a software api - “VSCP & Friends”. While the standard and software is readily accessible there is a limited number of devices which can make use of the protocol (only two are wifi enables, most are CAN based).

xAP xAP (eXtensible Automation Protocol) is an open protocol designed to facilitate communication between devices which normally are incompatible. A brief investigation reveals that this protocol does not support many available devices.

X10 X10 is one of the oldest protocols used in home automation. Developed in 1975 it is primarily power line based (which has the advantage of being able to make use of a buildings existing power infrastructure). It is open source and has a large number of available devices.

Z-Wave Z-Wave is primarily a wireless technology designed to allow devices to communicate with each other. There a number of software libraries available and it has a large number of devices readily available. The Z-Wave Alliance consists of 250 technology manufacturers.

6LoWPAN 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) is an IETF working group set up with the aim of getting low power devices using IPv6. While there are no devices supporting this protocol currently, it would appear that this protocol could be widely used in the future.

Zigbee Zigbee is a specification for a number of high level communication protocols. It is primarily a wireless technology. For non-commercial purposes the zigbee specification is free and there is good support from a number of organisations (e.g. Texas Instruments). There are many devices available for purchase that support zigbee (e.g. sensors, switches, alarms).

INSTEON Insteon is a home automation technology supporting many different devices. It supports multiple transmission media (i.e. power line, RF). The company that manufacturer Insteon chip set - Smartlabs - have recently announced collaborations with Microsoft and Google. There appears to be easy access to software and devices.

LightwaveRF LightwaveRF is a Wifi based home automation technology. There are many devices available and there appears to be some software support available.

Conclusion As is immediately apparent there are many competing standards and technologies which appear to have very similar functionality and features. The devices themselves are largely in line with competitor devices with regard cost. Some technologies do appear to enjoy more adoption and support than others; Z-Wave, Zigbee, X10, KNX, EnOcean, Insteon, and LightwaveRF are the main players in this space. Therefore it would appear that any implementation of a context-aware home automation system should be compatible or support some or all of these technologies.

3.2 Home Automation Middleware

The aim of this project is to provide a context-aware home automation system therefore there needs to be awareness of home automation technologies however this project is not responsible for low level tasks such as protocol support for devices. As a consequence of this rather than developing a new home automation system it would appear that enhancing an existing home automation system is an appropriate approach. Home automation systems which provide communication services are termed “home automation middleware”. To determine which home automation middleware is most suitable to our goal it is worthwhile conducting an analysis of the available systems.

3.2.1 Comparison of available middleware

In “Which AAL Middleware Matches My Requirements? An Analysis of Current Middleware Systems and a Framework for Decision-support”[35] Zentek et al evaluate 43 AAL middleware systems and summarise the top 10. It is this list which is examined for suitability for this project.

1. **Contronics:** Main drawback is that in can only control devices which use the FS20 or HomeMatic protocol. Committing to one or two specific protocols is a high risk.
2. **FHEM:** Based on Perl, can run on many devices. Last release was nearly a year ago, but does support many protocols. This is a good candidate.
3. **iP-Symcon:** Stable and widely used, based on PHP. Appears to run on windows only, source code does not appear accessible.
4. **LogicMachine:** Supports many protocols (EnOcean, KNX, Modbus). Very much a hardware product, source code does not appear accessible.
5. **Loxone:** Supports many different protocols however is very much a commercial product, source code does not appear accessible.
6. **OpenRemote:** Many major protocols are supported, good community support and possible to modify source code (Java based). This is also a good candidate for this project.
7. **openHAB:** Written in Java and based on OSGi, has support for a number of protocols. The Eclipse SmartHome project shares code with openHAB. Also has a number of mobile clients. This is a good candidate for this project.

8. **Open Source Automation (OSA):** Developed in C# and uses .Net framework. Supports many major protocols. Authors note that OSA appears unstable and they were unable to install this correctly.
9. **universAAL:** Written in Java and based on the OSGi framework, this middleware supports a number of protocols. This appears to be a possible candidate.
10. **Vera:** Tied to proprietary hardware.

Smirek, Zimmermann, and Zieglar[36] compare MyUI, Universal Remote Console (URC), and openHAB. Their comparisons provides the following findings

1. **MyUI:** This middleware has the strength of being "close to the user" in that the system is able to dynamically adapt the user interface to the systems context. It achieves this based on manually created patterns and the authors note that there are only a limited number of patterns available.
2. **URC:** The main idea of URC is that any device can be controlled from any controller that best fits the users needs (e.g. in a hotel room every guest could use their smart phone to control the TV). The system supports various protocols (KNX, UPnP) however the authors note that only a few target adaptors and discovery modules are available currently.
3. **openHAB:** Support for multiple protocols, developers can add new functionality through **bindings**. The authors comment that openHAB suffers from a lack of support for appropriate user interfaces.

These two papers present a number of middleware for consideration. Any middleware that makes use of the OSGi framework would appear to be the best approach as it has a well defined method of adding new functionality. Furthermore, support for multiple protocols is important as the project should not suffer due to some protocol specific issue i.e. if there is an issue with one protocol it should be possible to discard it for another choice. This narrowed the options to OpenRemote, openHAB and universAAL. openHAB however appears to have strong support for cloud services (i.e. Google Calendar, weather services, Dropbox) as well as support for wearable computing devices (i.e. Samsung Gear smart watch).

3.2.2 OSGi

Before examining openHAB in detail we need to educate ourselves on the framework it is build upon - OSGi. In their paper *A Gentle introduction to OSGi*[37] Tavares and

Valente give an overview of OSGi. OSGi was designed to address the problems of highly coupled “spaghetti-like” systems that result from a lack of restrictions around accessing public classes in Java packages and the problem of updating modules at deployment time only (the latter being an issue felt in domains such as consumer electronics or automation).

In 1998 the OSGi Alliance proposed a framework which provides for dynamically adding and removing existing modules (called bundles). These bundles are independent modules which provide well-defined services. The framework also controls the life-cycle of bundles which include adding, removing, and replacing bundles at runtime. The main implementations of the OSGi specification are; Knopfler-fish, Apache Felix, and Eclipse Equinox. The OSGi runtime and installed bundles exist within one Java Virtual Machine. One immediate advantage to this approach is the ability to run multiple applications within the one JVM therefore minimising the memory usage and facilitating low cost inter application communication.

Another important component of the OSGi runtime is the Service Registry. This registry maintains a record of the services registered for use within the OSGi framework. The registry provides the mechanism for bundles to publish their services and also a method of retrieving services. The OSGi framework makes use of a *Whiteboard Pattern* to facilitate client bundles making use of services from other bundles. This is achieved by bundles publishing their services to the framework and other bundles that are interested in making use of services (listeners) can simply subscribe to the registry.

To communicate to the service registry the services a bundle provides a developer must complete a specific XML file which the framework understands. When the bundle is introduced to the system to framework reads the services from the XML file however no instance of the bundle is initiated. This only occurs when a requested method is executed. After the method is complete the module returns to a standby state. Figure B.7 in appendix B illustrates the relationship between bundles, the service registry, the OSGi framework, and the JVM.

Certainly, based on the Tavares and Valente paper, it would seem a reasonable choice that developers would use an OSGi framework to build their home automation system on. If using a small and hardware constrained server to act as the home controller having only one instance of a JVM running has obvious benefits. Add to this the ability to publish additional services without a system restart plus providing developers a common interface to build bundles and one can see why so many smart home systems have used the OSGi framework.

3.2.3 openHAB Overview

A good deal of this overview comes from the seminar paper “openHAB - Empowering the Smart Home - History, Concepts, Examples” by Manuel Raffel[38]. openHAB (open Home Automation Bus) acts a central controller for a smart home. Originally developed by Kai Kreuzer, it currently is at version 1.7 supports many smart home technologies and protocols such as KNX, EnOcean, Z-Wave and X10 as well as many cloud services such as Google Calendar, Dropbox, and various online weather services.

3.2.3.1 Architecture

The architecture of openHAB is illustrated in figure B.8 in Appendix B. It is worth noting that each unit in figure B.8 represents an OSGi bundle. In order to gain an understanding of the system the following paragraphs describe each section of the architecture

Core The core bundle represents the centre of the openHAB middleware. It contains definitions for all other bundles to build on. In the core there exists the following basic concepts

- **Items:** An Item is an abstract object that can be read from or written to. An example of an Item could be a Zigbee socket.
- **Event Bus:** Two types of events are transmitted on the event bus - Commands and Status updates.

Repository The repository provides a central mechanism for keeping track of items state information. It avoids the need for each bundle to keep track of each item. The repository listens on the event bus and updates stored items according to status updates.

REST Service This offers an API for various purposes. It facilities the sending of commands to an item (for example from a remote client) or to allow a client to subscribe to an Item. The client will then get automatic updates of the subscribed items resources.

User Interfaces The UI of openHAB is built on sitemaps. A sitemap is a text file describing the items in a tree structure. There exists an Android client, iOS client, and a Web UI to interact with openHAB.

Automation Logic The automation logic consists of scripts and rules. Scripts are blocks of code that can be reused in different places. Rules consists of triggers (i.e. the condition that starts the execution of a rule) and actions.

Protocol Bindings Bindings are what provides the ability for different kinds of technologies to communicate on the openHAB event bus. There is a guide for creating protocol bindings and as of July 2015 there are 122 distinct bindings.

Item Provider An item provider defines what widget (a widget is a connection between an Item and the UI) is to be used for a particular item. This allows for dynamic changing of UI attributes.

3.2.3.2 Configuration

The openHAB server takes a number of user specific parameters from a file called *openhab.cfg*. On installation a user would populate this file with any information relating to bundles (e.g. API key for an online weather service).

3.2.3.3 Persistence

To ensure data is not lost in the case of a system failure openHAB supports multiple forms of persistence. Examples of supported persistence are MySQL, MongoDB, InfluxDB or simple Logging.

Chapter 4

COAcHMAN Design & Implementation

This chapter documents the main effort of this project. As we now have an understanding of the problem domain, the tools and theory available and have identified some common patterns for context aware development we can now put it into practice.

To achieve the end goal we firstly attempt to gain a basic understanding of the problem we are trying to solve and the high level design of the solution. The result of this task is a set of requirements, a system specification, and a set of use-cases which can be used to validate our requirements and also to be used as a set of test cases for the COAcHMAN system.

Once we have a set of requirements to work from the next stage is the architectural design of COAcHMAN. To achieve this goal we will need to document the architectural styles and patterns employed, construct appropriate UML models, identify data flows in the system (using data flow diagrams), and produce a design specification of the system.

Following on from the architecture design phase we document the software implementation, the technical challenges discovered and overcome, and the various external interfaces and software which were interacted with.

Finally we present a number of real-world use cases and how they perform. These use-cases will validate that our requirements have been met and serve as a useful base for any demonstration of the project.

4.1 Project Requirements

In this section we will explore why COAcHMAN is needed, under what circumstances it is expected to run, who will want to use COAcHMAN, and what will be required of the system.

4.1.1 Inception

In "Software Engineering - A Practitioners Approach" Roger Pressman[39] presents the very first stage of a software project - inception - as a step where we need to "establish a basic understanding of the problem, the people who want a solution, and the nature of the solution that is desired". To that end it is useful to have a clear problem statement.

All context-aware home automation systems developed thus far are presented as academic research. Any person who wishes to enhance their home via home automation cannot use these systems as they are not readily available nor implemented with software development techniques.

The problem statement highlights the missing step in all the context-aware systems we examined previously, namely the leap from theoretical to implementation. So while their systems may have a prototype to demonstrate the theory presented in their research, they did not develop the prototype with any deployment or general release in mind. Therefore there is a shortage of context-aware middleware (this is explicitly mentioned by Perera et all[3] where their research has shown that "the majority of the IoT middleware solutions do not provide context-awareness functionality") and also the lack of useful techniques for developing context-aware middleware (Hong et al[20] explicitly mention the fact very little research has been conducted into design patterns for context-aware systems).

The users of this project are expected to be the inhabitants of a smart-home. While the area of AAL is possible a beneficiary of the project the users of AAL systems are not the primary target of this solution.

Bearing in mind therefore that we wish to develop a system with users in mind it appears make sense to align the development with an already existing system to eliminate any overlapping work.

