Universidade Federal Fluminense

**João Elias Brasil Bentes Júnior**

# Handover in Ambient Assisted Living

NITERÓI

2015

Universidade Federal Fluminense

**João Elias Brasil Bentes Júnior**

# Handover in Ambient Assisted Living

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science. Topic Area: Artificial Intelligence.

Advisor:
Prof. D.Sc. José Viterbo Filho

NITERÓI

2015

**Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF**

B475 Bentes Júnior, João Elias Brasil
Handover in ambient assisted living / João Elias Brasil Bentes Júnior. – Niterói, RJ : [s.n.], 2015.
108 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2015.
Orientador: José Viterbo Filho.

1. Computação ubíqua. 2. Ambiente de vida assistida. 3. Handover. I. Título.

CDD 004.36

# Handover in Ambient Assisted Living

João Elias Brasil Bentes Júnior

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Master of Science. Topic Area: Artificial Intelligence.

Approved by:

[illegible]

Prof. D.Sc. José Viterbo Filho / IC-UFF (Advisor)

[illegible]

Prof. D.Sc. Orlando Gomes Loques Filho / IC-UFF

[illegible]

Prof. D.Sc. Karin Koogan Breitman / EMC Brazil

[illegible]

Prof. PhD. Anita Pinheiro Sant'Anna / Halmstad University

Niterói, March 12th, 2015.

To my family, friends and Bárbara.

# Acknowledgments

First and foremost this thesis would not have been possible without the help, patience, and counsels of my advisor, Prof. José Viterbo Filho. Thank you for believing in me and stimulating me to think outside the box.

A great thank to my fellow postgraduate students and friends João Gazolla, Jean Zahn, Luiz Guilherme Santos, Welton Barbosa, Douglas Marelli, Gleice Ramos, Adriano Antunes, Lidson Jacob, Carlos Heitor Moreira, Marco Aurélio Gonçalves, André Brandão, David Batista, Giancarlo Taveira, Christian Ruff and Diego Barboza for promoting a cooperative academic and social environment. Thank you for providing me wonderful moments.

Next up I would like to express my gratitude to the IS-lab team of the Halmstad University, in particular Anita Sant'Anna and Wagner de Morais, who directly contributed for me to find and narrow the subject of my thesis.

Thank you Prof. Esteban Clua and Prof. Marla Geller for the patience and opportunities they give me to grow as a researcher and as a person.

A special thought goes out to Cmte. José Colares and Cmte. Claudio Coreixas from the Brazilian Navy. Thank you for the leadership and life lessons.

I would like to acknowledge the academic and financial support of the Universidade Federal Fluminense and CAPES.

Last but not least, I would like to thank my parents João and Elvira Bentes for supporting my decisions and teaching me and my siblings — Kelly and Yuri — the importance of a good education. Without your help I never get this far.

And lastly, my deepest appreciation goes to my wife Bárbara Rocha. Thank you for your patience and your love during these two years and a half. Thank you for your moral support and for sharing the dreams that help me go forward.

# Resumo

Sistemas de Ambiente de Vida Assistida (AAL) empregam conceitos e técnicas de Computação Ubíqua e Ambientes Inteligentes visando estender o tempo que as pessoas idosas ou com deficiência podem viver de forma independente nos lugares onde preferem estar, tais como suas casas, escritórios, academias de ginástica. Os serviços oferecidos pelos sistemas AAL são chamados de Serviços AAL e os espaços aos quais estes sistemas estão incorporados são chamados de Espaços AAL. Normalmente, abordagens de AAL têm foco em oferecer serviços apenas na própria casa, uma vez que este é o lugar onde os idosos preferem ficar a maior parte do tempo. Desta forma, Serviços AAL, em geral, não são desenvolvidos para extrapolar os limites da casa, sendo geralmente providos apenas dentro dessas áreas bem delimitadas. No entanto, AAL visa também aumentar a qualidade de vida, a autonomia, a independência, a participação social, as habilidades e a empregabilidade das pessoas idosas ou com deficiência. Assim, alguns sistemas de AAL poderiam tirar proveito de um nível de continuidade de serviços. Quando usuários deixam suas casas, poderiam se beneficiar de alguns Serviços AAL, se estes também fossem disponibilizados para além das fronteiras domésticas. Ampliar a cobertura de serviços impõe desafios significativos para plataformas de Serviços AAL, em particular no caso da migração do usuário através de diferentes ambientes atendidos por diferentes plataformas. Em computação móvel, várias abordagens propõem métodos para permitir que um nó móvel migre através de áreas cobertas por redes diferentes, enquanto a conexão e a cobertura do serviço são mantidas. *Handover* é o termo geral empregado para descrever esta operação. Por analogia, formulamos o problema que abordamos nesse trabalho como "assegurar o *handover* de Serviços AAL", ou seja, permitir que usuários se movam livremente através de diferentes Espaços AAL, enquanto os Serviços AAL são providos de forma contínua. Com esta finalidade, propomos um arcabouço flexível para apoiar o *handover* de serviços no projeto de Serviços AAL. Assumimos que existem dois padrões de *handover*: *handover* direto, quando o usuário migra entre Espaços AAL cujas áreas de cobertura de serviço têm alguma interseção; e *handover* indireto, quando o usuário migra entre Espaços AAL sem interseção entre suas áreas de cobertura. Nossa proposta leva em consideração cada um desses padrões de *handover*. Além disso, no caso do *handover* indireto, propomos o uso de *smartphones* para agir como uma versão móvel de um Espaço AAL, garantindo a continuidade no provimento de serviços. Além disso, especificamos os componentes da arquitetura necessários para a implantação do suporte ao *handover* em plataformas de Serviços AAL. A partir da implementação desses componentes, descrevemos o nosso arcabouço para o desenvolvimento de Serviços AAL com suporte a *handover*. Finalmente, para demonstrar a viabilidade de nossa abordagem, projetamos, implementamos e avaliamos um protótipo baseado nos conceitos de nosso arcabouço flexível.

**Palavras-chave: Ambiente de Vida Assistida, Ambientes Inteligentes, Handover de Serviços**.

# Abstract

Ambient Assisted Living (AAL) systems comprise a set of ubiquitous technologies (AAL Services) embedded in a living space (AAL Space), which is aimed towards extending the time that elderly and disabled people can stay independently in whichever place they prefer, such as their homes, gym and office. Usually, AAL approaches focus on covering only the home environment, since their own house is the place where elderly people prefer to stay most of time. Hence, AAL Services for assisting and monitoring home dwellers are not commonly tailored to extrapolate the boundaries of the house, being generally provided only inside those well-delimited areas. However, AAL is not just about home-based health care, it also aims at increasing the quality of life, autonomy, independence, participation in social life, skills and employability of elderly and disabled people. Thus, some AAL systems might take advantage from a level of continuity. When users leave their houses, they would be benefited from some services commonly provided by AAL environments, if such services were also made available beyond the house's boundaries. Expanding the coverage of AAL imposes significant challenges to AAL platforms, particularly in the case of the migration of the user through different environments covered by AAL services. In mobile computing, several approaches propose methods to allow a mobile node to move efficiently and seamlessly through areas covered by diverse networks, while preserving connection and service coverage. Handover is the general term for this operation. By analogy, we formulate the problem we approach in this work as ensuring the handover of AAL services, in order to allow users to move freely through diverse AAL spaces, while providing and maintaining a continuous service. In order to accomplish this, we propose a flexible framework for supporting service handover in the design of AAL Services. We assume that there are two handover patterns: direct handover, when the user migrates between AAL Spaces whose service coverage areas have some intersection; and indirect handover, when the user migrates between AAL Spaces with no intersection between their coverage areas. Our proposed framework takes in consideration each of these different handover patterns. Furthermore, in the case of indirect handover, we propose the use of smart phones to act as a mobile version of an AAL Space. We also specify the architectural components necessary for deploying handover support in AAL Platforms. Based on that, we describe our framework to develop AAL Services with handover support, implementing the proposed architectural components. Finally, to show the feasibility of our approach, we designed, implemented and evaluated a prototype based on the concepts of our flexible framework.

**Keywords: Ambient Assisted Living, Ambient Intelligence, Service Handover**.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The world's population is ageing rapidly with an estimation of 1 in 5 people over 65 years old by 2030 compared to 1 in 10 nowadays [5]. In Brazil, there were around 14.9 million adults over 65 years old in 2013, which represents 7,4% of the total population [50]. This proportion is expected to grow up to 26,7% (about 58.4 million people) in 2060 [50]. Institutionalization of elderly people, which is a common proceeding in Brazil, is strongly related with the quality of the aging process of such individuals [63]. This practice usually focuses on the biological aspects of elderly care. Nevertheless, the process of "aging-well" should also contemplate social and psychological aspects such as interacting with family and friends, being in the preferred places, having autonomy to perform daily activities [61]. Elderly people would prefer to stay at home rather than move to care facilities because those are a more relaxed setting for them [16]. Providing care at home has several clear benefits for elderly people, particularly promoting social integration and autonomy, this has strongly motivated the development of mechanisms to improve quality of life at home.

Some studies in the Ambient Intelligence (AmI) field [3, 4] propose environments that are suitable for supporting wellbeing [49], senior assistance [23, 51, 11] and health support [36, 28] at home. An established approach in the literature consists of integrating data from different sensors in order to understand the behaviours of a resident and using actuators for changing the environment according to the person's demands [22]. Solutions focused on e-health and telemedicine have also been proposed [32, 57] such as the one proposed by *Tempo Laboratory*: *H-SAUDE* [24]. Smart home approaches have been considered as interesting solutions to provide health care at home for elderly people [47, 51]. According to [31], a smart home system may involve not only assistive and everyday technologies, but also medical technologies. Smart home systems are capable of delivering services to provide comfort, safety, independence and health care for the residents in the same environment, involving different types of users, not only the patient/resident, but also other actors such as relatives, caregivers and health care professionals.

Systems in the AmI domain that are tailored to provide health assistance at home for elderly people are called Ambient Assisted Living (AAL) systems. Such systems have specific technical requirements [74], which are focused on the peculiar needs of elderly people and related with the main goal of allowing them to live a life with the highest level of independence. In this context, AAL Systems can be defined as a set of ubiquitous technologies, named AAL Services, that are embedded in a living space, named AAL Space, which is aimed towards extending the time that elderly and disabled people can live independently in whichever place they prefer such as their homes, neighborhoods, offices and gyms [58].

AAL research initiatives involve multidisciplinary teams and focus on developing innovative ICT-based products, services and systems for allowing people to age well at home, in the community and at work [58]. However, AAL is not just about home-based health care, it also aims at increasing the quality of life, autonomy, independence, participation in social life, skills and employability of elderly people[75]. Last but not least, one of the objectives of AAL research is to reduce the need for social and health care.

Since 2008, Europe has invested around € 600,000 in research & development innovation ICT-based projects aiming at delivering solutions under the Ambient Assisted Living Joint Programme [1]. It demonstrates the concern with the ageing population and the consequent demographic change in Europe. In Brazil, the old age population has also been increasing in the last few years, as illustrated by the statistics previously presented. This scenario demonstrates the importance of developing such technologies also in Brazil.

Usually, Ambient Assisted Living approaches focus on covering only the home environment, since their own house is the place where elderly people most prefer to stay. Hence, AAL services for assisting and monitoring home dwellers are not commonly tailored to extrapolate the boundaries of the house, being generally provided only inside those well-delimited areas. However, some AAL systems might take advantage of a level of continuity. When users leave their houses, they could benefit from some services commonly provided by AAL environments, if such services were also made available beyond the house's boundaries. In other words, the migration of the users through different environments, i.e. the user's mobility, imposes a demand for the expansion of the boundaries of AAL coverage, which may be achieved by the integration of several different AAL Spaces

[1]AAL Joint Programme: http://www.aal-europe.eu/

or by the empowerment of outdoor services [19] with the support of mobile devices [74].

## 1.1 Problem Setting

Expanding the coverage of AAL imposes significant challenges to AAL Service Platforms, particularly regarding the continuous service provision for users moving through different AAL Spaces. In general, an AAL Service Platform must be able to transfer the provision of a given service to another AAL Service Platform, allowing the user to move seamlessly through AAL Spaces in the covered area, while preserving connection and service coverage. Handover is the general term for this operation. Although the handover problem was massively explored in Mobile Computing, handover in AAL is still an open problem. Most AAL Platforms described in the literature comprise mechanisms to provide AAL Services only in the well-delimited space of the house, imposing a limit for the user's mobility. In this context, we formulate the problem that we approach in this work as ensuring the handover of AAL Services, allowing users to move freely through diverse AAL Spaces while maintaining the continuous service provision.

## 1.2 Goals

The goal of this master thesis is to propose a flexible framework for supporting service handover in the design of AAL Services. We assume that there are two handover patterns: (a) direct handover and (b) indirect handover. In the former, the user migrates between AAL Spaces whose service coverage areas have some intersection. In the latter, the user migrates between AAL Spaces with no intersection between their coverage areas. Our proposed framework takes in consideration each of these different handover patterns. Furthermore, in the case of indirect handover, we propose the use of smart phones to act as a mobile version of an AAL Space. We also specified the architectural components necessary for deploying handover support in AAL Platforms. Based on that, we describe our framework to develop AAL Services with handover support, implementing the proposed architectural components. To show the feasibility of our approach, we designed, implemented and evaluated a prototype of a continuous health monitoring system with support for handover in three different environments: a house, a gym and an outdoor space.

## 1.3 Contributions

The main contributions of this master thesis are the following:

- A comparative study of five different approaches of AAL Platforms towards service handover.
- The formalization of handover protocols in terms of AAL Spaces and AAL Services.
- The definition of a functional architecture to support the handover of AAL Services by enabling the transfer of the service provision between AAL Platforms.
- The definition of a flexible framework extension to support the design of AAL systems with service handover, meeting user mobility requirements.
- The implementation and evaluation of a prototype for continuous health monitoring system as an AAL Service with service handover support.

In special, our proposed framework presents an original contribution for the Ambient Intelligence field, in general, and for the AAL domain, in particular, as it provides mechanisms to develop AAL Services considering the migration of users through different AAL Spaces, both indoors and outdoors.

## 1.4 Organization

This master thesis is composed by seven more chapters, organized as follows:

**Chapter 2: Fundamental Concepts.** In this chapter, we explain the fundamental concepts for the understanding of this work — Ambient Intelligence, Ambient Assisted Living and Service Coverage in Ambient Assisted Living —, presenting clear definitions about each of them.

**Chapter 3: Related Work.** In this chapter, we describe some research projects related to the topic of this master thesis — namely Persona, CASAS, Uranus and Greener Building —, explaining how they tackle the migration problem.

**Chapter 4: Handover Approaches.** In this chapter, we present an AAL scenario and discuss the patterns of handover in Ambient Assisted Living, relating both of them in order to clearly demonstrate the taxonomy.