4.1.2 Usage Scenarios

To develop a set of use cases we can follow Roger Pressmans steps[40] which are to develop a set of actors and then to answer the following questions with the actors in mind

- Who are the primary actors?
- What are the actors goals?
- What preconditions should exist before the story begins?
- What main tasks or functions are performed by the actors?
- What system information will the actors acquire, produce, or change?
- Will any of actors have to inform someone about changes in the external environment?

The actors involved are the inhabitants of the home, the home automation system itself (as this is expected to have some autonomy it should be regarded as an actor in its own right) and any visitors to the home. Some sample Use-Cases can be found in figure C.1 in Appendix C. As can be noticed by the use cases they focus very much on the users activity and location outside the home. This is an explicit design choice as the time and effort to implement location sensitive context updates within the home (e.g. determining if a user has fallen) was deemed too high for the duration of this project.

4.1.3 Requirements

The requirements derived from the use-cases in the previous section are divided into functional requirements and non-functional requirements. The definitions used for both are taken from “Software Engineering” by Ian Sommerville[41]. Some of the vocabulary used is defined by RFC2119 - Key words for use in RFCs to Indicate Requirement Levels.

4.1.3.1 Functional Requirements

In this section we provide a set of requirements which define the services that COAcHMAN must provide, how it reacts to particular inputs, and how it is expected to behave in certain situations. The requirements were created with some reuse from *Exploring the responsibilities of single inhabitant smart homes with use cases* by Lyons et al[42].

1. The system MUST support location tracking of users
2. The system MUST be able to deduce a users context
3. The system MUST support activity tracking of users
4. The system MUST be able to react to changes in users context
5. The system MUST NOT allow a user to be without a context
6. The system MUST allow users to specify how the system behaves when presented with updated context information
7. The system MUST be able to handle conflicting context information

4.1.3.2 Nonfunctional Requirements

These requirements are concerned with constraints on the services offered by the system. They relate to performance, security, and availability.

1. The system MUST support an arbitrary number of users
2. The system SHOULD require credentials for user access
3. The system SHOULD store context information for a period of time to facilitate context learning
4. The system SHOULD function with minimal user interaction

4.2 Architecture Design

Figure 4.1 shows how COAcHMAN integrates with openHAB. At this point we can take advantage of the research carried out earlier to derive a high level architecture for COAcHMAN. This is illustrated in figure 4.2. A number of aspects are worth mentioning; the context utilisation is in the form of a rules system which is already present in the openHAB system itself and the fact that the architecture is reasonably similar to the SOCAM Architecture (see figure B.6 in Appendix B).

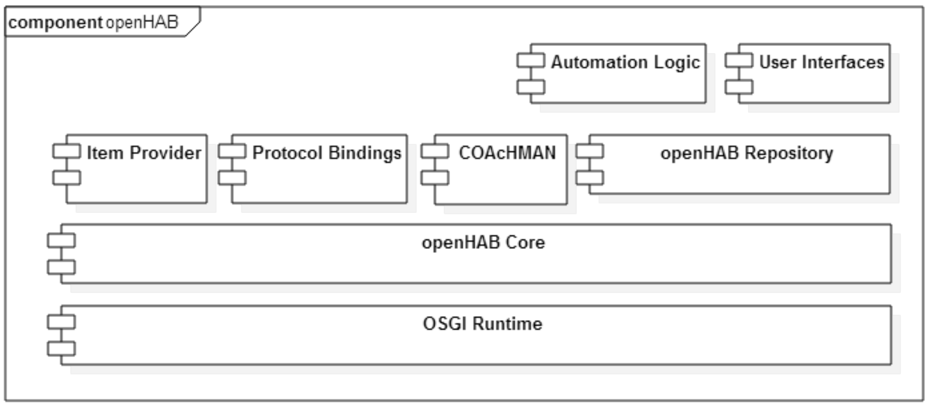


FIGURE 4.1: COAcHMAN integration with openHAB

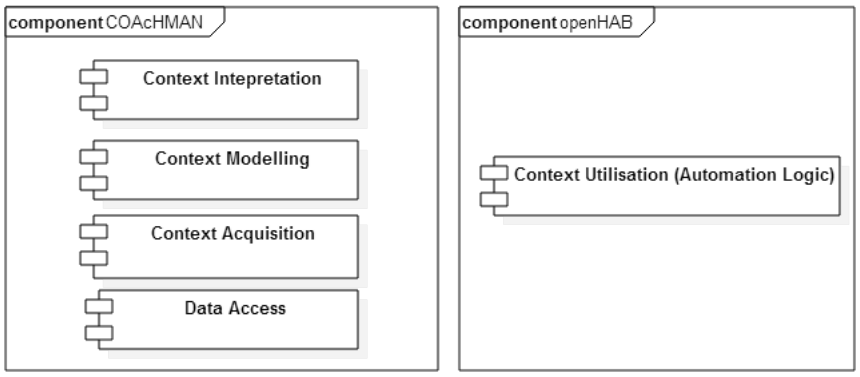


FIGURE 4.2: COAcHMAN Architecture

4.3 Implementation

This section gives an overview of how the COAcHMAN architecture is implemented. The order of discussion is from the lowest level in the architecture (i.e. data access) up to the highest (context utilisation).

4.3.1 Data Access

The data access component is implemented using a *Data Access Object (DAO) Pattern*[43]. The sensors we use most are virtual sensors for accessing users location and activity. The advantage of the DAO pattern is the abstraction and encapsulation it offers for accessing these virtual sensors. There exists only one instantiation of each object in the system, this is enforced through usage of the Singleton Pattern.

The three Data Access Objects implemented in COAcHMAN are

CalDAO The Calendar Data Access Object provides user activity information. Currently it supports only one function shown in figure 4.3. This implementation uses Google Calendar to access user events. The clear advantage of this is that it is a familiar interface.

```
/*************************************
 *
 * @param u - User object
 * @return Event - Current event for User u
 */
public Event getCurrentEvent(User u);
```

FIGURE 4.3: CalDAO Main Function

GeoDAO The Geographic Data Access Object provides user location information. The two functions supported are shown in figure 4.4. This implementation uses Google Maps API providing this service. Note, the GeoDAO is not responsible for locating users. This information is provided by the users mobile device (discussed in subsection 4.3.2).

```
/*******************************************
 * This function makes use of reverse Geocoding
 *
 * @param l - A Location object
 * @return A String representation of the
 *          Location object
 */
public String getOriginAddress(Location l);

/********************************************
 * This function makes use of Geocoding
 *
 * @param address as a String
 * @return longitude and latitude of the address
 */
public double[] getCoordinates(String address);
```

FIGURE 4.4: GeoDAO Main Functions

SQLDAO The SQLDAO offers a simple method for accessing COAcHMAN persistence. The functionality offered is shown in figure 4.5. This implementation uses MySQL.

```
/* This function stores a Context object into a database */
public boolean logUserContext(Context c);

/* Get the current Location for a user */
public String getUserLocation(User u);

/* Logs an event associated with a user. The events
 * location and summary are stored */
public void logEvent(User user2, String summary, String location, double lat,
    double lng);

/* Gets all locations of a certain category for a user */
public ArrayList<double[]> getLocations(String category, User u);

/* Generic execute select */
private ResultSet executeSelectQuery(String query);

/* Generic execute insert */
private boolean executeInsertQuery(String query);
```

FIGURE 4.5: SQLDAO Main Function

4.3.2 Context Acquisition

This layer in the architecture is concerned with reading values from the available sensors and providing the readings to the layers above. The sensors are accessed via the DAO however the sensors themselves get their data from other sources. The architecture of this layer is shown in figure 4.6. Also discussed in this section is the HABDroid client which provides location information to the COAcHMAN service.

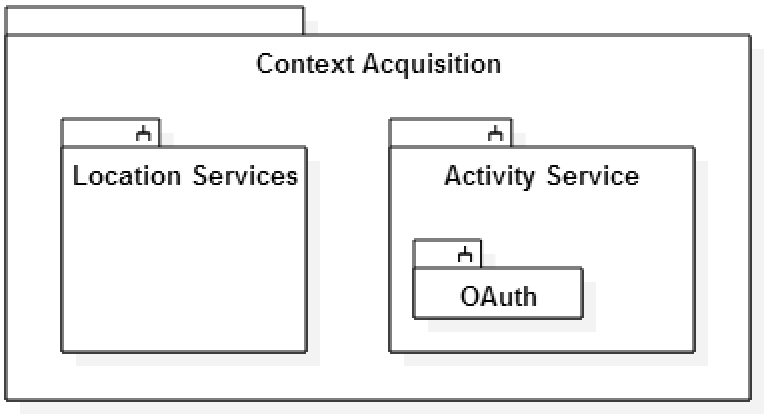


FIGURE 4.6: Context Acquisition Layer Architecture

HABDroid HABDroid is the open source Android application which provides a user interface to the openHAB server. It seemed appropriate therefore to modify this application to provide users location. This interaction is illustrated in figure 4.7. The location update functionality in the client is implemented via an *Intent*[1] which allows us to execute code which fetches the most recent known location (in longitude and latitude form) for a user and send the co-ordinates along with the users unique SIM card number plus user name to the openHAB server.

Also the app can be modified to display the current context of all the users of the system. A sample of this is shown in figure C.2 in Appendix C. The modified HABDroid source code is located on github at `https://github.com/aidanom1/openhab.android/tree/master/mobile`

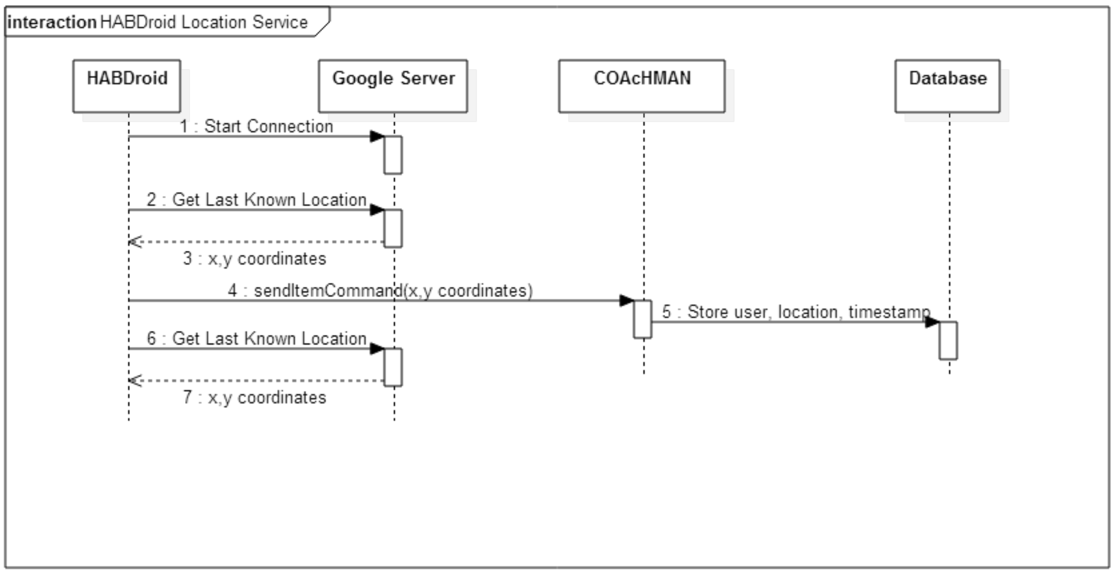

interaction HABDroid Location Service
HABDroid
Google Server
COAcHMAN
Database
1 : Start Connection
2 : Get Last Known Location
3 : x,y coordinates
4 : sendItemCommand(x,y coordinates)
5 : Store user, location, timestamp
6 : Get Last Known Location
7 : x,y coordinates

FIGURE 4.7: HABDroid Sequence Diagram

Location Services The Location Services packages uses the Google Maps API to provide data to the location virtual sensor. The virtual sensor is illustrated in figure 4.8

Activity Services The Activity Service component (illustrated in figure 4.9) provides a virtual sensor for reading a users current activity. As well as that there is an OAuth package (adapted from a third party implementation for Google Drive) which allows COAcHMAN access a users Google Calendar. It uses the "OAuth2.0 for devices" method

[1]An Intent is a messaging object you can use to request an action from another app component.

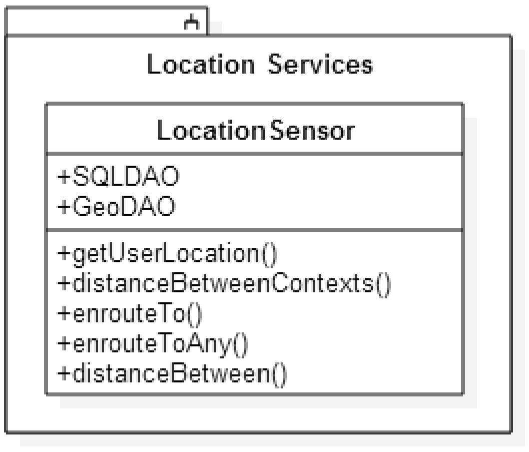


FIGURE 4.8: Location Services with virtual location sensor

for authentication. Figure C.3 in appendix C shows an example of entering an activity into a users calendar. This data provides virtual sensor data to the activity sensor.

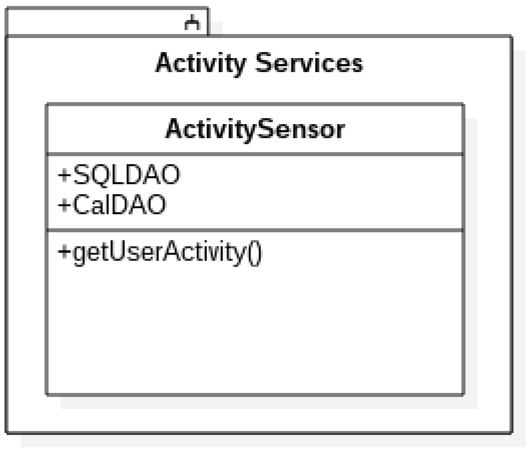


FIGURE 4.9: Activity Services with virtual activity sensor

4.3.3 Context Modelling

The model used by COAcHMAN is an object based model where context is represented as an object. The Context class has the following attributes

- **User:** The User this context is associated with
- **Location:** The Location of this context. Location is defined as Longitude, Latitude, Distance from home, Address of location as a String

- **Activity:** The Activity of this context. Activity is defined as start time, end time, summary of the activity and description of the activity
- **Date:** The time of the context. This is the time the context is created

The User class has the following attributes

- **Name:** The users name
- **Email address:** This is assumed to be unique
- **ContextType:** This is the current context flattened (context flattening is discussed in section 4.3.4).
- **Context:** The users current context
- **RecentContexts:** A LinkedList of the users previous 16 contexts. This is used as a FIFO queue so once the queue is storing 16 context objects each new context update forces the last one out of the queue.

On system startup COAcHMAN reads the user details and system location from the openHAB configuration file. A sample snippet of the configuration file is shown in figure C.4 in Appendix C.

4.3.4 Context Interpretation

Context Interpretation (or Context Reasoning) is the process of deducing high level context from the low level context data acquired in the lower layers of the architecture. In the following subsections we describe how we achieve this by combining two techniques; the context decision tree and application of the criteria pattern.

4.3.4.1 Context Decision Tree

The method employed by COAcHMAN for context inference is that of supervised learning where the system regularly gathers all context information and from this uses a context decision tree to generate the user context. A sample context tree is illustrated in figures 4.10 and 4.11 (the second tree is actually a follow on from the "Travelling" node in the first tree i.e. it is one single tree split into two for display purposes). These figures show a 4 level context decision tree which is traversed in a depth first fashion. The further down the tree we can traverse the more detailed the context is. However to make use of the decision tree we need to combine it with the Criteria Design Pattern.

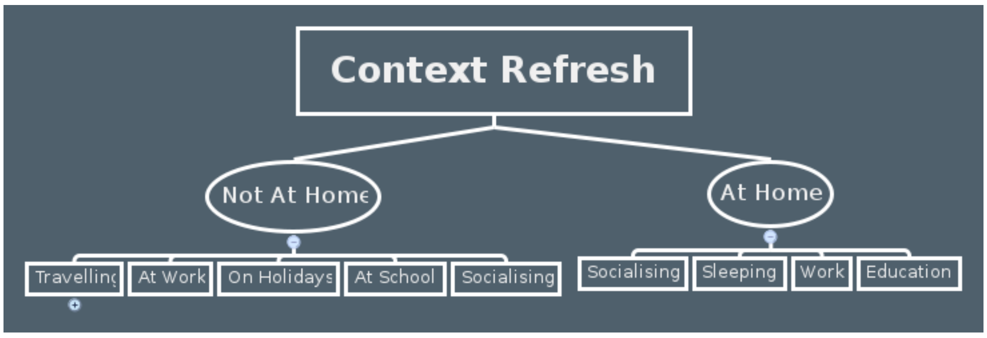


FIGURE 4.10: Context refresh decision tree

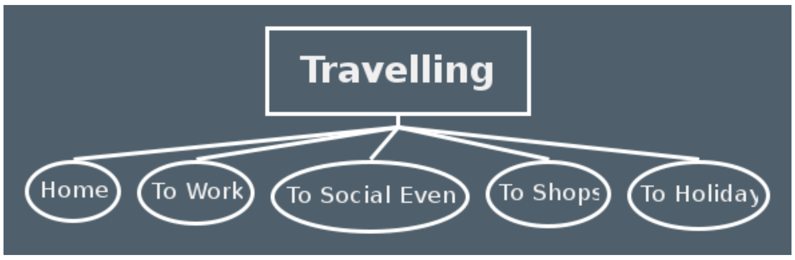
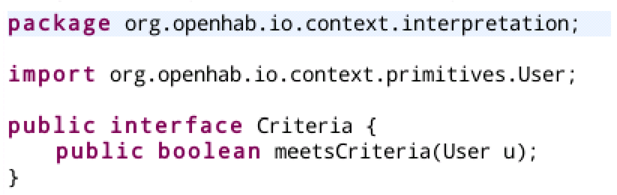


FIGURE 4.11: Level 3 Context refresh decision tree

4.3.4.2 Criteria Pattern

The Criteria Pattern (or filter pattern)[44] "is a design pattern that enables developers to filter a set of objects using different criteria and chaining them in a decoupled way through logical operations". To use this pattern we need an interface called *Criteria* (illustrated in figure 4.12). The usefulness of this pattern is that a context update can be "filtered" through the decision tree - at each node we evaluate if the new context meets the criteria of the context represented at that node. Each node therefore must implement the Criteria interface. All of the Context Criteria implementations are stored in a Context Criteria Library. The current set of criteria are illustrated in figure 4.13.

The core of the context interpretation layer is an algorithm which infers a simple "flattened" context. A "flattened" context is one of an enumeration of ContextTypes (illustrated in figure 4.14). the *getContext* algorithm uses the context decision tree and the context criteria pattern to determine which entry of the enumeration to assign to a User. When a User obtains a new ContextType it is transmitted as a status update event on the openHAB event bus. Any interested bundle subscribed to the event bus will be made aware of when a User has a new context.

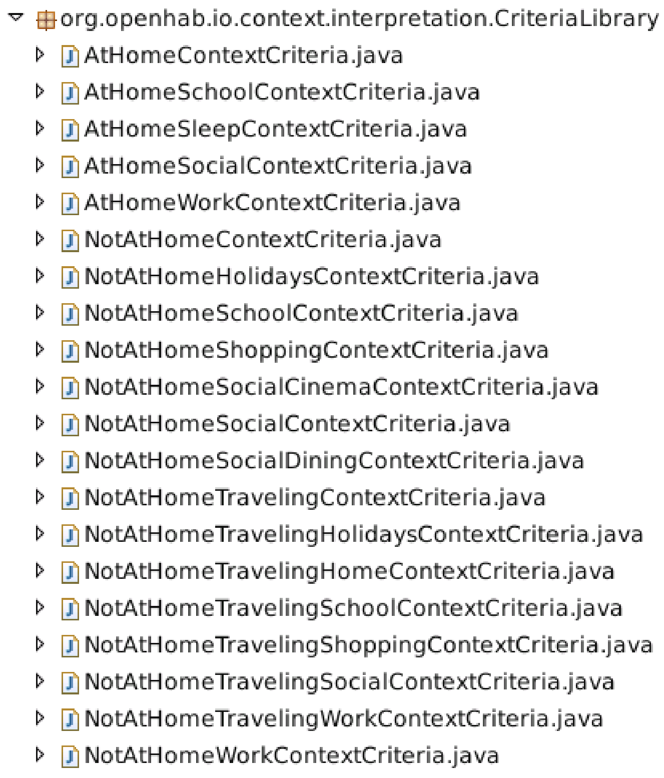


```java
package org.openhab.io.context.interpretation;

import org.openhab.io.context.primitives.User;

public interface Criteria {
    public boolean meetsCriteria(User u);
}
```

FIGURE 4.12: Criteria Interface



```
org.openhab.io.context.interpretation.CriteriaLibrary
    AtHomeContextCriteria.java
    AtHomeSchoolContextCriteria.java
    AtHomeSleepContextCriteria.java
    AtHomeSocialContextCriteria.java
    AtHomeWorkContextCriteria.java
    NotAtHomeContextCriteria.java
    NotAtHomeHolidaysContextCriteria.java
    NotAtHomeSchoolContextCriteria.java
    NotAtHomeShoppingContextCriteria.java
    NotAtHomeSocialCinemaContextCriteria.java
    NotAtHomeSocialContextCriteria.java
    NotAtHomeSocialDiningContextCriteria.java
    NotAtHomeTravelingContextCriteria.java
    NotAtHomeTravelingHolidaysContextCriteria.java
    NotAtHomeTravelingHomeContextCriteria.java
    NotAtHomeTravelingSchoolContextCriteria.java
    NotAtHomeTravelingShoppingContextCriteria.java
    NotAtHomeTravelingSocialContextCriteria.java
    NotAtHomeTravelingWorkContextCriteria.java
    NotAtHomeWorkContextCriteria.java
```

FIGURE 4.13: Criteria Library

```
  public enum ContextType implements PrimitiveType, State, Command {
    AT_HOME,
    AT_HOME_SOCIAL,
    AT_HOME_SLEEP,
    AT_HOME_WORK,
    AT_HOME_SCHOOL,
    NOT_AT_HOME,
    NOT_AT_HOME_TRAVELING,
    NOT_AT_HOME_TRAVELING_HOME,
    NOT_AT_HOME_TRAVELING_SOCIAL,
    NOT_AT_HOME_TRAVELING_WORK,
    NOT_AT_HOME_TRAVELING_SCHOOL,
    NOT_AT_HOME_TRAVELING_SHOPPING,
    NOT_AT_HOME_TRAVELING_HOLIDAYS,
    NOT_AT_HOME_SOCIAL,
    NOT_AT_HOME_SOCIAL_CINEMA,
    NOT_AT_HOME_SOCIAL_DINING,
    NOT_AT_HOME_WORK,
    NOT_AT_HOME_SCHOOL,
    NOT_AT_HOME_SHOPPING,
    NOT_AT_HOME_HOLIDAYS;
}
```