**Chapter 5: Our Approach.** In this chapter, we present our approach for designing health monitoring system with service handover, explaining it by means of examples extracted from the reference scenario.

**Chapter 6: Implementation.** In this chapter, we describe the implementation of a continuous AAL Service and a middleware based on the concepts of our approach.

**Chapter 7: Evaluation.** In this chapter, we present an evaluation of our approach based on the formulation of competency questions.

**Chapter 8: Conclusion.** In this chapter, we discuss the contributions and limitations of the proposed approach, presenting also some topics for future work.

# Chapter 2

# Fundamental Concepts

In this chapter, we present the main concepts underlying this work. In Section 2.1, we describe our motivating scenario, to which we will refer throughout this work, in order to exemplify and clarify our approach. Next, we discuss the main concepts of the Ambient Intelligence 2.2. In Section 2.3, we describe the main concepts related to Ambient Assisted Living (AAL). Finally, in the last section we argue about the handover in intelligent environments, relating these concepts with the AAL domain.

## 2.1 Scenario : Maria's story

Maria is a not-quiet 74 years old retired teacher who lives alone in a smart home since her husband passed away, 5 years ago. She is keen to live independently and in her own house. Nevertheless, Maria's family is concerned about her safety, comfort and communication with them. Therefore, they have invested in equipping Maria's house with a special wiring infrastructure to enable Maria to control or program an array of automated home electronic devices and computer systems, which can also monitor several aspects of her everyday activities such as sleeping, having meals and taking shower.

Every day, assistive systems help Maria in daily affairs, such as preparing meals and managing medication. All rooms are equipped with sensors used by a fall detection system. In the bedroom, an adjustable bed enhances the comfort when she lies on it. Sensors installed in the bed measure Maria's movements and breathing, and send further data to a system that is able to identify whether she is sleeping or not. Besides monitoring her vital signs, the environment is also able to trigger an alarm if abnormalities are detected, such as tachycardia, bradycardia or even a heart attack. The sofa, chairs and armchairs also measure vital signs, in particular heart beats. Furthermore, Maria's doctor can access anytime and anywhere her physiological, behavioral and medication records. The home shares data about Maria's vital signs to the doctor's office (or his house). Because of that, the Doctor is allowed to continuously check Maria's health condition. Also, if an emergency situation is detected the smart home automatically calls her doctor, caregiver, relatives or emergency services.

Using such set of health monitoring systems, the doctor could access long-term data that helped him to identify disturbances in the normal rhythm of Maria's heart rate. Maria was invited to go to a hospital, where further exams were performed. After this new set of tests, she was diagnosed with chronic arrhythmia, which might evolve to a systolic heart failure. Since that, the doctor suggested Maria to use a long-term cardiac monitoring system for managing her chronic disease. Such equipment will allow him to be alerted whenever any abnormal heart functioning is detected. Because Maria's smart home already has mechanisms not only for measuring her heart rate, but also for analyzing the collected data, few set-ups were necessary to deploy such long-term health monitoring system. For instance, Maria would install a new application aimed at integrating data from mobile sensors and providing constantly reports about her health condition.

Long-term cardiac monitoring has to be available also when Maria is performing outdoors activities, for example, going to the supermarket, walking in the park, spending time with her friends or working out. For this purpose, before going out, she picks up her smart phone and wears a smart bra equipped with a heart rate monitor. At home, her heart rate is acquired by sensors placed on the furniture, which were already used by systems in the house. When Maria leaves the house, the bra starts to send her vital signals to the smart phone, which processes the signals and alerts the doctor whenever any issue is detected. Besides that, all the systems that depend on Maria's presence (e.g. falling detection) are disabled. When she comes back, the smart phone synchronizes all Maria's cardiac monitoring data with the smart home. Once the synchronization is performed, the system in the smart phone is turned off. When indoors, the smart home takes over Maria's heart rate monitoring and starts to provide again other services that depend on Maria's presence inside the house.

Three times a week Maria attends a Senior Gym for doing some regular aerobic exercises. Equipped with sensors, actuators and a smart environment system, such smart gym provides some services for their attendees such as vital signs monitoring. Because of that, when there, Maria does not have to be concerned about the continuous monitoring prescribed by the doctor. Despite the fact that Maria is still wearing the cardiac monitor, at the gym she can take advantage of an environment equipped with ubiquitous systems, differently from when she is outdoors. Also, her smart phone may save energy since the gym becomes responsible for processing data and for establishing communication with her smart home system to send the collected data.

## 2.2 Ambient Intelligence

Ambient Intelligence (AmI) is a paradigm that represents the future vision of intelligent computing, where environments provide computational support to people who inhabit them [4, 79]. AmI also comprises the mechanisms that intelligently orchestrate the pervasive infrastructure able to provide an intelligent environment, which sensibly supports and assists people.

AmI systems are intelligent systems [68] with particular characteristics: (i) context awareness, i.e., they must sense the current environment and autonomously perform appropriate adaptations; (ii) personalization, i.e., they must be able to recognize the user and tailor their behavior to his needs; (iii) anticipation, i.e., they must anticipate a person's desires and a person's environment as much as possible, without user interference; (iv) adaptive, i.e., their behavior can change in response to a person's actions and environment's context; (v) ubiquitous, i.e., they are embedded and integrated into our everyday environments; (vi)transparency, i.e., they recede into the background of our daily lives in an unobtrusive way. [4, 78]

For implementing inherently distributed systems, such as AmI systems, it is necessary to count on middleware infrastructures for abstracting over the intricacies of the underlying communication technologies, machine architectures and operating systems; hiding the distribution of the different parts that comprise the system; providing higher-level core functionalities, such as context-awareness, authentication, etc [37]. A middleware should provide interfaces for applications accessing the context of the environment or controlling actuators [64]. A middleware also coordinates software entities and heterogeneous networked devices contained in an intelligent environment [70].

In particular, a middleware should facilitate AmI applications by providing [64]: (i) uniform development support, i.e., common way to express the software's context awareness generically, not restricted to a specific language, operation system or environment; (ii) uniform and independent-platform, able to continuously acquire and analyze context data; (iii) interfaces for applications expressing their context data needs without knowing how the data was acquired ; (iv) context triggered actions, i.e., providing facilities for the applications performing actions in the environment (e.g. notifying the user, updating context data, controlling actuators etc) whenever the corresponding context values are observed; (iv) transparent support for ad hoc communication, i.e., to abstract the details of the ad hoc network communication from the applications to facilitate the interoperability between network types.

In general, flexible and light-weight operating system are necessary for connecting sensors and actuators and providing seamless connection from devices to users. Regarding context-aware computing, contextual information (e.g. location, time, temperature etc) must be handled by applications in some predefined formats. To accomplish this, data conversion and even data fusion from many different devices must be handled by the middleware. In Figure 2.1 we describe a reference AmI architecture [7] with a middleware layer that abstracts from the application the access to the sensors and actuators. The middleware also exports core services such as the discovery service, which proactively discovers new devices and software entities, establish new communication links, and notify the applications whenever it finds a compatible device. The Figure 2.1 also shows a high-level vision of AmI where the middleware acts as a layer that provides the context-sensitive interaction between the user and the environment.

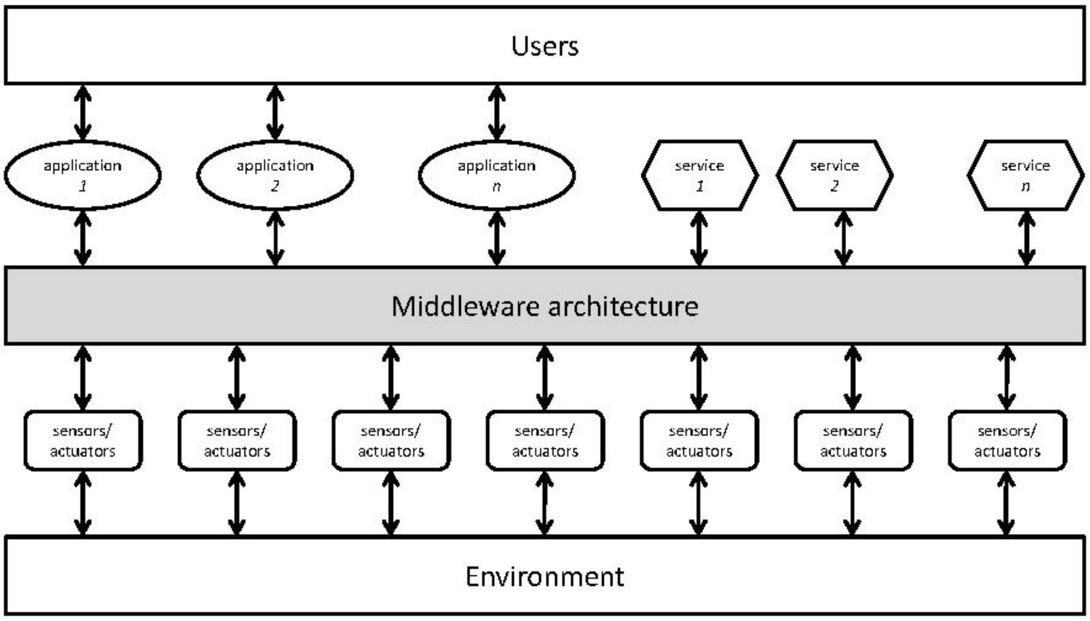


Figure 2.1: AmI reference architecture with middleware [7]

Research projects focused on building systems for supporting assistive technologies toward a user within the house represent an important type of AmI systems. Early works, such as *Gaia* [70], *Olympus* [13], *MavHome* [23], *EasyLiving* [15] and *HomeLab* [2], introduced this concept that has been improved over the years, reaching well-defined approaches such as *CASAS* [21], *PERSONA* [74], *AMIGO* [76] and *SM4ALL* [10]. Regarding other smart spaces, *NIST Smart Space* [72] and *Meeting Room* [72] represent the early projects of AmI systems for office environments that may be occupied by different users. In addition, recently systems such as *Greener Building* [30] already is able to handle different offices empowered by AmI systems in the same building. In Chapter 3, we discuss in further details the projects that are directly related with our work.

## 2.3 Ambient Assisted Living

In the last decade, there has been great research interest in topics related to AmI technologies. In particular, several works have proposed the creation of intelligent environments for enhancing comfort and safety of home dwellers, as well as for managing and conserving energy in those environments [44]. One of the main concerns of such works is offering comfort and safety for elderly people, that need to live independently without the constant presence of caregivers. Such technologies were employed for supporting health care assistance, such as health monitoring [57], daily activities assistance [46] and caregiver remote presence [48]. However, AmI systems focused on health assistance for older and disabled individuals at home have specific user and technical requirements [74], which can be summarized as follows:

- User requirements:
  - the system must not be designed as a full replacement of personal care by relatives, neighbors or friends;
  - the automatic decisions of the system must have the user's confirmation and remain traceable for him/her;
  - services should conform to usability standards and apply design-for-all with a focus on older adults;
  - user interfaces should incorporate features for coping with access impairments and capability changes due to aging;
  - when using different modalities (e.g. voice, text, graphics, signals, actions and state changes), the communication with the user should be done in a consistent way;

    - the user shall be able to prevent devices from gathering information about him/her at any time he/she wishes;
    - transmission of data to third parties must be kept to a minimum and security of data transmission to external entities must comply with legal requirements.

- Technical requirements:
    - guarantee a high level of flexibility in the distribution of features and facilitate the integration of arbitrary devices and applications into the system;
    - support ad-hoc networking and different communication patterns (e.g. call-based and event-based);
    - provide service discovery mechanisms and service chaining;
    - provide mechanisms for event aggregation and support parallel processing of events.

The requirements described above are based on the elderly needs in regard to a life with the highest level of independence, which represents one of the main goals of Ambient Assisted Living (AAL). In this context, AAL can be defined as a set of ubiquitous technologies embedded in a living space, named AAL Services, targeting the extension of the time that elderly and disable people live independently at their preferred place, i.e., AAL space, such as home, neighborhood, office and gym [58]. AAL initiatives, like *Ambient Assisted Living Joint Program*, involve multidisciplinary researchers and focus on the development of innovative ICT-based products, services and systems for ageing well at home, in the community or at work [58]. However, AAL is not just about home-based health care, it also aims at increasing the quality of life, autonomy, independence, participation in social life, skills and employability of elderly people[75]. Last but not least, one of the objectives of AAL research is to reduce costs with social and health care.

In this context, an AAL Service can be defined as a composition of one or more features that covers specific needs of elderly or disabled people, allowing them to live more independently [74, 26]. In general, the AAL Services can be classified in four categories: (i) AAL Services supporting social inclusion and experience exchange; (ii) AAL Services supporting elderly users in their daily life activities; (iii) AAL Services supporting elderly people to feel more confident, safe and secure,and helping their relatives to manage risky situations; (iv) AAL Services fostering mobility and supporting elderly people outside their homes. Indeed, the main difference between AmI systems and AAL Services is that the latter must address the requirements inherent in the elderly needs of independence.

AAL Services also can be classified according to the environment scope as indoor assistance or outdoor assistance. Systems for indoor living assistance work as well in a well-defined local scope: houses, apartments, offices, gyms and elderly care homes. They can be built upon a well-known hardware and software installation in the environment, then tending to provide a stable environment. Examples of such systems can be: cooking assistance, medication assistance, services for finding things and emergency prediction. Outdoor living assistance systems support people during activities outside their homes: while shopping, traveling, and during other social activities. These systems are faced with highly unstable environmental conditions such as availability of wireless communication, accessibility of network services and acquisition of context information. Instances of such systems can be: shopping assistance, banking assistance and orientation services. Also, challenging for outdoor systems are elders who cannot walk and/or coordinate their physical activities in order to participate in activities outside their homes, travel, and interact in social activities [66].

Approaches based on different software architectures for supporting systems in AAL domain have been proposed along the years, including SOA [81], MAS [6] and P2P [67]. PERSONA [74], SOPRANO [80], universAAL [45] and AMIGO [76]. Those architecture can be considered as AAL Service Platforms, which means a framework (specific or open) that must provide the necessary infrastructure for supporting AAL services, performing various commons functions in order to avoid duplicating efforts, and facilitate communication and collaboration between various components. In order to increase the interoperability, the framework can provide ontologies [77], communication protocols and data exchange formats. AAL Services, as AmI systems, are highly distributed and can apply AI techniques which highlight the importance of the interoperability framework [55].

## 2.4 Service Coverage and Handover

In this section, we discuss the concepts of service coverage related to Ambient Intelligence. We also present the migration process in wireless networks, including the general approach of the handover protocol and its fundamental challenges (Subsection 2.4.1). Finally, we discuss the idea of the handover protocol in the AAL domain (Subsection 2.4.2).