FIGURE 4.14: Context Types enumeration

4.3.4.3 getContext

The *getContext* algorithm is where the decision tree and criteria pattern are combined to map a users updated context to one of the context types which is transmitted on the event bus. The algorithm is illustrated in pseudocode form in figure 4.15

```
function getContext(ContextData) {
    ContextType := null;
    if(ContextData meets Criteria of AT_HOME)
        ContextType := AT_HOME;
        if(ContextData meets Criteria of AT_HOME_SOCIALISING)
            ContextType := AT_HOME_SOCIALISING;
        else if(ContextData meets Criteria of AT_HOME_SLEEPING)
            ContextType := AT_HOME_SLEEPING;
        else if(ContextData meets Criteria of AT_HOME_WORKING)
            ContextType := AT_HOME_WORKING;
        else if(ContextData meets Criteria of AT_HOME_SCHOOL) // e.g. Studying
            ContextType := AT_HOME_SCHOOL;
    else if(ContextData meets Criteria of NOT_AT_HOME)
        ContextType := NOT_AT_HOME
        if(ContextData meets Criteria of NOT_AT_HOME_SCHOOL)
            ContextType := NOT_AT_HOME_SCHOOL;
        else if(ContextData meets Criteria of NOT_AT_HOME_SOCIAL_CINEMA)
            ContextType := NOT_AT_HOME_SOCIAL_CINEMA;
        else if(ContextData meets Criteria of NOT_AT_HOME_SOCIAL_DINING)
            ContextType := NOT_AT_HOME_SOCIAL_DINING;
        else if(ContextData meets Criteria of NOT_AT_HOME_SOCIALISING)
            ContextType := NOT_AT_HOME_SOCIALISING;
        else if(ContextData meets Criteria of NOT_AT_HOME_SHOPPING)
            ContextType := NOT_AT_HOME_SHOPPING;
        else if(ContextData meets Criteria of NOT_AT_HOME_HOLIDAYS)
            ContextType := NOT_AT_HOME_HOLIDAYS;
        else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING)
            ContextType := NOT_AT_HOME_TRAVELING;
            if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_HOME)
                ContextType := NOT_AT_HOME_TRAVELING_HOME;
            else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_SOCIAL)
                ContextType := NOT_AT_HOME_TRAVELING_SOCIAL;
            else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_WORK)
                ContextType := NOT_AT_HOME_TRAVELING_WORK;
            else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_SCHOOL)
                ContextType := NOT_AT_HOME_TRAVELING_SCHOOL;
            else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_SHOPPING)
                ContextType := NOT_AT_HOME_TRAVELING_SHOPPING;
            else if(ContextData meets Criteria of NOT_AT_HOME_TRAVELING_HOLIDAYS)
                ContextType := NOT_AT_HOME_TRAVELING_HOLIDAYS;
    return ContextType;
}
```

FIGURE 4.15: getContext Algorithm

Each class which implements the Criteria interface contains the logic required to determine from the sensor data provided if it meets the criteria that object represents. It is worth showing an example of this algorithm determining a Context Type for one set of context data.

getContext Example Scenario: User is finished college for the day and is getting ready to go home. The Context data consists of the following information

- **User:** Bob
- **Date:** 17:50:00 09/07/2015
- **Location:** Latitude 51.883593, Longitude -8.531726, Address Rossa Ave, Bishopstown
- **Activity:** None in calendar

COAcHMAN is configured with the information in C.4.

1. ContextType is set to null
2. Does the context data meet the criteria for being at home? (table 4.1 shows the criteria for being at home). The context data will not meet this criteria.
3. Does the context data meet the criteria for not being at home (see table 4.2). The context data does meet this criteria.
4. Does the context data meet the criteria for not being at home and being at school (see table 4.3). The context does meet this criteria.
5. This is a leaf node in the decision tree. No further processing possible
6. Return ContextType NOT_AT_HOME_AT_SCHOOL

Condition #	**Condition**
1	Is the context location less than 40 meters from the COAcHMAN server? If yes then criteria is met

Table 4.1: AT_HOME Criteria

Condition #	**Condition**
1	Is the context location greater than or equal to 40 meters from the COAcHMAN server? If yes then criteria is met

Table 4.2: NOT_AT_HOME Criteria

Condition #	**Condition**
1	Is the context data activity set to SCHOOL? If yes then criteria is met.
2	Is the context location less than 100 meters away from a location the user has indicated as SCHOOL in the past? If yes then criteria is met

Table 4.3: NOT_AT_HOME_AT_SCHOOL Criteria

At this stage the Context Interpretation has achieved its goal. There are a number of further details not fully discussed in this section (such as refresh intervals, detecting and handling subtle changes in context, handling erroneous data from the virtual sensors) as they would be issues that any software developer would be able to detect and overcome without any in depth knowledge of the topic.

4.3.5 Context Utilisation

At this point the COAcHMAN module has successfully interpreted the low level context data and generated a high level context. This context update along with the user associated with it is sent as a status update event on the openHAB event bus.

At this point the responsibility for utilising the context data falls to the openHAB automation module. For the automation module to make use of the context data we do need to provide rules so the context aware rules, while not integrated into COAcHMAN, are still critical to the context aware home automation capabilities.

4.3.5.1 Rules

The openHAB rules are used for automating tasks. The automation module listens for all events on the event bus and if an event triggers a rule then that rule will invoke a script which will perform certain tasks (e.g. turn on the lights). In the next section we will discuss the syntax used to write rules.

4.3.5.2 XText and Xbase

XText is a set of tools and APIs provided by the Eclipse foundation for creating domain specific languages. The compiled byte code from these languages run on a JVM. A domain specific language is a *small programming language which focuses on a particular domain. The idea is that its concepts and notation is as close as possible to what you have in main when you think about a solution in that domain*[45]. Xbase is an expression language which is implemented using Xtext[46]. Xbase expressions provide *both control structures and program structures.* The notation and syntax of Xbase make it suitable for creation of rules in a form which is easily read and understood by users who may not be completely comfortable with programming.

4.3.5.3 Context Aware Rules

openHAB currently supports three different categories of rule triggers. These are

- Item based triggers. These are triggers based on reactions to events on the openHAB event bus.
- Time based triggers. These react at certain times
- System based triggers. These react on certain system changes.

The context aware rules are a form of Item based triggers where the events on the event bus are context updates. Each rule will intercept the event and will determine if it meets the condition of the rule. If it does then the script will execute. The rules are generally in the form illustrated in figure 4.16. The rules shown in figures 4.17, 4.18, and 4.19

```
rule "rule name"
when
    <TRIGGER CONDITION1> or
    <TRIGGER_CONDITION2> or
    <TRIGGER_CONDITION3>
    ...
then
    <EXECUTION_BLOCK>
end
```

FIGURE 4.16: Rule outline

show three possible context aware rules which could be used in a smart home setting.

Rule 1 shows the scenario where the home empties and as a result a number of sockets are turned off to save energy. Rule 2 shows the scenario where the home is occupied and the temperature rises above 20 °C and Rule 3 demonstrates the scenario where one of the occupants is travelling home and the home is cold, in that case the heating turns on to heat up the home.

```
rule "Rule 1 - turn off sockets when nobody at home"
when
        Item user1 received update or
        Item user2 received update
then
        if(user1.state != AT_HOME && user2.state != AT_HOME) {
            logInfo("rule1","Match for Rule1 - nobody at home, turn off all sockets")
            sendCommand(Kitchen_Socket,OFF)
            sendCommand(Locker_Socket,OFF)
            sendCommand(Bedroom_Socket,OFF)
        }
end
```

FIGURE 4.17: Sample Rule 1

```
rule "Rule 5 - turn on air conditioning if someone at home and temperature goes above 20
    degrees"
when
    Item Outside_Temperature received update
then
    var temp = Outside_Temperature.state as DecimalType
    if((user1.state == AT_HOME ||
        user2.state == AT_HOME) &&
        temp > 20) {
        logInfo("rule5","Match for Rule5 - very hot and someone at home, turn on aircon")
        sendCommand(AirConditioning, ON)
        sendCommand(AirConditioning, PLUS_10_DEGREES) // set aircon to 10 degrees celsius
    }
end
```

FIGURE 4.18: Sample Rule 2

```
rule "Rule 6 - turn on heating if someone is coming home and its cold"
when
    Item user1 received update or
    Item user2 received update
then
    var temp = Outside_Temperature.state as DecimalType
    if((user1.state == NOT_AT_HOME_TRAVELING_HOME ||
        user2.state == NOT_AT_HOME_TRAVELING_HOME) &&
        temp < 10) {
        logInfo("rule6","Match for Rule6 - turn on heating when someone coming home and
    it is cold")
        sendCommand(HeatingSystem,ON)
        sendCommand(HeatingSystem,PLUS_20_DEGREES)
    }
end
```

Figure 4.19: Sample Rule 3

4.4 Deployment of COAcHMAN

To demonstrate the work of this project I have created a mobile prototype smart home which is easy to setup and demonstrate. The smart home controller is that of a Raspberry Pi combined with a Z-Wave USB controller and a number of Z-Wave sockets. The sockets can be used to illustrate various rules in action. Figure C.5 in Appendix C illustrates the top-bottom deployment on a Raspberry Pi.

The source code for COAcHMAN can be found at https://github.com/aidanom1/openhab/tree/master/bundles/io/org.openhab.io.coachman

4.5 Requirements Validation

A final stage remains to ensure that COAcHMAN can fulfil the usage scenarios presented in section 4.1.2 and that is the requirements validation phase. To achieve this we will examine each requirement and how COAcHMAN meets that requirement.

Functional Requirements.

1. The system MUST support location tracking of users.

  - Achieved through HABDroid modifications

2. The system MUST be able to deduce a users context
    - Achieved through Context Interpretation Layer
3. The system MUST support activity tracking of users
    - Achieved though Google Calendar integration
4. The system MUST be able to react to changes in users context
    - Achieved through Context Utilisation Layer
5. The system MUST NOT allow a user to be without a context
    - Achieved though *getContext* algorithm
6. The system MUST allow users to specify how the system behaves when presented with updated context information
    - Achieved through rules editing
7. The system MUST be able to handle conflicting context information
    - Achieved through Criteria Pattern - each criteria is responsible for ensuring a clear Boolean value is assigned to context data

Nonfunctional Requirements

1. The system MUST support an arbitrary number of users
    - Achieved through configuration file
2. The system SHOULD require credentials for user access
    - Not completely achieved however this is possible via HABDroid client, it supports user authentication
3. The system SHOULD store context information for a period of time to facilitate context learning
    - Achieved through SQLDAO
4. The system SHOULD function with minimal user interaction
    - Once rules are initially setup the user has no requirement to manually interact with COAcHMAN

4.6 Challenges

During design and implementation a number of challenges emerged.

- Accurate GPS tracking is a considerable issue. It is generally only accurate to approximately 20-30 meters which renders it useless when attempting to locate users inside the home.
- Depending on the Google APIs to provide information such as distance carries unnecessary risk. Initially COAcHMAN used Google Maps to provide distance information however if your GPS location puts you a few meters away in another road for example it would calculate your distance using the assumption you were using footpaths or roads to travel. In one instance standing at the back of the garden put the user in a completely different road and by Googles computation put the user 1 mile away. There is a commonly used algorithm which computes the distance between two coordinates and COAcHMAN uses this.
- There is some effort on determining what constitutes a context change. Does a ten meter change in location constitute a context change? If the user is at home then yes, if the user is driving from a to b then no.
- There is potential for a huge amount of data generation. It did not present itself as an issue on this project however in time it would be.

Chapter 5

Further Development

A number of features were investigated but not implemented due to time constraints. It is the authors opinion that any full context aware system would strongly benefit from these features. These features are discussed in this chapter.

5.1 Authentication and Generic Profiles

Security and authentication are obvious concerns for owners and inhabitants of Smart Home Environments (SHE). Consider the case of a visitor entering the SHE, what rules should apply to them? One possible solution that presents itself during COAcHMANs development was that of generic profiles. A generic profile is a profile which can be assigned to a new user without them being added as a permanent inhabitant of the smart home (although it may be useful for permanent users to have a generic profile also). The constitution of a generic profile is illustrated in figure 5.1. A generic profile will have one of the attributes from each of the three layers i.e. a user could be a Trusted-Expected-Child such as a friend of one of the children living in the home or a user could be an Untrusted-Unexpected-Adult such as a door to door salesperson.

The use case diagram shown in figure 5.2 demonstrates how generic profiles could be used to facilitate access to the smart home site. As per [47] time is also shown as an actor. Time is used in conjunction with the users generic profile to determine if access to the site or the home itself should be granted. An example might be that between certain hours that any untrusted and unexpected adults are not given access to the site.