### 2.4.1 Service Coverage

Ambient Assisted Living approaches usually cover the home environment because the house is the place where elderly people prefer to stay longer. Hence, AAL services for assisting and monitoring home residents are not commonly tailored to extrapolating the boundaries of the house, being provisioned only in well-delimited areas. However, as we discussed in the motivating scenario (Section 2.1), some AAL systems may benefit from a level of continuity. When the user leaves the house, there are some services provided by AAL environments that should also be made available beyond the house boundaries. In other words, the user's migration demands the expansion of the AAL coverage, and this may be achieved by the integration of several different AAL environments or by the empowerment of outdoor services with the support of mobile devices.

Expanding the coverage of AAL imposes significant challenges to AmI middleware platforms, in particular in the transition of the user between different covered environments. In mobile computing, several approaches offer methods to allow a mobile node to efficiently and seamlessly move through network-covered areas such as [53, 9, 18]. The general term for this operation is handover. The term handover means an action of relinquishing property or yielding control of something [1]. In mobile computing, handover can be generally defined as a method for transitioning a mobile network node between two network cells [65]. The transition should be executed as seamlessly as possible to provide continuity of service without the user perceiving the network change. This allows users to move freely between diverse types of access technologies and change devices, if necessary, while maintaining application continuity.

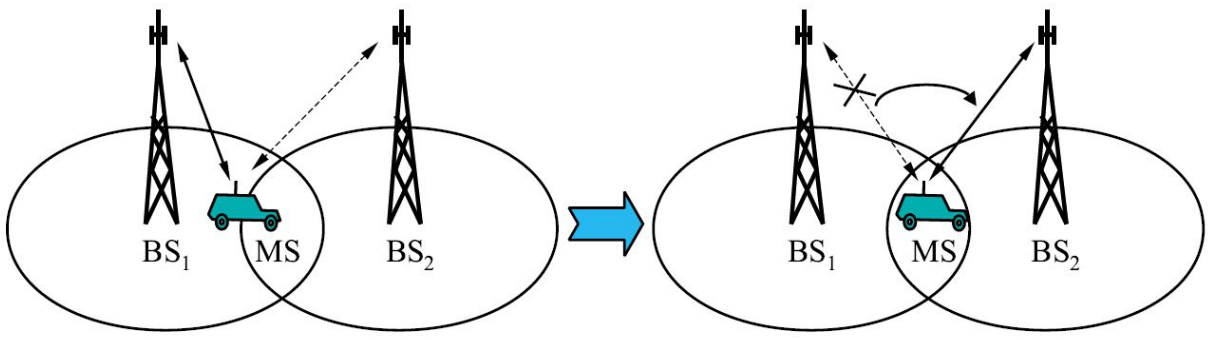


Figure 2.2: Basic handover process between two antennas [82]

In general, the process of the handover comprises two major phases: (i) detection, attribution and transference and (ii) update. The first phase involves migration detection (i.e., identifying the need to start a handover), the allocation and attribution of a new channel of communication and further transference of the mobile node from the old base station to the new base station. Network elements that handle mobile computer location information are notified and updated to guarantee that the traffic of packets can be correctly redirected to the new location. The Figure 2.2 illustrates the general concept of the handover. A mobile network node, i.e., mobile station (MS), moves from a network cell covered by Base Station 1 (BS1) to another network cell covered by Base Station 2 (BS2). The handover process starts when the MS enters the intersection between BS1 and BS2 coverage. Once the MS migrates to the BS2, the communication with BS1 is terminated [82].

The handover can be classified according to many factors, such as the network level at which mobility is handled, the distance between antennas, and the wireless network technology. For instance, based on the wireless network technology, the handover can be classified as a vertical handover or a horizontal handover. In the latter, the handover occurs between network cells of the same type (e.g., UMTS to UMTS, WLAN to WLAN). In the former, the handover is between two different types of cells (e.g., UMTS to WLAN), which may be different in size as well [82].

Although AAL Spaces comprise different network technologies (e.g., body sensor networks, WLAN, Bluetooth, etc.), we assume that AAL Spaces are network cells in a homogeneous network. In this work, we focus on continuously providing services through different AAL Spaces, thus enabling mobility for the user. The user transition between environments can be related to the transition of mobile nodes between network cells; however, the abstraction level of our problem does not address the network architecture or even the network handover protocols. Our concern is how to manage service continuity and user mobility in AAL Platforms, in particular by deploying middleware services for providing such features in the AAL Space. Therefore, our approach is inspired on the concept of horizontal handover discussed above, which we will designate as the handover.

### 2.4.2 Handover Protocol

The movement of a mobile node between two network cells can also be related with the concept of *migration*, which is a process that "occurs when a *migration element ME* moves between two domains *domains DomO* and *DomN* while it is interacting with one or several *correspondent elements CE*" [33]. Each domain has a *Domain Representative*, and *DomRep* is responsible for serving any *ME* currently located in the corresponding domain. Additionally, the *Domain Representatives* of *DomO* and *DomN* (called DomRepO and

DomRepN, respectively) must also execute some coordinated actions when an *ME* leaves *DomO* and enters *DomN* [65]. In mobile computing, the *ME* and the domains represent the mobile node and the network cells, respectively. Additionally, the representatives can be related as the bases stations, in particular the access points (e.g., antennas).

The coordinated actions described above composes the general steps of a *handover protocol*. In the Figure 2.3 we illustrate the *handover protocol* in the following way:

1. handover detection and beginning;
2. authenticate *ME's* identity and check its permission to be served by *DomRepN*;
3. register *ME* at *DomRepN* and de-register at *DomRepO*;
4. (pre-)allocate resources at *DomRepN* and de-allocate them at *DomRepO*;
5. update some network-resident state of *ME* at some nodes and/or transfer this state from *DomRepO* to *DomRepN*;
6. ensure that all properties of the service are guaranteed during and after the migration.

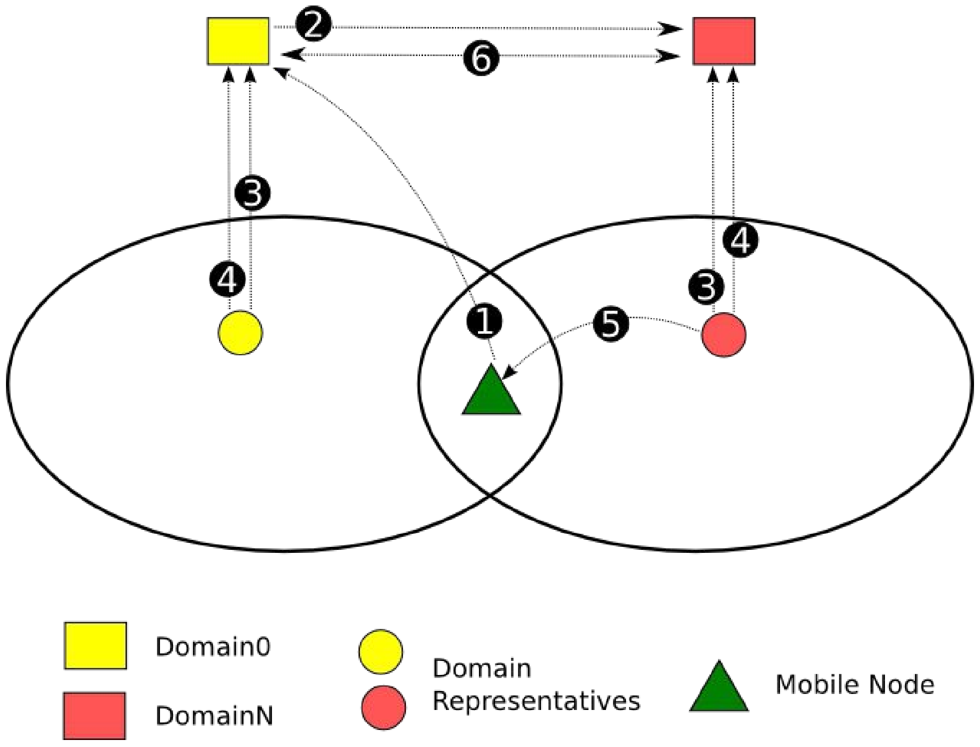


Figure 2.3: Major events of a handover protocol

It is worth mentioning that one of the most important handover challenges in wireless networks is to reduce the *handover latency* and the the amount of packet loss during the *handover protocol* [73]. According to the *handover protocol* discussed above, the mobile element cannot receive the packets on its new point of attachment (i.e., DomN) until the handover ends. The elapsed time from handover detection until the mobile element is fully served by the *DomN* is called *handover latency.*

During the handover, the mobile element may experience connectivity interruptions and be subject to extra security threats, while users would like to receive their services seamlessly. Certain handover approaches also address such issues (e.g., reconnect the node). The time spent to recover such issues also increases the handover latency. In the case of our domain, the level of tolerance with handover latency and packet loss is higher than in wireless networks because the frequency of migration between AmI tends to be lower than the migration between network cells. In addition, the handover steps are not necessarily executed in order. However, our research problem does not demand such execution flexibility. In this work, we will discuss the handover protocol considering the sequence presented above and in the Figure 2.3.

### 2.4.3 Handover in AAL

In the context of Ambient Assisted Living, the handover can be considered a mechanism that enables continuous service provision for a resident regardless of his location. The concepts regarding the handover process can also be related with the AmI domain. An *ME* can be related to a resident (mobile node), a *Domain* with the environment in which the resident is and in which the AmI service is covered, and the *DomReps* with the AAL Space or a mobile device (e.g., smart phones, tablets) that is responsible for provisioning AmI services. For instance, in our reference scenario (Section 2.1), the continuous health monitoring system remains with Maria even when she leaves the house. As the outdoors is an unknown environment, the smart phone acts as an AAL Platform that provides a small AAL Space around Maria. In this case, a migration between two AAL Spaces happens when she leaves the house, i.e., she moves from the house to the outdoors. During the movement between the environments, a handover protocol is executed to transfer the provisioning of the monitoring service from the AAL Platform for the house to the AAL Platform for the outdoors (i.e., the smart phone).

Despite the fact that the *handover protocol* should be followed for migrating a migratin element from one environment to another, some AAL Platforms provision migration for services without necessarily employing a handover protocol as presented above. Some approaches implement their own simplified protocols to address resident migration. In the next chapter, Related Work (Chapter 3), we present, discuss and compare some approaches aimed at supporting migration between two or more AAL Spaces.

# Chapter 3

# Related Work

In this chapter, we discuss works that present different approaches for expanding the service coverage of Ambient Assisted Living. First, we describe each one, discussing their service coverage range. As our investigation focuses on expanding service coverage, we selected AAL approaches that were designed with user mobility and the migration process in mind. The Gator Tech Smart House also appears because it represents one of the first architectures that proposed mechanisms for extrapolating the well-defined space of the house, without considering the provision of services beyond the house boundaries.

## 3.1 Gator Tech

The *Gator Tech Smart House* [49] is a project proposed by the University of Florida's Mobile and Pervasive Computing Laboratory, which consists of a programmable pervasive space. *Gator Tech* implements the *Atlas middleware* [54], which is a reference middleware architecture that proposes a service-oriented smart space. Figure 3.1 describes the composition of the *Gator Tech* reference architecture, which comprises six layers. On the bottom, the physical layer consists of the various devices and appliances the residents use such as blenders, TVs, doorbells and lamps. Additionally, sensors and actuators such as smoke detectors, air conditioning and heating thermostats are part of the physical layer. The sensor platform layer provides an interface between the physical layer and the service layer, converting each device (e.g., sensors, actuators and complex monitors) to an OSGi bundle service [56]. Above, the service layer implements basic services such as service discovery and service registration and allows the creation of services according to the domain, for example, voice recognition, media streaming and scheduling. At the same level, the context management layer provides mechanisms to create and to register contexts of interest. In addition, it has a context engine able to interpret context data. The knowledge layer maintains the ontologies of the services, applications and devices, which are used for registering and discovering services. On top, the application layer comprises an application manager to activate and deactivate services and an integrated development environment (IDE). The IDE is graphical-based and consists of a set of tools to assist in the creation of smart spaces.

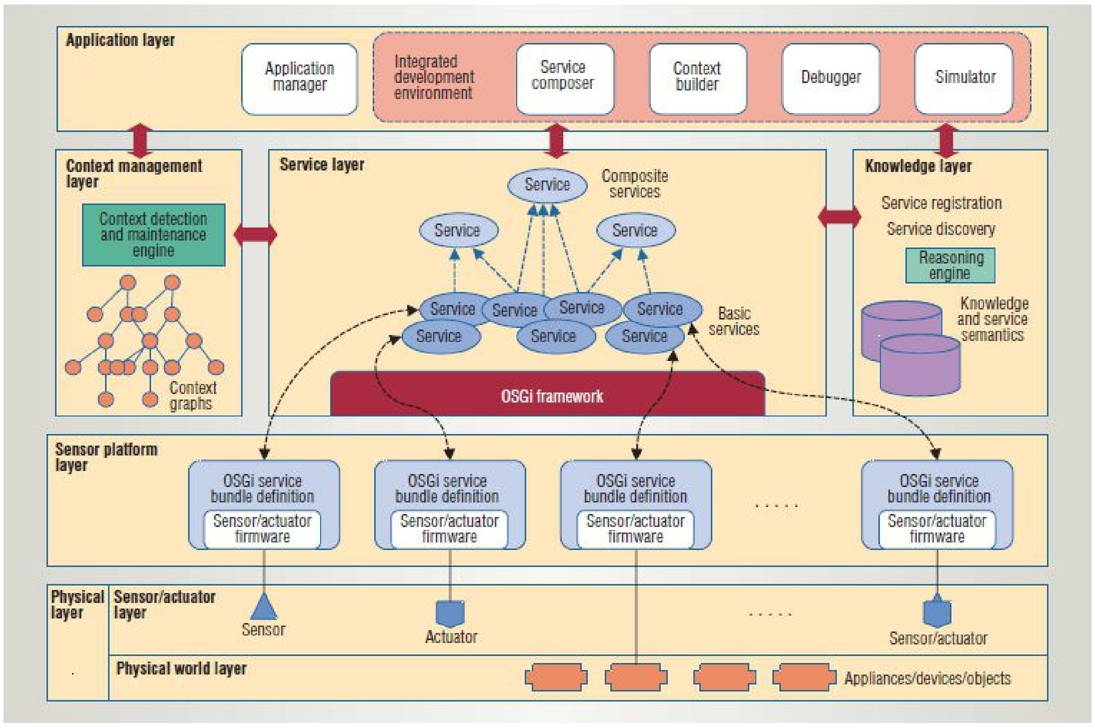


Figure 3.1: *Gator Tech* layers [54]

**Service Coverage.** Services running on top of *Gator Tech* are available only in-house. The author does not mention the possibility of provisioning services outside the area of the house. The reference architecture also does not employ components to handle user migration or even communication with another smart space In [48] the *Gator Tech* was modified to support remote monitoring, which was an attempt to expand the range of services running outside. In this case, the smart house implements services to assist elderly people, especially services that attempt to provide a remote presence of the resident's caregivers within the house, as well as providing notifications in cases of urgency. However, they did not mention services that continue working when the patient/resident is off-site.