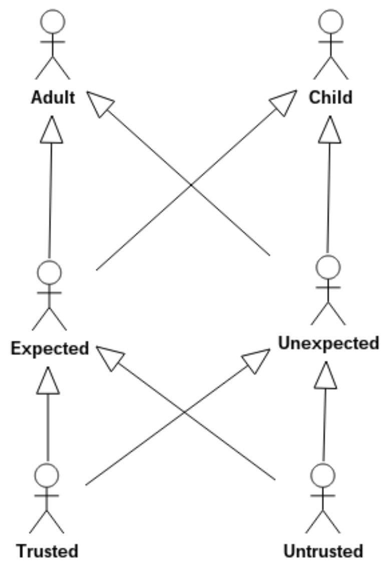


FIGURE 5.1: Generic Profiles

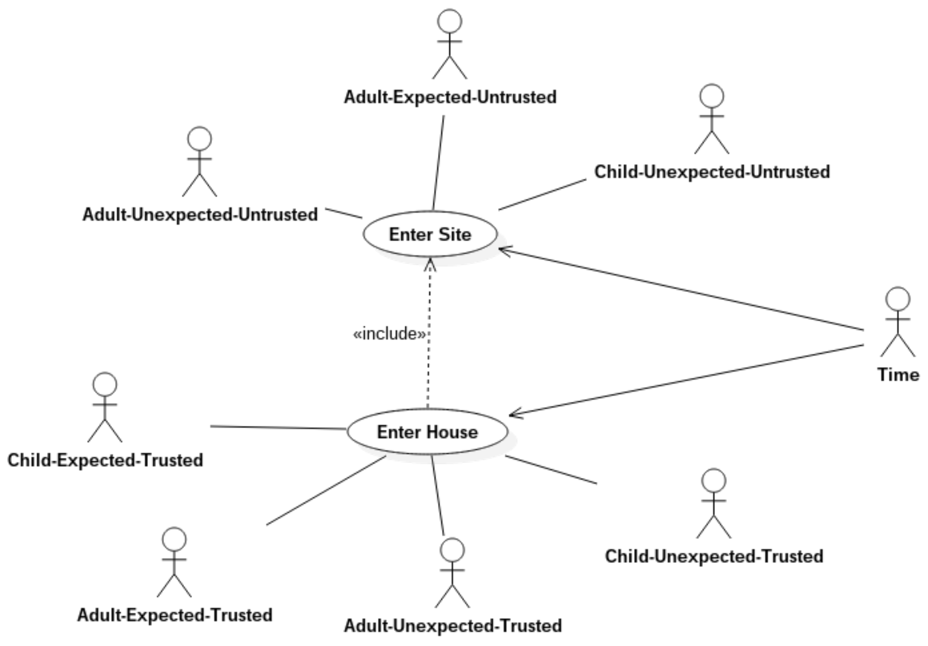


FIGURE 5.2: Generic Profile Site Access Example

5.2 Case Based Reasoning

Case Based Reasoning (CBR) is a problem solving technique that reuses previous cases and experiences to find a solution for current problems[48]. In essence CBR attempts to fill in information when information is lacking using data from previous situations.

Two research papers provide mechanisms for implementing CBR within a SHE. They are “Context-Reasoning for Smart Homes Using Case Based Reasoning” by Li, Liu and Zhou[49] and “Context-aware implementation based on CBR for smart home” by Ma et al[48]. They both produced methods of comparing contexts (sometimes referred to as *cases*) and computing a similarity metric.

Li et al defined a case as consisting of *time, location, physical-information* and the *objects* the user is using. Then *similarity(T,S)* where T is the base case and S is the new case is computed as $sim(T,S) = \frac{Common}{Common+Different}$ where $Common$ is the number of similar attributes between two cases and $Different$ is the number of different attributes between two cases.

Ma et al have a similar method. They define a case as

$case = (caseID, personID, habitID, environmentID, activeID, time)$

where these IDs are primary keys into a table of a database and each table stores data on that topic (e.g. the environment table stores the values of various sensors in the environment). They then run an algorithm which computes the *distance* between two cases (it shares some similarity to the work of Li et al).

Certainly it is the opinion of this author that CBR would be a valuable addition to COAcHMAN if further iterations were considered.

5.3 Internal Sensor Data

One of the main limitations of GPS location tracking is the poor accuracy especially indoors. This has a direct impact of the COAcHMAN system as the virtual sensor supplying location data cannot currently provide sufficiently accurate information. There are two solutions to this problem. Firstly, augment the location virtual sensor to acquire data from motion tracking sensors within the home (possible combined with some technology such as RFID to identify the users) or, secondly, wait until centimeter accurate GPS[50] is more widely available.

Chapter 6

Conclusion

While global energy and aging population concerns exist there will be research into smart home technology. In this project we have seen the possibilities for a context aware home automation system and how such a system can be achieved however as with all projects of this size there are unanswered questions and issues remaining. We should also reflect on how our system can integrate into domestic life and how our system aligns itself with predicted directions in smart home technology. We will discuss these topics in this chapter.

6.1 Issues and questions

At the end of the project it is prudent to document the issues and questions which remain. These are not technical issues as they have already been discussed in the development chapter rather these issues are concerned with the functional aspects of COAcHMAN and are worthy of investigation at a future date. Some of these issues are

- How does COAcHMAN detect and react to context data which indicated emergency situations?
- How can the system automatically incorporate new users?
- What protections exist for users data? The system records every movement of a user so special attention should be taken to this aspect.
- How do we ensure access is not granted to an attacker who has stolen a users phone?

- Can the system facilitate black out periods where context data is not collected when a user chooses?

6.2 Usability evaluation

Ferre, Juristo, Windl, and Constantine in "Usability basics for software developers"[51] propose the following attributes to use to measure how usable a system is. They are *Learnability*, *Efficiency*, *User retention over time*, *Error rate*, and *Satisfaction.* A more thorough investigation should be carried out independently however the author is willing to make some remarks on these attributes.

- **Learnability:** The measure of this attribute depends very much on whether we include setup or not. If the setup (creating the rules) was done by an expert (perhaps an installer) then a user would be able to complete tasks using the system almost immediately. If the users are required to write the rules themselves then the learning curve is very steep. Therefore it is recommended that an expert in the system carry out installation.
- **Efficiency:** After the installation phase the user never has direct interaction with COAcHMAN, therefore it can be considered quite efficient using the definition that efficiency is measured by the tasks per unit time it takes the user to perform a task.
- **User retention over time:** Once a user accepts COAcHMAN as a permanent appliance in the home then retention would be very high.
- **Error rate:** The definition of error rate supplied by Ferre et al is not the error rate of the system rather it refers to the number of errors the user makes while performing a task. As the tasks are based on rules the user themselves are unlikely to cause errors however there are risks around conflicting rules and invalid/incorrect context data being used. These are more likely to cause errors on the users behalf.
- **Satisfaction:** On first encountering COAcHMAN the user satisfaction is expected to be high however there is a risk of very high dissatisfaction in the case of failures as the system is responsible for automating many tasks within the home. Failure of these tasks may have significant annoyance factor.

To address some of the usability issues it may be wise to make use of expert installers to minimise the risk of conflicting rules and early system failures.

6.3 Smart Home trends and predictions

Major technology companies have been active in the smart home space for a number of years. Intel launched their "Edison" board designed for IoT applications (when it was launched the CEO of Intel demonstrated the "Edison" running in a baby monitoring system called Nursery2.0). Apple have launched "HomeKit" which is a home automation framework (integrated with Siri). Google recently announced project "Brillo" and project "Weave". "Brillo" is described as *an operating system for the Internet of Things* and Weave will *allow connected devices to more easily talk to each other.* Google describe Weave *as a common language that Brillo devices can use to communicate to Android devices and the cloud.* So clearly there is significant interest in smart home technology.

There are currently a number of predictions as to how much the Smart Home market will be worth in the next few years. Forbes predicts $11 billion dollars by 2017, Juniper Research predict $71 billion globally by 2018, and Business Insider are predicting $500 billion by 2019. Regardless of which figure is correct, all analysis is predicting double digit growth in sales of smart home technologies for the foreseeable future.

One does not have to look globally to see the influence and importance of smart home technology. In recent months EMC and Vodafone have announced investment in Cork based research worth €2 million into the area of IoT technology. Combine the EU Horizon 2020 projects into smart homes mentioned in Chapter 1 with the looming problem of our ageing population and the benefits smart home technology offers then we can see how this is an area of research which is only going to grow in importance.

6.4 Final Words

The importance of smart environments and the Internet of Things is slowly being realised by the technology industry. In this project we have produced a context aware home automation system which is integrated into a well known smart home middleware and which takes advantage of popular cloud services to enable virtual sensors for context data acquisition. A number of useful patterns have been identified and applied. The goals proposed at the outset - to enable user smart home acceptance and to handle changes in user routines - have been achieved.

Our system, COAcHMAN, provides a transparent method for users to both enjoy their homes and for their homes to behave more efficiently. There are a number of avenues available for any future development should the interest exist however the system in its current stage is usable. In the near future, when context aware smart homes are common place, it is hoped that this research will have been of some benefit.

Bibliography

[1] Fei Fei. Why smart grid?, 2014. URL http://www.gl4b.org/2014/11/why-smart-grid.html.

[2] Lisa Harmer. The internet of things: What does it all mean?, 2014. URL http://www.control4.com/blog/2014/04/the-internet-of-things-what-does-it-all-mean-infographic.

[3] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454, First 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.042313.00197.

[4] R. Want, B.N. Schilit, and S. Jenson. Enabling the internet of things. *Computer*, 48(1):28–35, Jan 2015. ISSN 0018-9162. doi: 10.1109/MC.2015.12.

[5] Diane Cook and Sajal Das. *Smart Environments: Technology, Protocols and Applications (Wiley Series on Parallel and Distributed Computing)*, pages 2–3. Wiley-Interscience, 2004. ISBN 0471544485.

[6] Taewoo Nam and Theresa A. Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, dg.o '11, pages 282–291, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0762-8. doi: 10.1145/2037556.2037602. URL http://doi.acm.org.ezproxy.cit.ie:2048/10.1145/2037556.2037602.

[7] European Commision. Horizon 2020 projects light the way for smart cities across europe, 2015. URL http://cordis.europa.eu/news/rcn/122384_en.pdf.

[8] H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8 (1):18–28, January 2010. ISSN 1540-7977. doi: 10.1109/MPE.2009.934876.

[9] A.R. Al-Ali, A.H. El-Hag, R. Dhaouadi, and A. Zainaldain. Smart home gateway for smart grid. In *Innovations in Information Technology (IIT), 2011 International Conference on*, pages 90–93, April 2011. doi: 10.1109/INNOVATIONS.2011.5893876.

[10] Sang Hyun Park, So Hee Won, Jong Bong Lee, and Sung Woo Kim. Smart home &ndash; digitally engineered domestic life. *Personal Ubiquitous Comput.*, 7(3-4): 189–196, July 2003. ISSN 1617-4909. doi: 10.1007/s00779-003-0228-9. URL `http://dx.doi.org.ezproxy.cit.ie:2048/10.1007/s00779-003-0228-9`.

[11] A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: Challenges and opportunities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2115–2124, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979249. URL `http://doi.acm.org.ezproxy.cit.ie:2048/10.1145/1978942.1979249`.

[12] L.M. Camarinha-Matos, F. Ferrada, A.I. Oliveira, J. Rosas, and J. Monteiro. Care services provision in ambient assisted living. {*IRBM*}, 35(6):286 – 298, 2014. ISSN 1959-0318. doi: http://dx.doi.org/10.1016/j.irbm.2014.08.001. URL `http://www.sciencedirect.com/science/article/pii/S1959031814000852`. Healthcom 2013.

[13] M Eichelberg, L Rölker-Denker, and A Helmer. Action aimed at promoting standards and interoperability in the field of aal-aal use cases and integration profiles. Technical report, Technical report, AAL Joint Programme, 2014.

[14] Hyunjeong Lee, Wan-Ki Park, and Il-Woo Lee. A home energy management system for energy-efficient smart homes. In *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*, volume 2, pages 142–145, March 2014. doi: 10.1109/CSCI.2014.109.

[15] David Lillis, Taghg O'Sullivan, Thomas Holz, Conor Muldoon, Michael O'Grady, and Gergory O'Hare. *Recent Advances in Ambient Intelligence and Context-Aware Computing*, chapter 10. IGI Global, 2014.

[16] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22(7):97–114, 2009.

[17] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2010.05.010. URL `http://www.sciencedirect.com/science/article/pii/S1389128610001568`.

[18] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5 (1):4–7, January 2001. ISSN 1617-4909. doi: 10.1007/s007790170019. URL http://dx.doi.org/10.1007/s007790170019.

[19] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, Sept 1994. ISSN 0890-8044. doi: 10.1109/65.313011.

[20] Jong yi Hong, Eui ho Suh, and Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509 – 8522, 2009. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2008.10.071. URL http://www.sciencedirect.com/science/article/pii/S0957417408007574.

[21] David Pierce. Location is your most critical data, and everyone's watching, 2015. URL http://www.wired.com/2015/04/location/.