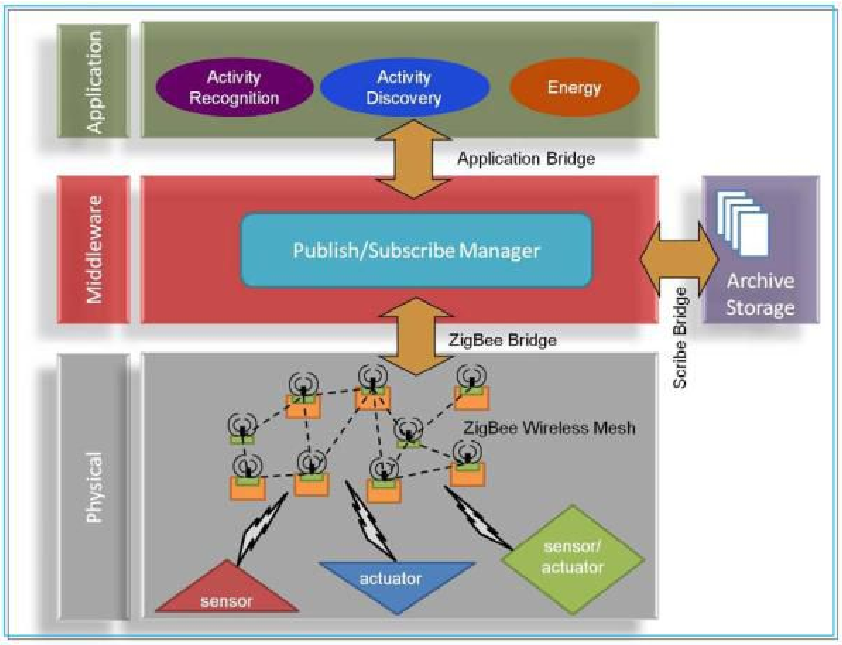


Figure 3.2: *CASAS'* architecture [21]

## 3.2 CASAS

CASAS is a project from Washington State University that aims to create a "smart home in a box". According to [21] the smart home kit fills in a small box because it is designed to be small in form, lightweight in terms of the infrastructure, extensible with minimum effort and easily installable. The CASAS architecture comprises the application layer above two more layers, the middleware layer and the physical layer, as depicted in Figure 3.2. The physical layer represents the hardware components including sensors and actuators, which are connected by a ZigBee wireless mesh [8]. The middleware layer is governed by a publish/subscribe manager, which provides named broadband channels that allow component bridges to publish and receive messages. Every architecture component communicates via a customized XMPP bridge to this manager. As indicated in Figure 3.2, the *Zigbee bridge* and the *Scribe bridge* are examples of such customized bridges as well as bridges for each software component in the application layer. In addition, the middleware provides valuable services such as adding time stamps to events and maintaining the side-wide sensor state.

Finally, in the application layer, some core services are executed such as activity recognition, activity discovery and energy. These services are deployed by CASAS to provide services focused on the needs of the residents; intelligent systems require information about the activities being performed by the residents. They employ machine learning techniques to label and to classify data streams collected from sensor data according to the activity performed by the resident. In addition to these core components, many applications can be deployed, including: (i) health assistance: using smart phones to prompt individuals to initiate important daily activities [27]; and (ii) energy efficiency: supporting energy-efficient behavior in the home [21]. In addition, the resident can visualize an overview of the house operation using a smart phone because the applications also provide a web-based user interface.

**Service Coverage.** *CASAS* proposes the concept of bridges, which are used for integrating different layers within the architecture. It also may be used for connecting other CASAS spaces. Bridges can be created, configured and integrated on the architecture without changing or even restarting the middleware. Therefore, bridges for linking multiple smart homes together can be created, allowing the ability to scale to communities of smart homes. In this case, the same service can be provisioned in different smart homes but not outside of them, i.e., between two smart homes, the services are not available. Although this approach seems promising, in [21], there is no information about how to link multiple environments using customized bridges. Indeed, there is no discussion about the handover process in this environment. As there is no resident migration between two spaces, we can observe that CASAS supports roaming instead of the handover.

## 3.3 Persona

Persona [74] is a project funded by the European Consortium aimed at developing a scalable open standard technological platform to build a broad range of Ambient Assisted Living services. Designed to support connectivity and interoperability, this AAL Platform proposes a multilayer middleware based on an OSGi (Open Service Gateway initiative) dynamic modular system, which comprises a service-oriented bus-based reference architecture for AAL spaces. The components of the reference architecture are depicted in Figure 3.3.

The components handle information from/to the environment through the following buses: the *Context Bus*, *Service Bus*, *Input Bus* and *Output Bus*. The *Dialog Manager* can be considered the central component of the architecture because it manages the system-wide dialogs and hides from the users the complexity of using the application services, providing a rule-based mechanism for associating service calls and situations. The historic data of all context events is gathered by the *Context History Entrepôt* in a central repository, which allows the *Situation Reasoner* to infer new contextual information and for such information to persist. In addition, depending on the application, other *special-purpose reasoners* can be instantiated for addressing the system requirements.

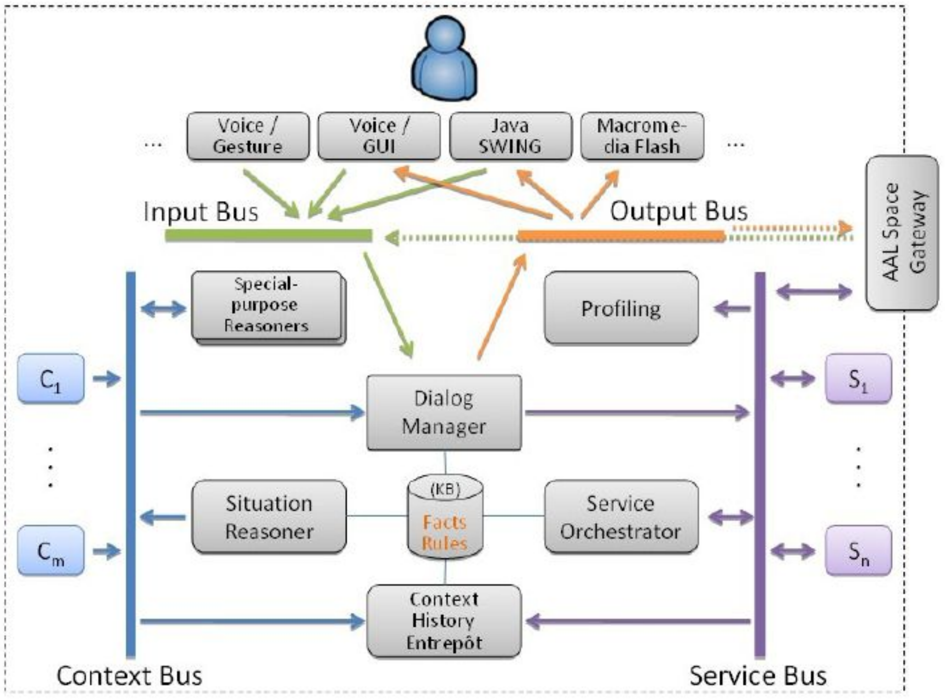


Figure 3.3: *Persona* reference architecture [74]

A service on the platform may exist only at a meta-level; hence, the *Service Orchestrator* is in charge of interpreting this metadata describing a composite service and performing the instructions within it, implementing that service and registering it on the appropriate bus. In a modular and dynamically evolving system, it is essential to be able to utilize the whole potential of the system on a meta-level that does not necessitate any sort of re-compile, re-build, re-deploy, or restart of any other component when new components are added or old ones are removed from the system. Components for managing user profiles (*Profiling*) and external connections (*AAL Space Gateway*) are also provided by the platform, as well as user interfaces and administration tools.

**Service Coverage.** To support the communication with external entities, this *AAL Platform* employs a special-purpose component called the *AAL Space Gateway.* This gateway may provide a link between the AAL space and a resident that is outside, providing operations such as remote notification and remote access. In addition, it enables the bridging between AAL spaces, as well as external service providers to advertise their services to the residents, such as on-line movies, video conferences with doctors, and nutritionist cooking assistance.

Services hosted within the AAL space are available under a fixed URL because the gateway acts within the AAL space as an interface integrator able to forward external requests to input buses and to return results from the output buses. The phone I/O of the gateway may be selected for supporting the interaction between the absent resident and the house, such as sending service notification by SMS. Furthermore, Web Services can be employed as an interface for third-party service providers advertising their services and solutions.

## 3.4 Uranus

Proposed by ICAR-CNR (Istituto di Calcolo e Reti ad Alte Prestazioni), Uranus is a middleware infrastructure that provides a set of basic services for rapid-prototyping and the development of Ambient Assisted Living and vital sign monitoring applications [26]. As illustrated in Figure 3.4, the Uranus architecture is based on a two-tier model composed of the residential gateway and the mobile-tier. While the residential gateway is implemented over the OSGi standard platform for integrating component sensors and stationary devices at home, the mobile-tier supports mobile devices, which may be located indoors or outdoors, using the J2ME platform for integrating the devices in the architecture.

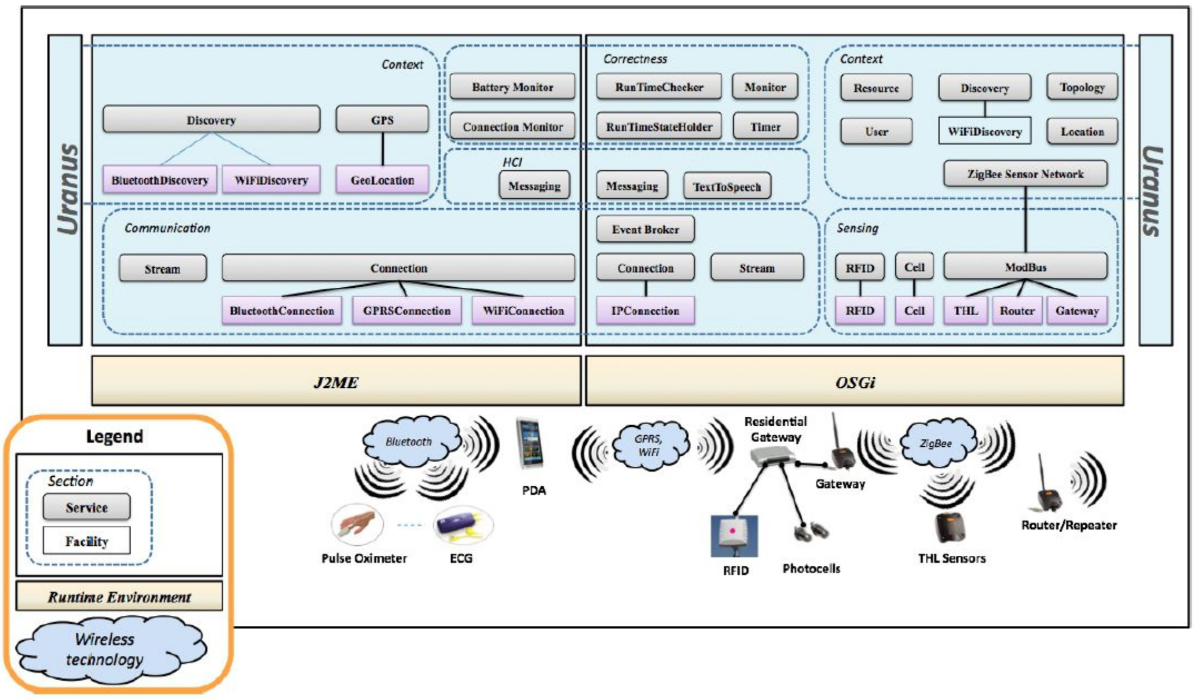


Figure 3.4: *Uranus* architecture [26]

In the residential gateway, services for handling context information are disposed in the *Context Section*. In particular, the *Topology Service* and *Location Service* manage information about the location of users or devices (sensors). Locations are classified as semantic locations (e.g., rooms, building, floor, specific parts of a room) and physical locations (e.g., the area covered by a Wi-Fi access point). While the *Topology Service* provides a unique and uniform representation of the topology of the environment, the *Location Service* is in charge of handling the physical location information about the resident and devices, considering this topology representation. Managing topology of the space allows Uranus to expand the house domain by controlling the resident migration through the environments.

Through a Bluetooth network, sensors worn by the resident send data to a mobile device (e.g., PDA), which then sends these data to the *Residential Gateway* using either the domestic Wi-Fi or the GPRS network. Thus, the resident's personal mobile device performs a central role in the Uranus architecture, in particular for health monitoring applications. Applications that are implemented on top of the *Residential Gateway*, use all the components provided by Uranus such as *Stream Service* for handling streaming data and *(*Event Service) for exchanging messages with other components. Additionally, depending on the domain application, other components may be implemented and deployed on Uranus.

In the mobile-tier, the core components for the discovery services and obtaining resident location (e.g., GPS) also compose the architecture, although in specialized versions, such as *Bluetooth Discovery* and *Wi-Fi Discovery*. When the resident moves from one environment to another, the mobile device tends to connect to a different wireless network, which may be of a different technology as well. Additionally, wireless networks (e.g., Wi-Fi, Bluetooth, GPRS) usually have constant connection issues. In this context, the *Connection Monitor* identifies the connections issues or interruptions and further informs the Connection component, further handling device connection by specialized components, such as *Bluetooth Connection*, *GPRS Connection* and *Wi-Fi Connection*.

Mobile devices play an important role in Uranus because the critical services for the resident are provisioned by them, for example, a heart monitoring system running on the mobile-tier of Uranus, in particular in a PDA. However, they also represent the most critical part of the architecture, especially with respect to the high energy consumption of devices such as PDAs and smart phones. To minimize such issues, the *Battery Monitor*, in conjunction with the *Coordinating Midlet*, provides a mechanism for switching the role of the main mobile device from one device to another without restarting or reconfiguring the Uranus. For example, the *Battery Monitor* alerts the *Coordinating Midlet* when the user's PDA reaches a certain battery threshold, next the *Coordinating Midlet* switches from the current PDA to a spare PDA (if the user is equipped with one).

**Service Coverage.** Because a mobile device does not depend on the *Residential Gateway* for running services in Uranus, the resident is able to move freely between different environments, even with some limitations such as limited battery energy of the smart phone and dependency of an Internet connection. The *Connection Monitor* on the mobile-tier is in charge of identifying the connection lost and triggering the start of the resident migration. For the long-term health monitoring application displayed in [26], the *Monitor Connection* checks the presence of the user by the availability of the domestic Wi-Fi network. This is possible because the user is equipped with a PDA that supports connection with both Wi-Fi and GPRS. Then, when the *Monitor Connection* detects a Wi-Fi connection failure, it requires the *Connection Service* to switch the PDA connection from Wi-Fi to GPRS. On the other hand, when the user is at home and the domestic Wi-Fi network reveals the availability, *Monitor Connection* requires the *Connection Service* to switch the connection back.

## 3.5 Greener Building

Greener Building [30] is a project funded under the European Seventh Framework Programme that proposes a service-oriented approach to design and develop a building management system (BMS) focused on energy-ware adaptation of public buildings. For that purpose, the energy management system of the *BMS* is based on activity sensing. Regarding architecture, as illustrated in Figure 3.5, it is composed of the following layers: Physical, Ubiquitous and Composition.

The *Physical layer* handles the devices of the system by the *Sensor/Actuator Gateway*, which presents raw information from the sensors and actuators in a uniform manner to the upper layers. The other component of this layer, the *Interconnection with Smart Grids*, as its name indicates, is responsible for interconnecting the middleware with Smart Grid systems. Thus, this component provides to the building the ability to be aware of the external pricing of energy providers and internal vs external availability. User activity is recognized, abstracted and managed in the *Ubiquitous layer*. For this purpose, it employs three services: repository, context and planner. The *Repository* is the central database of

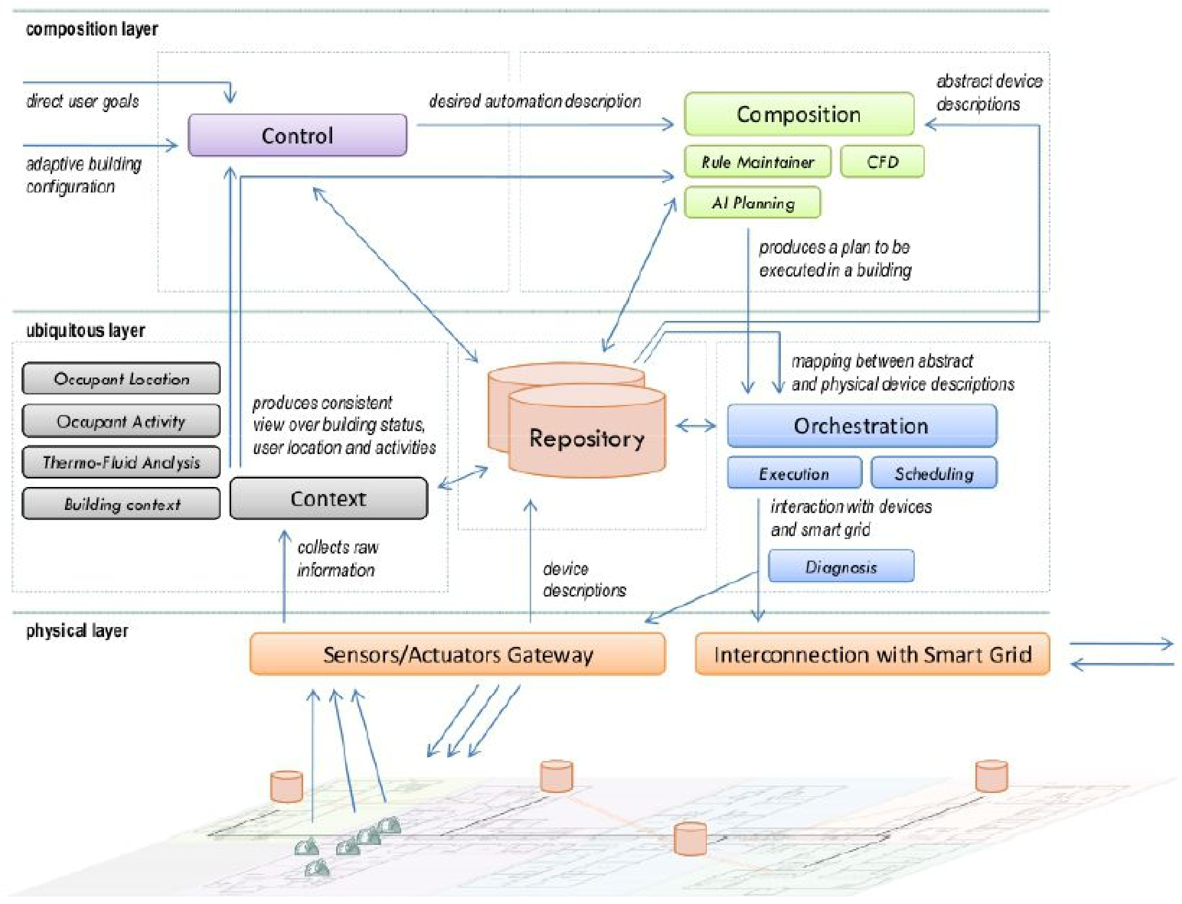

composition layer
direct user goals
adaptive building configuration
Control
desired automation description
Composition
abstract device descriptions
Rule Maintainer
CFD
AI Planning
produces a plan to be executed in a building
ubiquitous layer
Occupant Location
Occupant Activity
Thermo-Fluid Analysis
Building context
produces consistent view over building status, user location and activities
Context
Repository
mapping between abstract and physical device descriptions
Orchestration
Execution
Scheduling
interaction with devices and smart grid
Diagnosis
collects raw information
device descriptions
physical layer
Sensors/Actuators Gateway
Interconnection with Smart Grid

Figure 3.5: *Greener Building* layers and components [30]

the system. Smart building environments usually are composed of a set of smart devices, such as light sensors, smart plugs, actuators, smart doors, and central heating. Hence, the *Repository* stores information about such devices (configuration, states, available actions and energy consumption) and historical information about environment states (for further analysis). The *Context* collects sensor data, transforming it into high-level information, such as user activity and the building's overall state. The Orchestration executes the actions chosen by the Planner. Such actions are abstracted as web services.

The *Composition layer* contains a *Control service* and a *Composition service*. The former represents the system's user interface, including web interfaces, smart phone applications, tablet applications, dashboards, etc. The *Composition service* represents the main reasoning and decision making of the system. It is composed of the following components: *Rule Maintainer Engine (RME)*, *AI Planning* and *Computation Fluid Dynamics (CFD)*. The *RME* gathers information about user preferences and the environment context for deciding the goal state at which the system should be transitioned. The *AI Planning* is a reasoning mechanism to find actions to be performed by the actuators to achieve the goal state indicated by *RME.* Finally, the *Computation Fluid Dynamics (CFD)* handles the heating part of the building.

**Service Coverage.** Greener Building (GB) may support many different spaces - not only rooms within a building but also a set of buildings - because its service-oriented architecture stores all the services information in a centralized database and can instantiate services considering a physical location. Additionally, for managing distributed service instances, GB employs the *Distributed Configuration Service (DCS).* It is responsible for handling information about addresses, parameters, and the physical location of every system service instance.

As previously mentioned, the *Repository* centralizes the data of the system, managing different store needs, which are classified as building description, service description and context history. It is worth mentioning that the *Building description* is a hierarchical tree-structure representation of the building, including its physical locations. Because the *Repository* uses the concept of a *cell* for definition, a physical location contains other devices or other cells. Because of this conceptual representation, the GB supports even bigger abstractions of buildings such as university campuses, industrial parks and office blocks.

For example, a university campus is composed of two faculties (A and B) and an outdoor open area connecting them. While the campus represents the general and abstract environment, each faculty and the open area represent the specific area where the user can be. A service provisioned in faculty A can be provisioned in faculty B and in the open area because GB allows the movement of entities through the building semantic locations. Therefore, GB allows expanding the service coverage of the building, thus allowing the movement of users and entities through the faculties and the open area.

In conclusion, GB seems a promising approach for supporting resident migration between two or more intelligent environments, in particular because it proposes components to manage locations hierarchically. Hence, the GB provides the opportunity to project a large intelligent environment composed of a set of other small ones. However, in [29], only a simple building description is evaluated.

## 3.6 Discussion

In this chapter, we analyze the architectural aspects of the approaches for AAL discussed above in regards to resident migration between two or more AAL Spaces, and in particular, the components that can deploy a handover protocol, even a simple one, for expanding the service coverage of an AAL Service Platform. In this context, we presented the Gator Tech, which together with Gaia[70] and MavHome[23], may be considered the one of the first middleware systems to support an implementation of AAL systems because they propose services integration toward the resident within a home domain. In fact, Gator Tech proposes a middleware based on OSGi bundles that inspire the new service-oriented smart environment architectures. We also discussed four architectures that handle the handover using specific components or services, which are CASAS, Uranus, Persona and Greener Building, thus providing a certain level of mobility for the residents.

Among them, it may be said that CASAS proposes lightweight mechanisms for providing a link between AAL spaces. As the main purpose of CASAS is to be extensible, easily configurable and portable, its bridge-based architecture allows integrating new layers, new components and even new bridges. Because of such characteristics, other CASAS environments may be connected with the house through a bridge created for such a purpose.However, CASAS cannot be considered to support resident migration and is only concerned with connecting two or more CASAS environments. In addition, CASAS does not consider provisioning services in an area not empowered by CASAS architecture, for instance, an outdoor open space area between two CASAS environments. Additionally, CASAS does not propose a lightweight architecture version for running in mobile devices during the resident stay outside the house. Hence, CASAS is not able to implement a continuous service such as the health monitoring systems proposed in Maria's Scenario(Section 2.1).

Uranus and Persona best address provisioning of services outside the house, allowing a resident to perform activities not only in outdoor spaces but also indoors. Because Uranus was designed to support monitoring applications, mobile devices play a central role within the architecture, in particular because they gather data streams from the wearable sensors and handle the network handover through the *Connection Monitor*, which switches the mobile device connection between Home Wi-Fi and GPRS. Additionally, Uranus abstracts the AAL space using the *Topology Service*, which stores hierarchically the physical locations of the environment. Therefore, an AAL space may be composed of rooms, outdoor spaces, other houses, etc. In a somewhat different way, Persona employs a component called the *AAL Space Gateway* for establishing a link not only with external services and absent residents but also with other AAL spaces. The gateway interfaces external service providers (e.g., third-party health monitoring enterprises, hospitals, clinics) on the correct buses in the AAL space; therefore, these external services are able to act as local services. In addition, mobile devices (e.g., smart phones, tablets) may gather data streams from wearable sensors or act as a user's input and output interface.

Greener Building also proposes a component to manage the environment location abstraction, the *Building description*. This makes it possible to manage the user migration between the physical spaces that compose the general abstract location. As Greener Building reaches all such physical spaces, we can affirm that the service coverage of GB is directly related to the number of spaces that compose the general abstract location. Additionally, handover protocols should be deployed to migrate mobile entities (e.g., user and services) between two different physical spaces that are part of the GB coverage. Similar to Uranus, Greener Building stores in a central database the location information about the physical spaces of the environment. As with Persona, mobile devices (in such cases as smart phones) act as the user interface within a Greener Building environment, also providing centralized control of the system. While Uranus, CASAS and Persona propose a centralized reasoning, Greener Building proposes a distributed reasoning because its services may be spread out through the smart space in such a way that each service is in charge of executing its purpose independently. In addition, although Greener Building does not focus on AAL spaces, its architecture composition may support AAL services.

Table 3.1 presents a systematic comparison of the aforementioned AAL Service Platforms. After discussing the differences and similarities of the related work, we discuss in the next chapter some handover approaches in the Ambient Assisted Living domain considering our scenario.

Table 3.1: Comparison of Smart Environment Architectures

| | **CASAS** | **Persona** | **Uranus** | **Gator Tech** | **GreenerBuilding** |
|---|---|---|---|---|---|
| **Handover** | Yes | Yes | Yes | No | Yes |
| **Handover Approach** | Bridge | Gateway | Mobile device | - | |
| **Handover Scope** | Direct | Direct | Indirect | - | Direct |
| **Handle Data Stream** | Yes | Yes | Yes | Yes | - Yes |
| **Communication failure** | | | | | |
| **System Model** | MAS | SOA | SOA | SOA | SOA |
| **Reasoning** | Centralized | Centralized | Centralized | Centralized | Distributed/Local |
| **Mobile Devices** | Consumer | Consumer | Provider | - | Consumer/Interface |

# Chapter 4

# Handover Approach in AAL

In this chapter, we discuss two basic scenarios in which handover techniques are useful for expanding the service coverage of the AAL systems. We classify the handover patterns in such scenarios as the Direct and Indirect Handover. We call the Direct Handover the handover in a scenario in which a resident migrates directly from one AAL Space to another. The Indirect Handover, on the other hand, consists in providing service coverage even when the resident is going through a blind area. The most important challenge of Direct and Indirect Handover is to provide the same service within different environments regardless of whether they are empowered by AAL Platforms.

To present our handover approaches by classifying the AAL Platform displayed in Related Works (Chapter 3), we relate Maria's scenario (Section 2.1) with those systems, discussing how their features can address the necessities presented in the story. This scenario will be referred to throughout this work; it describes the story of Maria, which is a generic scenario that contemplates a long-term monitoring of vital signs, the AAL systems for supporting daily activities and an application of a handover that is both direct and indirect. Finally, we suggest some architectural components that an AAL Platform should implement for supporting direct and indirect handovers.

## 4.1 Handover Patterns

Residents of AAL spaces may perform activities outside their homes, either in outdoor spaces or indoor spaces where there is no smart infrastructure or in spaces equipped with AAL platforms. Stadiums, supermarkets, parks, squares, streets, etc. are examples of the former case. On the other hand, smart gyms, smart offices, smart homes and smart hospitals can be considered examples of indoor spaces equipped with AAL platforms. We call the Direct Handover the act of seamlessly providing service for the resident when he migrates from one AAL space to another, the service coverage areas of which intersect each other. We call the Indirect Handover the act of providing service coverage when the resident migrates to a different AAL space and goes through some spaces where there is no smart infrastructure, i.e., a blind area between the different AAL space coverage areas.

## 4.1.1 Direct Handover

In Maria's scenario there is no intersection between the smart gym and the smart house coverage areas. However, to discuss the direct handover, in this subsection, we will consider that these AAL spaces are contiguous to each other, and hence, the smart home's service coverage area intersects the smart gym's service coverage area. In Figure 4.1, this concept and the environments are represented.