[22] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society. ISBN 978-0-7695-3451-0. doi: 10.1109/WMCSA.1994.16. URL http://dx.doi.org/10.1109/WMCSA.1994.16.

[23] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66550-1. URL http://dl.acm.org/citation.cfm?id=647985.743843.

[24] Louise Barkhuus and Louise Barkhuus. Is context-aware computing taking control away from the user? three levels of interactivity examined. In *In Proceedings of Ubicomp 2003*, pages 149–156. Springer, 2003.

[25] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, December 1995. ISSN 1071-5819. doi: 10.1006/ijhc.1995.1081. URL http://dx.doi.org/10.1006/ijhc.1995.1081.

[26] X.H. Wang, Da Qing Zhang, Tao Gu, and H.K. Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22, March 2004. doi: 10.1109/PERCOMW.2004.1276898.

[27] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems. *Expert Syst. Appl.*, 36(4):8509–8522, May 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2008.10.071. URL http://dx.doi.org/10.1016/j.eswa.2008.10.071.

[28] Daqing Zhang, Tao Gu, and Xiaohang Wang. Enabling context-aware smart home with semantic technology. *International Journal of Human-friendly Welfare Robotic Systems*, pages 12–20, 2005.

[29] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, January 2005. ISSN 1084-8045. doi: 10.1016/j.jnca.2004.06.002. URL http://dx.doi.org/10.1016/j.jnca.2004.06.002.

[30] Abdur Forkan, Ibrahim Khalil, and Zahir Tari. Cocamaal: A cloud-oriented context-aware middleware in ambient assisted living. *Future Gener. Comput. Syst.*, 35:114–127, June 2014. ISSN 0167-739X. doi: 10.1016/j.future.2013.07.009. URL http://dx.doi.org/10.1016/j.future.2013.07.009.

[31] A. Hristova, A.M. Bernardos, and J.R. Casar. Context-aware services for ambient assisted living: A case-study. In *Applied Sciences on Biomedical and Communication Technologies, 2008. ISABEL '08. First International Symposium on*, pages 1–5, Oct 2008. doi: 10.1109/ISABEL.2008.4712593.

[32] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 434–441, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303126. URL http://doi.acm.org/10.1145/302979.303126.

[33] Richard Harper. Inside the smart home: Ideas, possibilities and methods. In *Inside the smart home*, pages 1–13. Springer, 2003.

[34] Kevin Fong. Can technology help defuse the dementia time bomb?, 2014. URL http://www.bbc.co.uk/guides/z3d99j6#zgj887h.

[35] Tom Zentek, CanOliver Yumusak, Christian Reichelt, and Asarnusch Rashid. Which aal middleware matches my requirements? an analysis of current middleware systems and a framework for decision-support. In Reiner Wichert and Helmut Klausing, editors, *Ambient Assisted Living*, Advanced Technologies and Societal Change, pages 111–125. Springer International Publishing, 2015. ISBN 978-3-319-11865-9. doi: 10.1007/978-3-319-11866-6_9. URL http://dx.doi.org/10.1007/978-3-319-11866-6_9.

[36] Lukas Smirek Gottfried Zimmermann and Daniel Ziegler. Towards universally usable smart homes – how can myui, urc and openhab contribute to an adaptive user interface platform? In *CENTRIC 2014, The Seventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, page 29 to 38, oct 2014.

[37] Andre LC Tavares and Marco Tulio Valente. A gentle introduction to osgi. *ACM SIGSOFT Software Engineering Notes*, 33(5):8, 2008.

[38] Raffel. openhab - empowering the smart home. history, concepts, examples, June 2014. URL http://wi.wu-wien.ac.at:8002/rgf/diplomarbeiten/Seminararbeiten/2014/201406-Raffel-OpenHAB.pdf.

[39] Roger Pressman. *Software Engineering: A Practitioner's Approach*, chapter 5, pages 121–122. McGraw-Hill, Inc., New York, NY, USA, 7 edition, 2010. ISBN 0073375977, 9780073375977.

[40] Roger Pressman. *Software Engineering: A Practitioner's Approach*, chapter 5, pages 133–138. McGraw-Hill, Inc., New York, NY, USA, 7 edition, 2010. ISBN 0073375977, 9780073375977.

[41] Ian Sommerville. *Software Engineering (7th Edition)*, chapter 6, pages 119–120. Pearson Addison Wesley, 2004. ISBN 0321210263.

[42] Paul Lyons, H Joe Steinhauer, Stephen Marsland, Jens Dietrich, Hans W Guesgen, and An C Tran. Exploring the responsibilities of single-inhabitant smart homes with use cases. 2010.

[43] Sun Microsystems. Core j2ee patterns - data access object, 2002. URL http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

[44] Tutorials Point. Design patterns - filter pattern, 2015. URL http://www.tutorialspoint.com/design_pattern/filter_pattern.htm.

[45] Eclipse Foundation. What is xtext?, 2015. URL http://www.eclipse.org/Xtext/documentation/.

[46] Sven Efftinge, Moritz Eysholdt, Jan Köhnlein, Sebastian Zarnekow, Robert von Massow, Wilhelm Hasselbring, and Michael Hanus. Xbase: implementing domain-specific languages for java. In *ACM SIGPLAN Notices*, volume 48, pages 112–121. ACM, 2012.

[47] Scott Ambler. Uml 2 use case diagramming guidelines, 2015. URL http://www.agilemodeling.com/style/useCaseDiagram.htm.

[48] Tinghuai Ma, Yong-Deak Kim, Qiang Ma, Meili Tang, and Weican Zhou. Context-aware implementation based on cbr for smart home. In *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, volume 4, pages 112–115 Vol. 4, Aug 2005. doi: 10.1109/WIMOB.2005.1512957.

[49] Po-Sheng Li, A. Liu, and Pei-Chuan Zhou. Context reasoning for smart homes using case-based reasoning. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, pages 1–2, June 2014. doi: 10.1109/ISCE.2014.6884414.

[50] Kenneth M Pesyna Jr, Robert W Heath Jr, and Todd E Humphreys. Centimeter positioning with a smartphone-quality gnss antenna. In *Proceedings of the ION GNSS+ Meeting*, 2014.

[51] X. Ferre, N. Juristo, H. Windl, and L. Constantine. Usability basics for software developers. *Software, IEEE*, 18(1):22–29, Jan 2001. ISSN 0740-7459. doi: 10.1109/52.903160.

Appendix A

Smart Environments, IoT, Home Automation

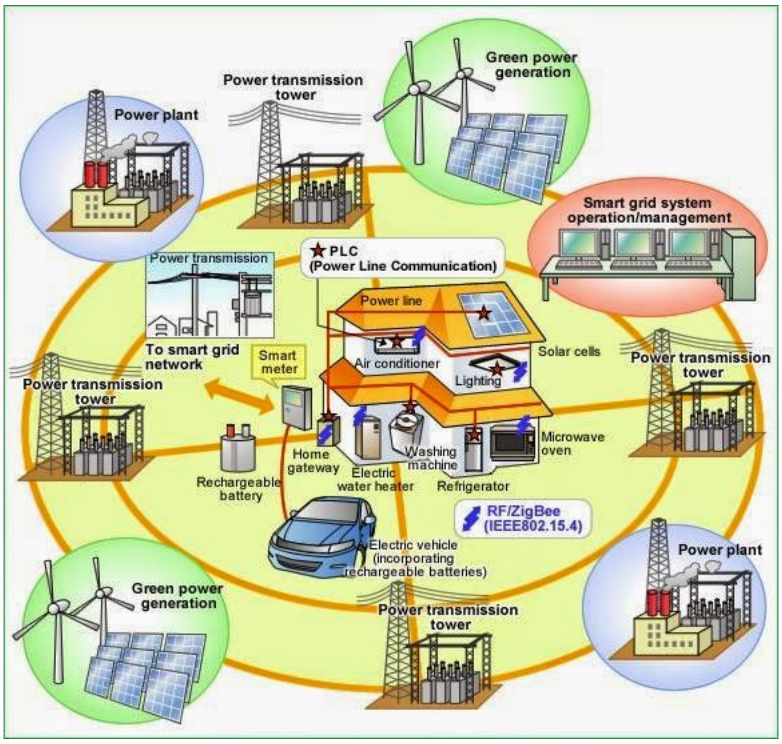


Figure A.1: Relationships between Smart Cities, the Smart Grid, and Smart Homes[1]

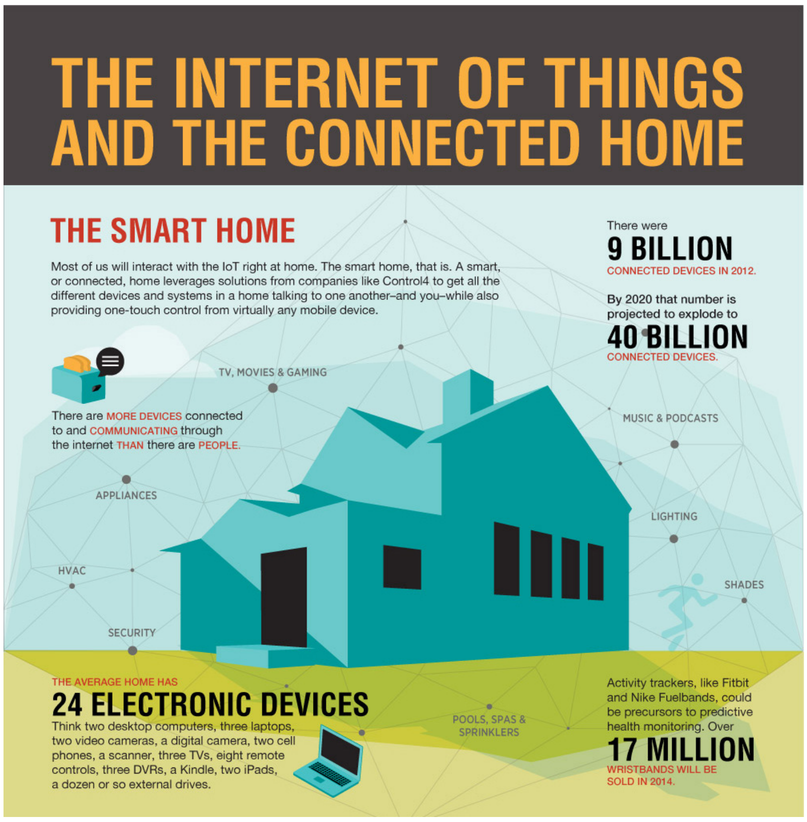
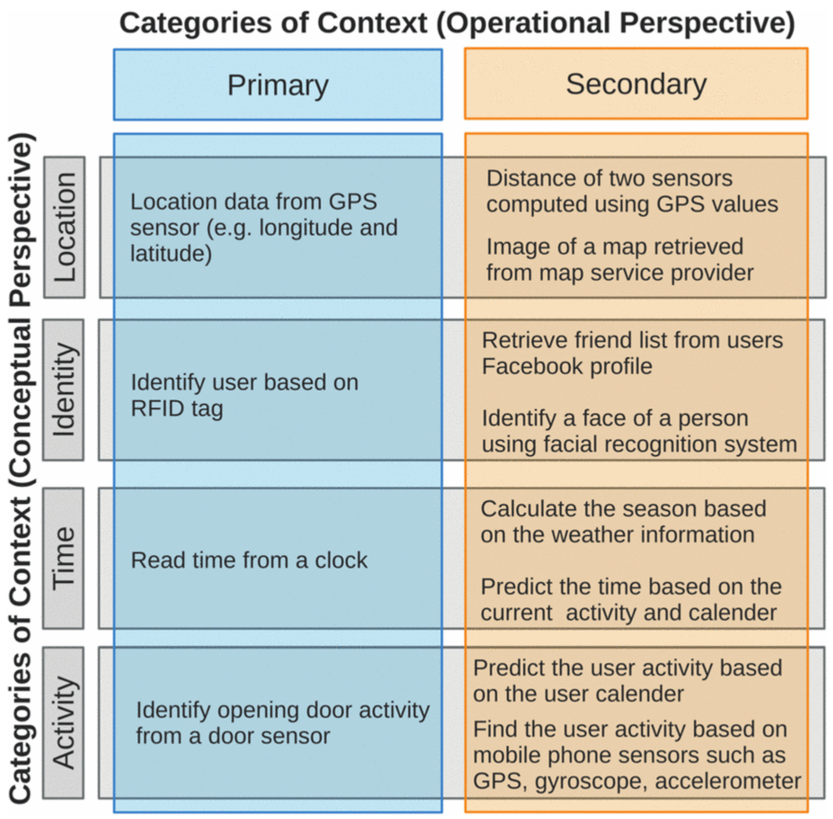

THE INTERNET OF THINGS AND THE CONNECTED HOME
THE SMART HOME
Most of us will interact with the IoT right at home. The smart home, that is. A smart, or connected, home leverages solutions from companies like Control4 to get all the different devices and systems in a home talking to one another–and you–while also providing one-touch control from virtually any mobile device.
There were 9 BILLION CONNECTED DEVICES IN 2012.
By 2020 that number is projected to explode to 40 BILLION CONNECTED DEVICES.
TV, MOVIES & GAMING
There are MORE DEVICES connected to and COMMUNICATING through the internet THAN there are PEOPLE.
MUSIC & PODCASTS
APPLIANCES
LIGHTING
HVAC
SHADES
SECURITY
THE AVERAGE HOME HAS 24 ELECTRONIC DEVICES
Think two desktop computers, three laptops, two video cameras, a digital camera, two cell phones, a scanner, three TVs, eight remote controls, three DVRs, a Kindle, two iPads, a dozen or so external drives.
POOLS, SPAS & SPRINKLERS
Activity trackers, like Fitbit and Nike Fuelbands, could be precursors to predictive health monitoring. Over 17 MILLION WRISTBANDS WILL BE SOLD IN 2014.