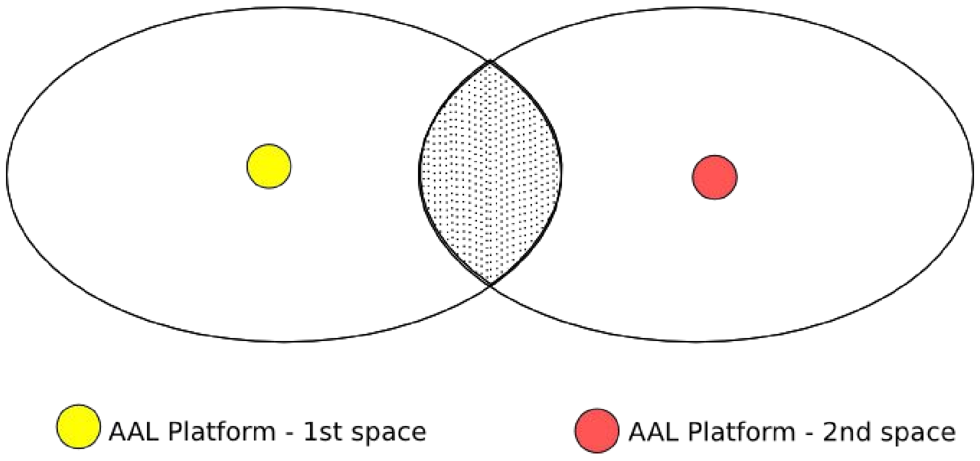


Figure 4.1: Overview of the direct handover including two AAL Spaces (ellipses) with their respective AAL Platforms.

The migration of Maria from the smart home to the smart gym represents a Direct Handover situation. Here, the handover process is executed to transfer the service provisioning from the house to the gym and vice-versa. For example, during the handover process, the smart home establishes a connection with the smart gym, downloads the necessary profiles to configure the continuous provision of services and assumes the role of providing such services. Then, Maria can move through these AAL spaces enjoying her personalized long-term heath monitoring service.

As the monitoring service infers the user's health status using data mining classifiers, the smart gym AAL Platform must acquire the last version of this decision model to provide the service to Maria. In addition, when Maria leaves the gym to come back home, the handover process is executed again. It becomes possible to retain the historical data of Maria's monitoring in both AAL spaces, thereby storing data that may be used to improve the monitoring system, in particular its decision model.

### 4.1.2 Indirect Handover

Maria also performs activities in open spaces such as going to the supermarket, walking in the park and spending time window shopping with her friends. Usually, such spaces are not covered by AAL services. In Figure 4.2 we illustrate the blind areas by displaying a set of AAL spaces disposed in an open area.

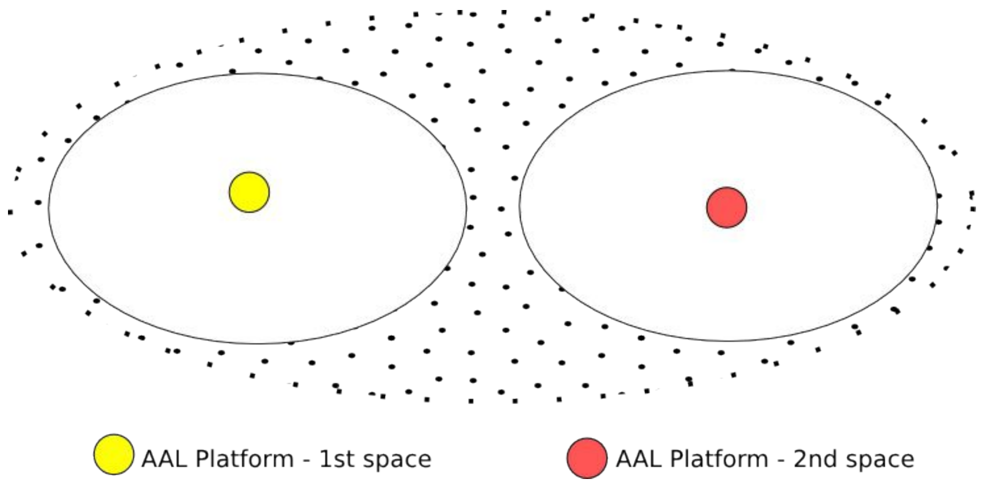


Figure 4.2: Overview of the indirect handover including two AAL Spaces (ellipses) with their respective AAL Platforms and a blind area (pattern of points).

As the long-term monitoring should be provided even during outdoor activities, we can classify this process as an indirect handover. To provide such monitoring outside of the house, a smart phone must be used as a temporary service provider, one that gathers and analyzes data from a smart bra worn by Maria. A presence detection mechanism shall be necessary for controlling who acts as provider, either the smart home or the smart phone. Similarly, in a direct handover scenario, the decision model update is a critical task that has to be performed before the service starts. Additionally, the learning mechanism update has to be executed when she returns; however, it may be performed anytime during her stay within her house.

## 4.2 Functional Architecture

Based on the concepts of Direct Handover and Indirect Handover, we analyzed the mechanisms that the several architectures discussed in Chapter 3 bear for addressing the service handover in the AAL domain.

Greener Building (GB) and Persona have components for addressing the direct handover scenario. GB contains a component responsible for storing and managing the semantic organization of a smart environment and a component for managing the occupant location. Additionally, its services may be distributed along those semantic locations, which can be within the same building or not. Because of this, the resident can move from one semantic location to another while retaining the continuous monitoring system.

Beyond the semantic organization of the AAL space, Persona employs a *AAL gateway* for connecting different AAL spaces and then provides such a direct handover. Unlike GB and Persona, CASAS does not manage the semantic organization of the AAL space but proposes generic bridges that may be created for different purposes such as connection between two CASAS spaces. Although there are no examples of bridges for AAL space connections, the publish/subscribe nature of the middleware and the support of different network technologies indicate the feasibility of designing a bridge able to handle resident migration between two CASAS spaces.

Persona and Uranus employ components to handle the provision of services for absent residents, thus supporting the Indirect Handover. Using the *AAL Space Gateway*, Persona allows external connections not only from another AAL Space but also from mobile devices such as smart phones and tablets. Hence, it is possible to deploy remote services within the smart home as well as to support the user's migration from inside to outside the home. In Persona, every available service acts as a local service, thanks to the gateway, which interfaces, for example, the messages from services running outside the *Input Bus* and from the *Output Bus* to services running outside. However, the authors do not provide an example demonstrating the user's migration to the outside of the house, and there are no discussions regarding migration processes or handover protocols.

Uranus proposes an architecture focused on mobile devices with components dedicated to the connectivity outside the house, battery monitoring and network switching. Hence, the Indirect Handover appears as a native feature of Uranus because mobile devices act as service providers both inside and outside the house. As a consequence, the seamless migration of a user between the home and an outdoor space is already provided by Uranus.

The *Connection Monitor* starts a sort of handover process when the mobile device has a connection interruption with the current network. For example, a mobile device is connected in a Wi-Fi network inside the smart home, and the resident leaves the house, thereby interrupting the Wi-Fi connection. This event triggers a simple handover process for restarting the mobile device connection using a GPRS network. In this context, the mobile device expands the service coverage of the AAL space even into the blind areas.

Considering the features of the previously discussed systems and the requirements that can be elicited from Maria's scenario, we conclude that the architecture of the proposed AAL Platform should implement the following components illustrated in the Figure 4.3:

- **Space Abstraction:** Component responsible for storing and managing the semantic organization of the AAL platform handover range, which represents the AAL spaces connected with the home including a generic open space. Only within such spaces can a service be provisioned. GB and Persona already employ a semantic organization of the space, which focuses on the internal organization (e.g., the living room, kitchen, bedroom). However, here, we represent spaces outside the house (e.g., gym, office, other house) that support the AAL systems with an external handover.

- **Location Service:** Component responsible for managing the context information about the user position considering the semantic organization. For example, the gym attended by Maria is powered by an AAL platform and supports the external handover, and it registers her presence there. Such information will be useful for managing the settings of the service. It may be implemented by a general context service.

- **Space Gateway:** Component responsible for handling connection requests from either other AAL spaces or the resident's mobile device and for establishing a connection among those entities and the home. Thus, the home is able to share the resident's profiles with other AAL spaces and to send messages (i.e., alert messages) to the resident's mobile devices.

- **Handover Handler:** Component responsible for managing the provision of services according to the user's location. A service may have different settings according to a semantic location. For example, the continuous health monitoring in Maria's story has particular executions if Maria is at home or outside, and then, the Handover Handler switches the service from the house to the smart phone when she leaves the house. Additionally, when the Smart Gym detects Maria's presence, it requests the transference role of managing the continuous monitoring from the smart phone.

- **Synchronization Service:** Although a single service may be provided by one or more AAL spaces, the house acts as the main provider because the learning mechanism is centralized and available just at home. Thus, the necessity of the component to manage the synchronization of profiles, settings and decision models between the home and the other providers is clear. In addition, such a component extracts the historical data produced by the other providers and stores the data in the centralized database, which will be used during the learning process.

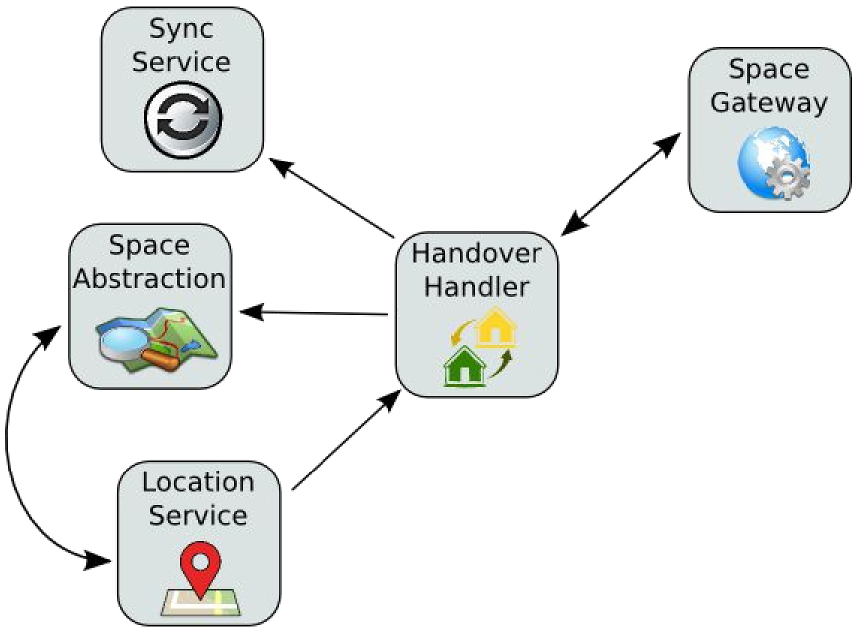


Figure 4.3: Functional architecture's components for supporting handover in an AAL Platform

The aforementioned components together do not compose a complete framework architecture for developing AAL Services. They should be part of an architecture able to support the handover patterns discussed in this section.

## 4.3 Discussion

In this chapter, we discussed the patterns of the handover in Ambient Assisted Living, which include the Direct and Indirect Handover. We used a fictional scenario for representing features presented in each type of handover. We also suggested components that should be part of an AAL Platform for supporting the handover. It is worth mentioning some challenges related to the handover in the AAL spaces. In the next chapter, we propose an approach to deploy direct and indirect handover in an AAL platform. For this purpose, we extend the framework for developing Ubiquitous Systems proposed by the Tempo Laboratory [59] [1]. Essentially, this framework aims at supporting the prototyping, development and management of ubiquitous applications for intelligent environments [59, 34].

[1]http://www.tempo.uff.br/

# Chapter 5

# Our Approach

In this chapter, we propose an AAL Platform, a framework to support the handover in AmI systems. For this purpose, we extend the flexible framework proposed in [12], which aims at prototyping, developing [59, 34] and managing ubiquitous applications for intelligent environments. In the following sections, we refer to this flexible framework as *UbiTempo* for clarity. For the same reason, we refer our approach as *UbiTempo+*. Based on the discussion about service coverage and handover approaches in Section 4.2, we propose some architectural and conceptual modifications in the *UbiTempo*.

In the next section, we present the *UbiTempo* together with its reference implementation *Smart Android* and discuss why it can be considered an AAL Platform. Next, we correlate the suggested handover components with the *UbiTempo* architecture, indicating which one should be deployed and how. Finally, we present our framework proposal, the *UbiTempo+*.

## 5.1 Reference Framework: UbiTempo

*UbiTempo* is a project from the Tempo Laboratory of the UFF (Federal Fluminense University) that consists of a framework for prototyping, developing and managing ubiquitous applications in intelligent environments. The general architecture of the conceptual framework was proposed by [59], the context rules composition was proposed by [34] and the GUI for prototyping and managing was proposed by [12]. The framework project began from ideas discussed in the SCIADS project [24]. SCIADS combines the concepts of Ubiquitous Computing for providing health services to patients (e.g., health monitoring and emergency triggers) in a domestic environment [25]. The UbiTempo attempts to fill the gaps opened by SCIADS and, in particular, is an approach that is able to support the development of ubiquitous applications not only for health services at home but also general services such as security, entertainment, accessibility, and energy saving.

The general architecture of the reference framework includes a Distributed Components Model [17], which defines the Resource Agent structure and the Resource Management Support Services, which offer basic management services to provide resource registration, discovery and localization, as also depicted in Figure 5.1. Additionally, the *SmartAndroid* project is a middleware built on top of the reference framework that includes a programming API to help the developers construct Ubiquitous Applications [7].

The *SmartAndroid* has been developed over Google Android platform since its authors believe that in a near future it can be widely used in sensors and devices embedded in furniture and home appliances. In addition, this option helped us to implement emulators of several devices using cheap Android smart phones and tablets [59].

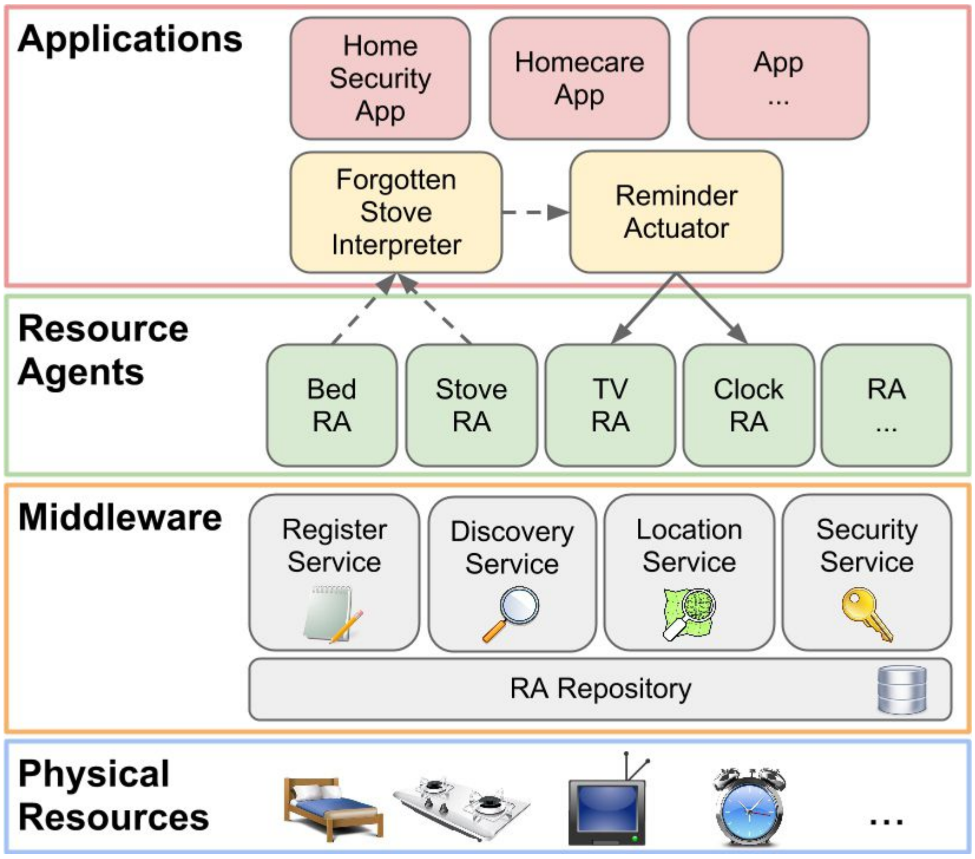


Figure 5.1: Reference architecture of the framework to develop ubiquitous applications [59, 34], the *UbiTempo*

Resource Agents (RAs) are entities that represent resources in an intelligent environment. They encapsulate the specificities of resources and expose their information to the environment through standard interfaces so that others can access them uniformly, thus reducing the complexity of integration. The first RA interface provides access to Context Variables (CV), and it is responsible for exposing the context information from the RA. The second one is called Operation (OP), which is responsible for exposing internal attributes (reading) or activating internal attributes (writing) belonging to RAs, enabling applications to manage the RA's internal state and explicitly to interact with the environment.

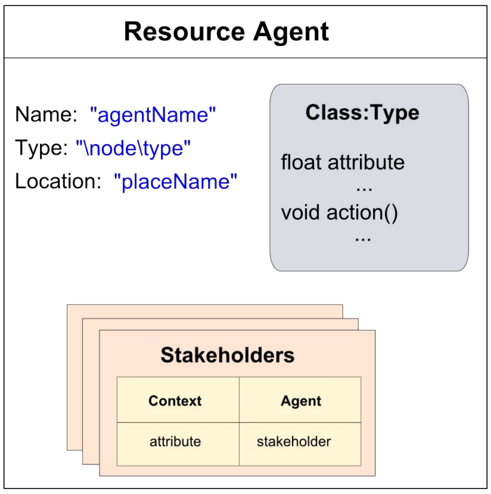


Figure 5.2: Description of a Resource Agent

The reference framework also defines classes of Resource Agents to represent the type of the resource with specific interfaces. In addition to name and location, the RA's structure has an attribute named type that represents a hierarchy of classes inspired by the ontology described by Ranganathan [69]. A visual description of a Resource Agent is displayed in Figure 5.2.

As depicted in Figure 5.1, the reference architecture of the framework comprises four layers. The Physical Resources Layer is the deepest layer. It includes the resources present in the environment, which, in an intelligent environment, are the smarter versions of daily objects, such as beds, stoves, TVs, and clocks. The Middleware Layer includes the main services that sustain the interoperability of the RAs. Moreover, it provides basic features for the development of Ubiquitous Applications, such as, managing services and the RA Repository, which maintains information about the RAs (addresses and descriptions) and the map data structure. The Resource Agents Layer is composed of all the RAs that represent the physical resources that are already living in the intelligent environment. Finally, the Application Layer, which is the topmost layer, includes all Ubiquitous Applications that use the middleware features and also all the software level RAs, which are represented in the figure by the "Forgotten Stove Interpreter" and the "Reminder

Actuator".

Regarding communication, the framework proposes two mechanisms: a standard Remote Procedure Call (RPC), which implements request-reply communication (synchronous); and a publish-subscribe mechanism (pub-sub), which implements asynchronous communication. In addition, the framework includes an *RA Name System* (RANS), which provides a unique identification for the agents in the intelligent environment. This identification is directly mapped to the Internet DNS; thus, the names remain the same even if the network address (IP) changes. The only change is the updating of the addressing table. AAL Systems that need a handover are not supported by UbiTempo. The reference framework does not consider joining one or more intelligent environments for expanding its service coverage. UbiTempo is also not concerned with provision services for the resident outside the well-delimited space of the home. As a consequence, UbiTempo has no components for handling resident migration between environments and for establishing a connection with other intelligent environments or external entities, such as third-party services and remote services. Additionally, each entity in the intelligent environment, in particular AAL systems, should know if it has handover support and which type of handover it implements.

## 5.2 Extended Framework: UbiTempo+

Our main goal is to design a framework for developing AAL services with handover support. Because the *UbiTempo* has some limitation in regard to handover support for AAL services, we propose to implement some architectural components in the *UbiTempo* in accordance with the requirements discussed in Chapter 4, which are: *Space Abstraction*, *Location Service*, *Space Gateway*, *Handover Handler* and *Synchronization Service.* Hence, the extended framework, which we designated UbiTempo+, should have the components to handle the communication between intelligent environments, for resident migration to another intelligent environment or to outdoor spaces not empowered by an AAL Platform. The Figure 5.3 illustrates the architecture of the *UbiTempo+*, in which we include a service for handling agent migration to another environment on the middleware layer of the *UbiTempo.* This service is called *Handover Service.* To provide communication with external entities (e.g. AAL Spaces, third parties and mobile devices), we propose a service called *Space Gateway*, which is based on the homonymous component of the functional architecture.

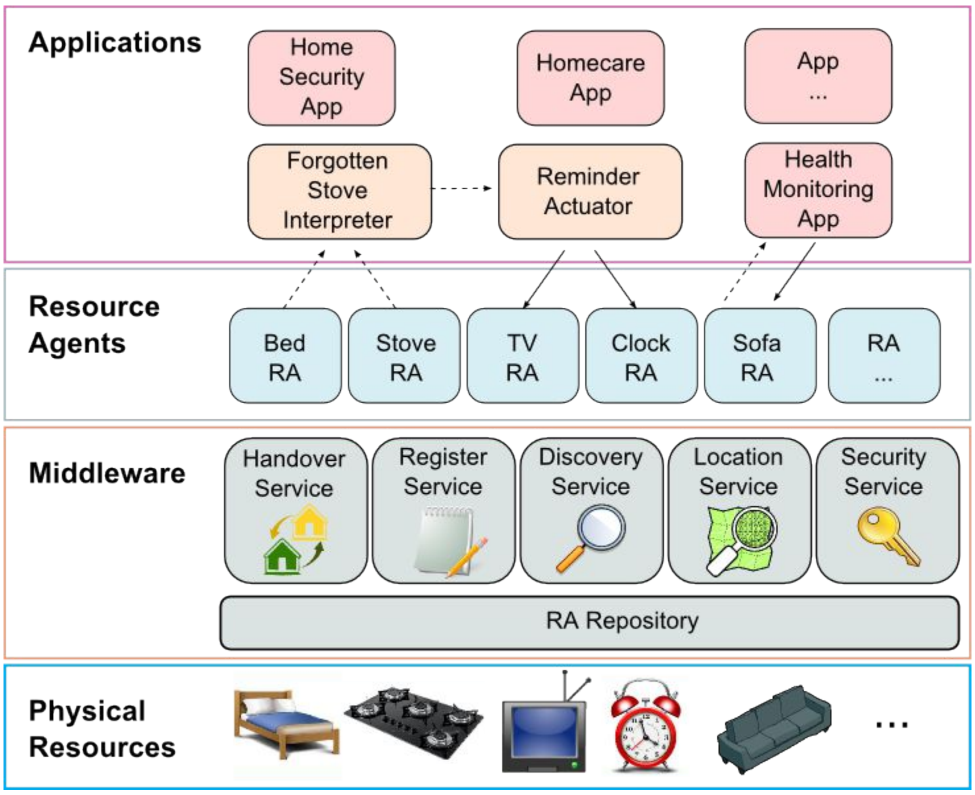


Figure 5.3: Architecture of the UbiTempo+ including the *Handover Service* in the middleware layer.

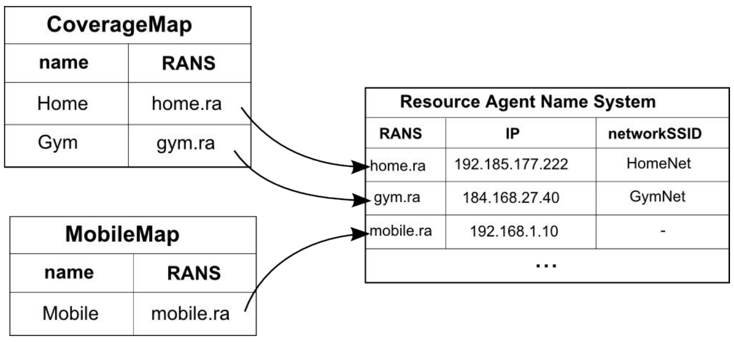
*UbiTempo* has a strong dependence on the *Discovery Service*because each AAL space must deploy its own Discovery Service. It does not represent a problem to implement the direct handover because the environment infrastructure is similar to the home. However, this characteristic of the UbiTempo can be a problem for implementing the indirect handover because the environment in which the resident stays is unknown and not covered by AAL Platforms. To solve this issue, during the process of the indirect handover, the Handover Service, which is described in Section 5.2.3, checks if there is an *Discovery Service*in the environment. If not, the mobile device starts an Discovery Service, transforming the mobile device into a small and mobile AAL space.

*UbiTempo* has a component named *Map* that represents the semantic division of an AAL Space. For example, a house usually is composed of a bedroom, a kitchen and a bathroom. All of these rooms can be represented as a semantic place within the home. The *Map* can be compared with the *Space Abstraction* component. In addition, the queries to the *Location Service* are resolved considering the database of the Map, which is composed of two main classes: Space and Place. While the former describes an AAL Space such as a house and a gym, the latter describes a subdivision of such spaces, which can be rooms such as a bedroom and a kitchen. The range of the Map just reaches the restricted area of the AAL Space, therefore this structure does not enough to represent service coverage area of the UbiTempo+. Because of that, we propose two new structures to complement the semantic division of the coverage area: *CoverageMap* and *MobileMap*. This approach is described in Section 5.2.1.

Although we do not extend the Security Management Service (SMS) to the Handover Service, the security of the handover process is an important issue. Our approach uses the SMS only for managing authentication and authorization for external entities that may request connection with the AAL Space. However, we plan to adapt the SMS for handling external requests and internal requests to external entities.

### 5.2.1 Data of the Coverage Area

In the *UbiTempo*, the *Map* structure represents the physical organization of an intelligent environment in a 2D space. Each semantic division of the environment (i.e., bedroom, living room and bathroom) can be represented by a geometric figure; in this case, these are a square, a circle or a mix of the two. This allows the `UbiTempo` to set the location of an agent using 2D coordinates [60]. Managing data of the *Map* is in charge of the *ResourceLocation* service, providing methods to update and query information on the space. In this context, we may suggest that the service coverage of the UbiTempo is represented in the *Map* structure.



CoverageMap

| name | RANS |
|---|---|
| Home | home.ra |
| Gym | gym.ra |

MobileMap

| name | RANS |
|---|---|
| Mobile | mobile.ra |

Resource Agent Name System

| RANS | IP | networkSSID |
|---|---|---|
| home.ra | 192.185.177.222 | HomeNet |
| gym.ra | 184.168.27.40 | GymNet |
| mobile.ra | 192.168.1.10 | - |
| ... | | |

Figure 5.4: An example of *CoverageMap* and *MobileMap* and their relation with the *RANS*

The level of detail employed by the *Map* is useful for representing a local and well-delimited space such as a home and an office. However, it seems unnecessary and even insufficient to represent the remote spaces covered by the *UbiTempo+*. To fill the gap, we propose creating a new structure to manage the data for remote spaces, designated as *CoverageMap*, and changing the name of the *Map* to *LocalMap*.

As illustrated in the Figure 5.4, the *CoverageMap* is a bi-dimensional data structure that stores the name and the *HandoverService* RANS of each remote space covered by our AAL Platform. Not included in the *CoverageMap*, the mobile spaces are stored in a structure similar to the *CoverageMap* called *MobileMap*. In this work, we limited the *MobileMap* to being a unique entrance to the mobile device used in the indirect handover protocol.

In our approach, the RANS has the additional attribute of storing the SSID of the space's wireless network. Additionally, *ResourceLocation* was modified to manage the data not only of the *LocalMap* but also of the *CoverageMap* and the *MobileMap*. The Figure 5.5 demonstrates the new role of the *Resource Location* including its relation with the *Handover Service*, which consumes the services by requesting data from the space data sets.

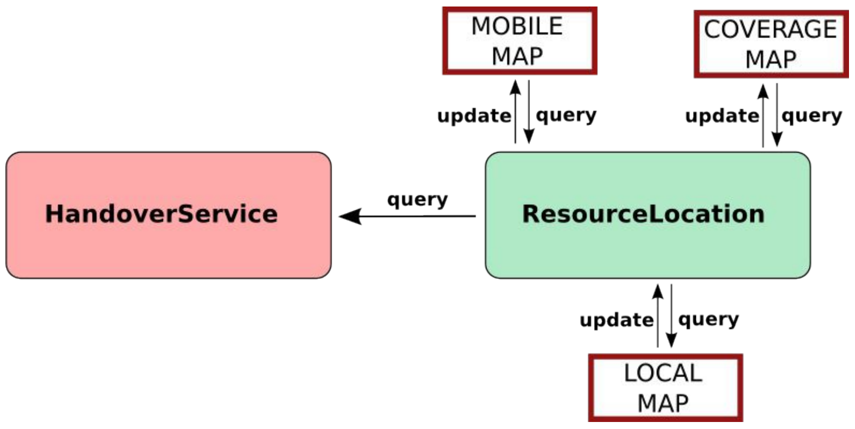


Figure 5.5: The role of the *ResourceLocation* and its relation with the *HandoverService*

### 5.2.2 Handover Agent

As previously mentioned, the central entity of the reference framework is the Resource Agent (RA). All entities that generate, consume or share context may be represented as an RA, such as a TV, a resident, a bed, and an application. However, as the reference framework focuses on restricted areas such as a house, a hospital, and a gym, a Resource Agent is not appropriate to represent the concept of the mobile entities required by an application that needs a handover. Thus, the central concept of including support for the handover in the reference framework architecture should be analyzed. To not drastically change the architecture, we propose a specialization of the Resource Agent: the Handover Agent.