Figure A.2: The Internet of Things and home connectivity[2]

Appendix B

Background Research



Categories of Context (Operational Perspective)

Categories of Context (Conceptual Perspective)	Primary	Secondary
Location	Location data from GPS sensor (e.g. longitude and latitude)	Distance of two sensors computed using GPS values Image of a map retrieved from map service provider
Identity	Identify user based on RFID tag	Retrieve friend list from users Facebook profile Identify a face of a person using facial recognition system
Time	Read time from a clock	Calculate the season based on the weather information Predict the time based on the current activity and calender
Activity	Identify opening door activity from a door sensor	Predict the user activity based on the user calender Find the user activity based on mobile phone sensors such as GPS, gyroscope, accelerometer

FIGURE B.1: Categories of Context[3]

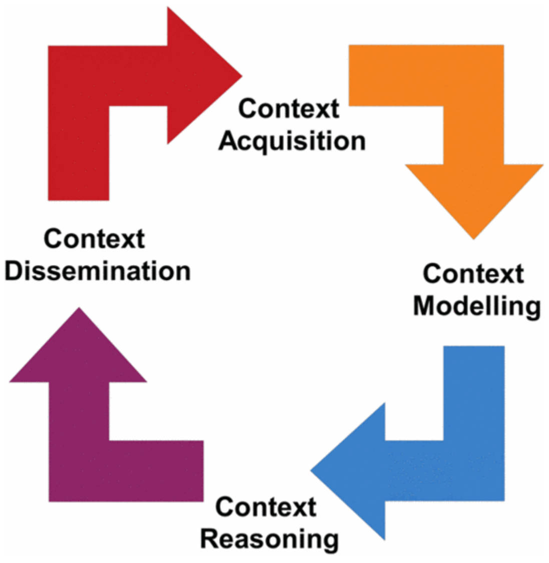
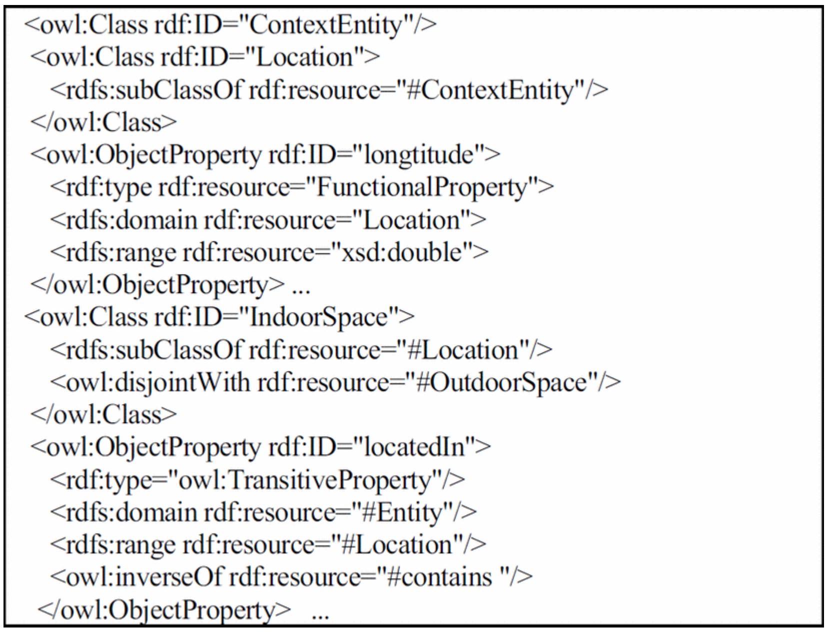


FIGURE B.2: Context Lifecycle[3]



```
<owl:Class rdf:ID="ContextEntity"/>
 <owl:Class rdf:ID="Location">
   <rdfs:subClassOf rdf:resource="#ContextEntity"/>
 </owl:Class>
 <owl:ObjectProperty rdf:ID="longtitude">
   <rdf:type rdf:resource="FunctionalProperty">
   <rdfs:domain rdf:resource="Location">
   <rdfs:range rdf:resource="xsd:double">
 </owl:ObjectProperty> ...
<owl:Class rdf:ID="IndoorSpace">
   <rdfs:subClassOf rdf:resource="#Location"/>
   <owl:disjointWith rdf:resource="#OutdoorSpace"/>
 </owl:Class>
 <owl:ObjectProperty rdf:ID="locatedIn">
   <rdf:type="owl:TransitiveProperty"/>
   <rdfs:domain rdf:resource="#Entity"/>
   <rdfs:range rdf:resource="#Location"/>
   <owl:inverseOf rdf:resource="#contains "/>
  </owl:ObjectProperty>   ...
```

FIGURE B.3: Partial OWL Serialization[3]

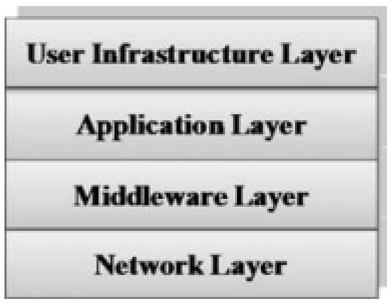


FIGURE B.4: Abstract Layer Architecture

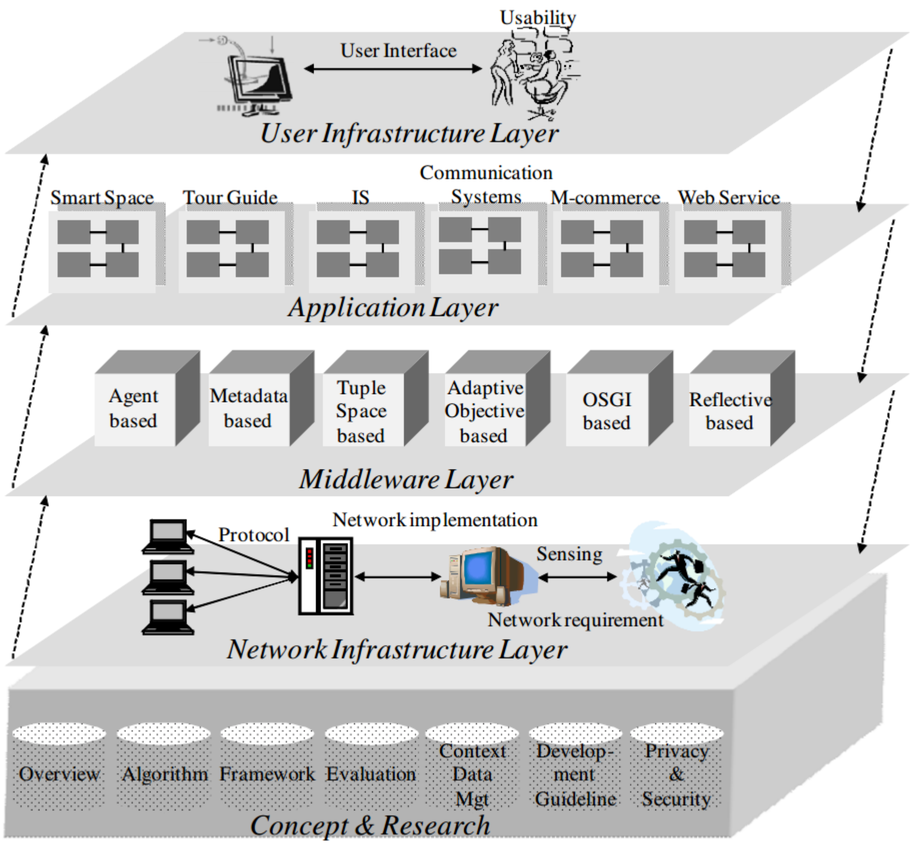


FIGURE B.5: Classification framework of context-aware systems

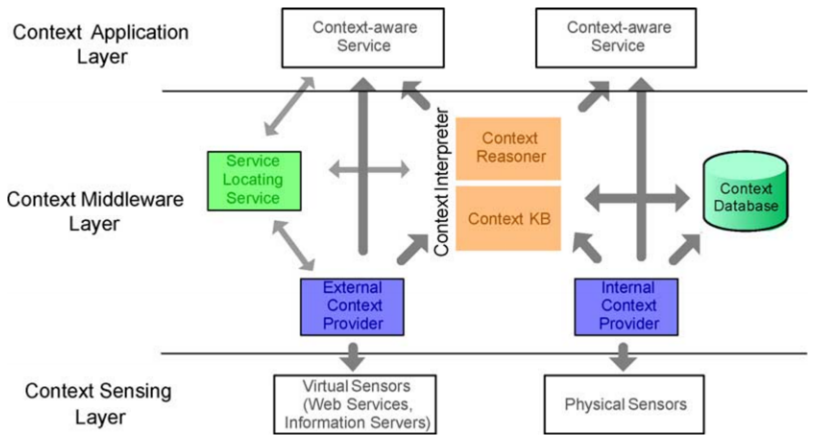


FIGURE B.6: SOCAM Architecture

Service type	**Rule**
Energy Saving	(?user status:sleeping) ∧ (?room light status:on) → ACTION:turn off light
Medication reminder	(?local time:17:00) ∧ (?medicine Y time:true) → ACTION:pop up reminder on phone
Fall detection	(?user status: lying down) ∧ (?fall detector status:on) → ACTION: alert emergency assistance

TABLE B.1: Example CoCaMAAL rules

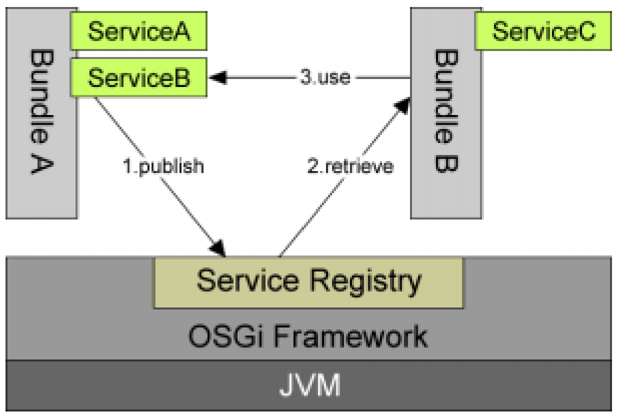


FIGURE B.7: OSGi Service Registry

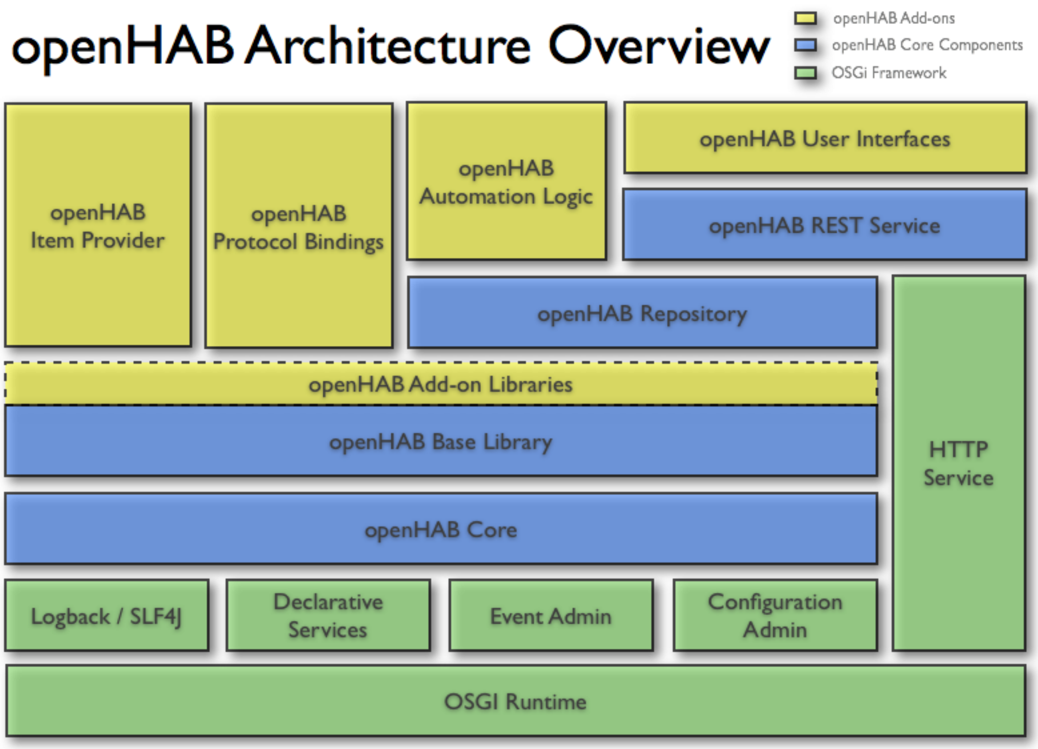


FIGURE B.8: openHAB Architecture

Appendix C

Implementation Appendix

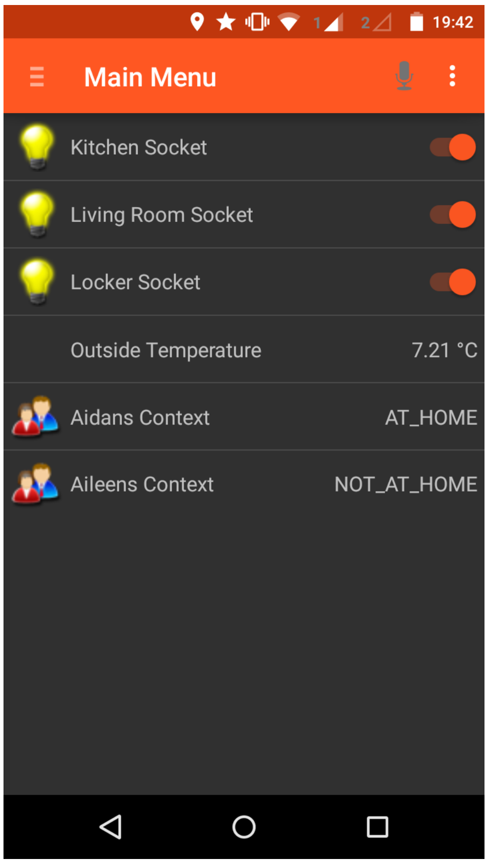
Use Case 1	
Primary Actors	COAcHMAN and User1
Actors Goals	User1 enters home
Preconditions	Home is empty
Main Functions	COAcHMAN determines User1 context has changed, COAcHMAN updates context
System Impact	Context change triggers action. As home is now occupied some lights turn on
Feedback Required	Lights are on

Use Case 2	
Primary Actors	COAcHMAN and all users
Actors Goals	Last user in home exits home
Preconditions	Only one user in home
Main Functions	COAcHMAN determines user context has changed, COAcHMAN updates context
System Impact	Context change triggers action. As home is now empty change is to turn off all lights in home to save power
Feedback Required	None

Use Case 3	
Primary Actors	COAcHMAN and User1
Actors Goals	User1 is scheduled to work from home
Preconditions	None
Main Functions	COAcHMAN determines user context has changed, COAcHMAN updates context
System Impact	Context change triggers action. Light in study turns on
Feedback Required	None

Use Case 4	
Primary Actors	COAcHMAN and User1
Actors Goals	User1 is travelling from work to home
Preconditions	None
Main Functions	COAcHMAN determines user context has changed, COAcHMAN updates context
System Impact	Context change triggers action. Light in study turns on
Feedback Required	None

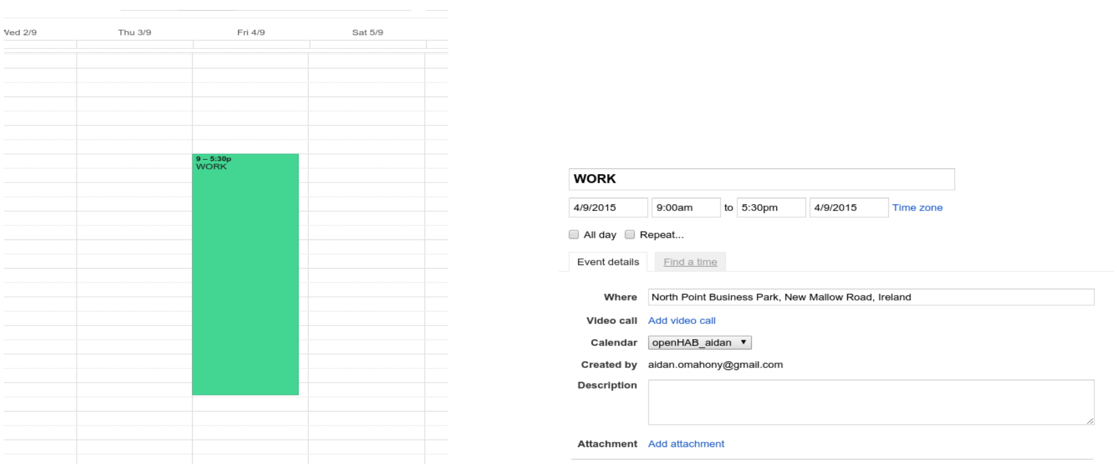
FIGURE C.1: Sample Use Cases


19:42

Main Menu

Kitchen Socket	
Living Room Socket	
Locker Socket	
Outside Temperature	7.21 °C
Aidans Context	AT_HOME
Aileens Context	NOT_AT_HOME

FIGURE C.2: HABDroid Screenshot



WORK

4/9/2015 9:00am to 5:30pm 4/9/2015 Time zone

All day Repeat...

Event details Find a time

Where North Point Business Park, New Mallow Road, Ireland

Video call Add video call

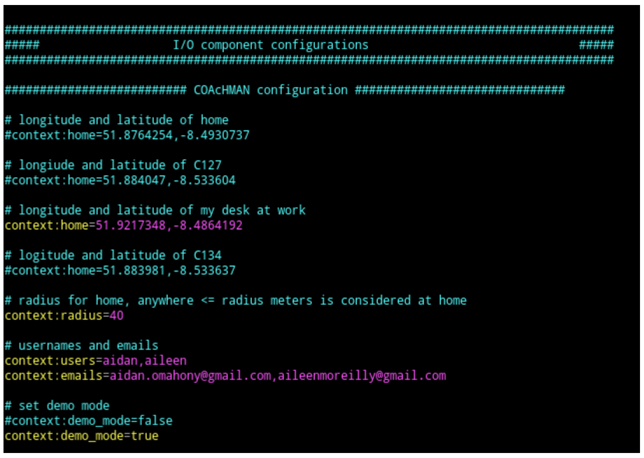
Calendar openHAB_aidan

Created by aidan.omahony@gmail.com

Description

Attachment Add attachment

FIGURE C.3: Google Calendar Example



```
#######################################################################################
#####                       I/O component configurations                         #####
#######################################################################################

########################## COAcHMAN configuration ##############################

# longitude and latitude of home
#context:home=51.8764254,-8.4930737

# longiude and latitude of C127
#context:home=51.884047,-8.533604

# longitude and latitude of my desk at work
context:home=51.9217348,-8.4864192

# logitude and latitude of C134
#context:home=51.883981,-8.533637

# radius for home, anywhere <= radius meters is considered at home
context:radius=40

# usernames and emails
context:users=aidan,aileen
context:emails=aidan.omahony@gmail.com,aileenmoreilly@gmail.com

# set demo mode
#context:demo_mode=false
context:demo_mode=true
```

FIGURE C.4: COAcHMAN configuration file

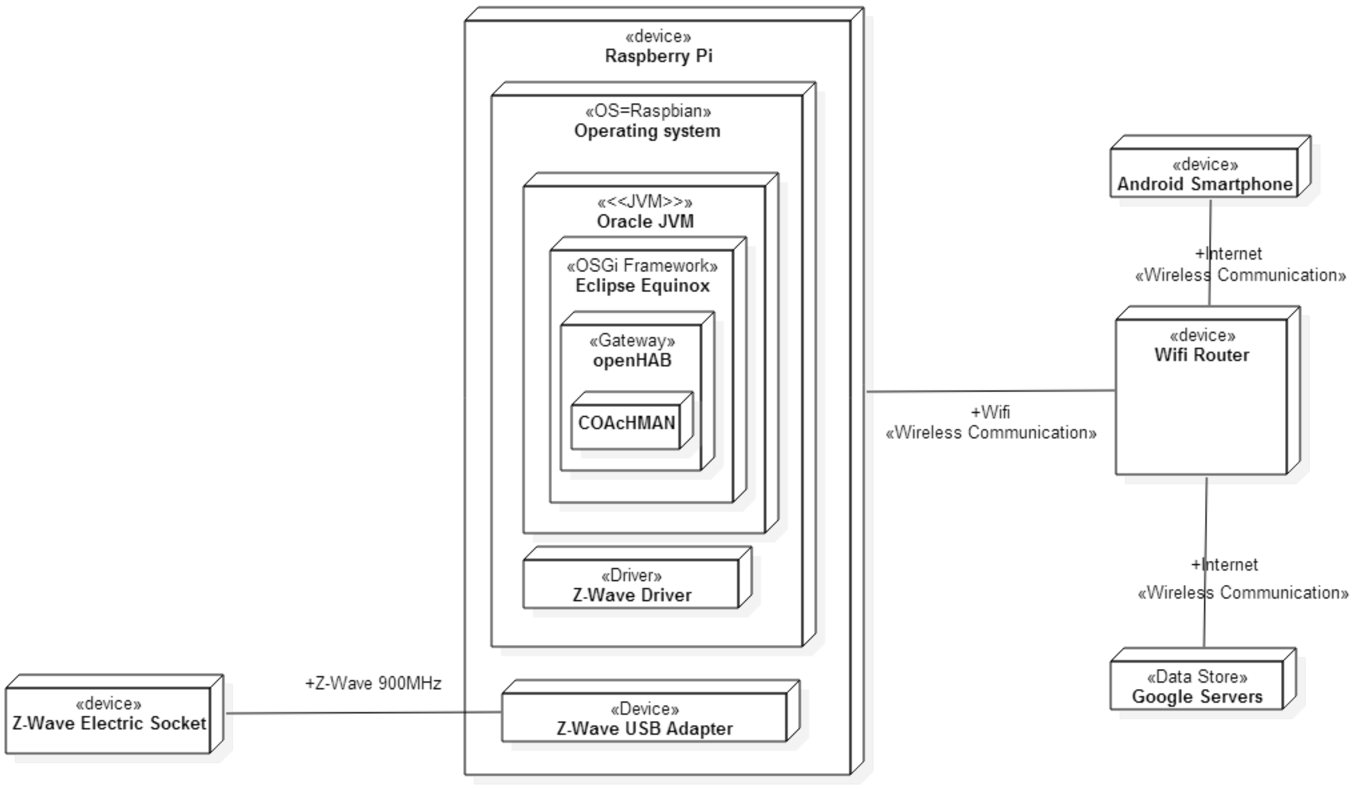


FIGURE C.5: COAcHMAN deployment on Raspberry Pi