As depicted in Figure 5.6, a Handover Agent is designed to provide the necessary information to the Handover Service performing the handover protocol. In addition,

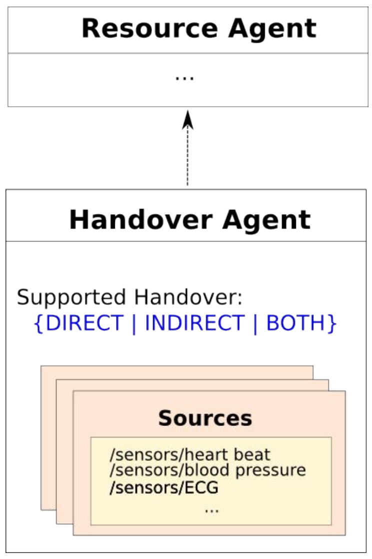


Figure 5.6: Handover Agent as a specialization of Resource Agent

a Handover Agent comprises all the attributes and methods of a Resource Agent and includes the following attribute:

- Supported Handover: The supported handover represents the type of handover that the agent is able to perform, which influences when the Handover Service will execute the handover protocol. For example, if an agent supports only the direct handover, the handover protocol is executed only when another AAL Space requests the handover action. In the indirect handover, the own AAL Space starts the handover protocol when the resident presence is no longer detected by the current AAL Space.

The reference framework also does not handle the changing sensors supported by an AAL Service because its domain does not require this type of feature. However, in our problem domain, the same AAL Service may handle different sensors depending on the AAL Space. For example, in Maria's scenario, the furniture measures her heart beat at home; outside, a smart bra takes charge of sensing her heart beat. To fill the gap, we include an attribute in the Handover Agent to represent the supported sensors as a list of Resource Agents, which are also sorted by importance. Then, an AAL Service based on the Handover Agent should handle the change in its supported sensors according to their availability. This is the reason that they should be sorted by importance.

### 5.2.3 Handover Service

Combining the components *Handover Handler* and *Synchronization Service*, we propose a new service called *Handover Service*, which is responsible for passing the control of the service provision from one environment to another. The *Handover Service* manages the migration of entities, including the resident, implementing a handover protocol for each type of handover, which are the direct handover and indirect handover.

As AAL is a user-centered technology, a Handover Service must work together with a resident presence system. Based on the context provided from such a presence system, the Handover Service may or may not start the handover protocol. The Handover Service only needs location data about the resident because it is the center of our domain. In addition, each AAL Space and the Mobile AAL Space have their own Handover Service, as is the case for the *Discovery Service.*

In general, the Handover Service transfers all the Handover Agents from the current environment to another. For example, in Maria' scenario, the health monitoring system may be implemented as a Handover Agent as well as Maria because the resident is also considered a Resource Agent. Additionally, when Maria leaves the house, the Handover Service transfers the health monitoring system (e.g., the current context and the model) from the house to the smart phone, which further represents the current environment.

During the handover protocol, the Handover Service also manages all the subscriptions from and to the Resource Agents, including the resident that is being transferred to the new environment. All the subscriptions have to be cleared from such Resource Agents but stored by the current environment, which does not mean that they must be rebuilt in the new environment.

Outside, the smart phone represents the environment because it comprises an Discovery Service, an Handover Service and an RA Repository. So, we can consider it a Mobile AAL Space because it creates a lightweight version of an AAL Space based on the framework extension around the user. As part of the Mobile AAL Space, the *Register Service* registers and de-registers Handover Agents and Resource Agents at the space.

## 5.3 Handover Protocols

Home is the central structure of the AAL Domain [33]. Therefore, the home plays an important role in our architecture because it centralizes the majority of handover requests, in particular from other AAL Spaces. In addition, only AAL Spaces are able to start a handover protocol because they can deploy resident presence detection, which is a prime requisite of the Handover Service. Furthermore, only the AAL Space that represents the home is able to create and delete Handover Agents, and then, the role of the other AAL Spaces is to execute them.

Unlike Wireless Networks, our handover approach cannot consider the ME responsible for initiating the handover protocol because we consider a trigger to start the handover protocol, which is the user presence in the current AAL Space. Additionally, the step to allocate resources seems unnecessary in our domain because we do not work with many mobile hosts and do not focus on load balance. In our domain, the handover protocol varies according to the handover type. In this section, we present the handover protocol for the Direct Handover and Indirect Handover, relating the actors presented in Section 2.4.2 with each domain.

### 5.3.1 Direct Handover

Direct handover happens when two AAL Spaces establish a connection for provisioning AAL services to the user, maintaining the continuity of the AAL service. In this case, the event that triggers the handover process is the user leaving the current AAL space, which will result in an event of entering into the new AAL space. The current AAL space has the most updated version of the handover agents, and thus, during the handover process, a step to update the agents in the new environment is considered, which also concerns synchronizing the contexts and models from the current AAL space with the new one.

In the context of Direct Handover, we relate the handover protocol actors as: (i) Migration Entity (ME): user, (ii) Domain0 (Dom0): house, (iii) Domain Representative0 (DomRep0): Handover Service of the house, (iv) Domain1 (Dom1): secondary AAL space, and (v) Domain Representative0 (DomRep0): Handover Service of the secondary AAL space. In the Figure 5.7, we illustrate the handover protocol for the direct handover with its actors, environments and the following steps: all steps, actors and environments. Once registered, the Handover Agents look for the Discovery Service; allowing them starts the interaction with the other agents at Dom1.

1. Dom1 detects the user presence.
2. DomRep1 identifies the last user space.

3. DomRep1 authenticates its identity and request authorization to DomRep0.
4. DomRep0 selects the Handover Agents labeled as the Direct Handover from the Dom0 RA Repository.
5. DomRep0 transfers the selected Handover Agents to DomRep1.
6. DomRep1 inserts or updates the Handover Agents in the Dom1 RA Repository.
7. Dom1 *Register Service* registers the transferred Handover Agents.
8. DomRep1 closes the DomRep0 connection.

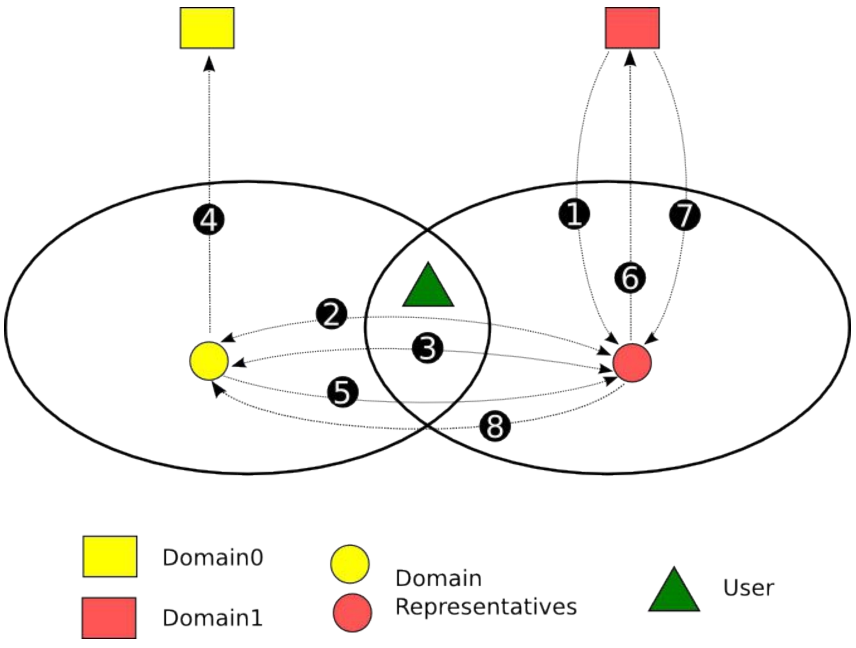


Figure 5.7: Protocol for direct handover considering two AAL Spaces

We discuss the handover protocol from the house perspective because it can be considered the main space of the AAL domain. However, the protocol of the direct handover can be executed between other AAL Spaces that are not the user's house. For example, the protocol may also run during the user migration from a smart gym to a smart office.

### 5.3.2 Indirect Handover

Indirect handover happens when an AAL Space establishes a connection with a mobile device for providing services to the user in a blind area, i.e., in an area not covered by an AAL Platform. In the context of the Indirect Handover, we relate the handover actors as: (i) Migration Entity (ME): resident, (ii) Domain0 (Dom0): house, (iii) Domain Representative0 (DomRep0): *Handover Service* of the house, (iv) Domain1 (Dom1): mobile device, (v) Domain Representative1 (DomRep1): Handover Service of the mobile device.

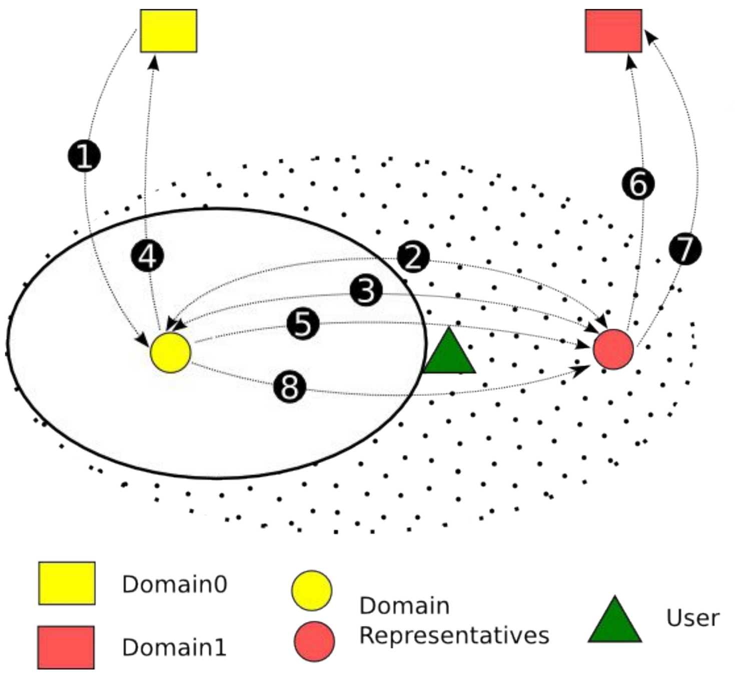


Figure 5.8: Protocol for indirect handover considering the user leaving an AAL Space (ellipses)

Unlike the Direct Handover, two events can trigger the handover process in this context: (i) the user leaving an AAL Space and (ii) the user entering an AAL Space. We treat both events individually because the agent synchronization is a critical step in the indirect handover. For the resident leaving event, the handover protocol consists of the following steps depicted in the Figure: 5.8:

1. Dom0 detects the resident leaving.
2. DomRep0 identifies the mobile space.
3. DomRep0 authenticates its identity and request authorization to DomRep1.
4. DomRep0 selects the Handover Agents labeled as Indirect Handover from the Dom0 RA Repository.
5. DomRep0 transfers the selected Handover Agents to DomRep1.
6. DomRep1 inserts or updates the Handover Agents in the Dom1 RA Repository.
7. Dom1 *Register Service* registers the received Handover Agents,

8. DomRep0 closes the DomRep1 connection.

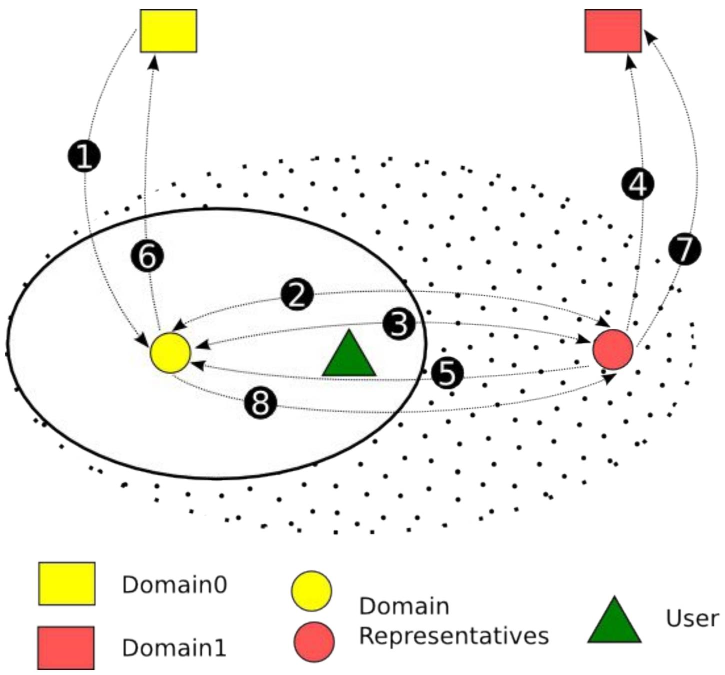


Figure 5.9: Protocol for indirect handover considering the user entering/backing to an AAL Space (ellipses)

For the resident entering event, which may represent the resident returning home, the handover protocol consists in the following steps illustrated in the Figure 5.9:

1. Dom0 detects the resident presence.
2. DomRep0 identifies the mobile space.
3. DomRep0 authenticates its identity and request authorization to DomRep1 .
4. DomRep1 selects all the Handover Agents from the Dom1 RA Repository.
5. DomRep1 transfers the Handover Agents to DomRep0.
6. DomRep0 updates the Handover Agents in the Dom0 RA Repository.
7. Dom1 *Register Service* de-registers the Handover Agents.
8. DomRep0 closes the DomRep1 connection.

## 5.4 Discussion

In this chapter, we discussed our framework, the UbiTempo+, indicating the components that should be deployed in the reference framework architecture to support both the direct and indirect handover. Additionally, we proposed architectural changes in the UbiTempo (i.e., the reference framework) including the extension of the Resource Agent to the Handover Agent and the deployment of a component to handle handover protocols, which is called Handover Service. Protocols that are inspired by the basic handover protocols discussed in Section 2.4.2 were presented and discussed.

To demonstrate the feasibility and evaluate the UbiTempo+, in the next chapter, we implement a prototype using the direct and indirect handover for solving some of the problems presented in Maria's scenario (Section 2.1).

# Chapter 6

# Implementation

In this chapter we describe an instance of the framework *UbiTempo+*, the *SmartAndroid+*. We discuss the components of the *SmartAndroid+* involved in the support of the handover process. We also present the Heart Monitoring System, a prototype AAL Service implemented as a *Handover Agent*, which measures the heartbeat of the user and can be migrated through AAL Spaces.

## 6.1 Prototype Overview

Maria's story (described in the Section 2.1) describes two AAL Spaces: a home and a gym. Additionally, an open space is cited, in particular, when Maria goes to the supermarket or goes for a walk in the park. These three environments are described in Section 2.1. Our prototype focuses on these three spaces, designing and implementing each one.

Regarding AAL Services, three are shown in the following scenarios: (i) a continuous health monitoring, (ii) a resident presence detection, and (iii) an emergency call. The first one represents the service that has to be available for Maria regardless of her location. We therefore modeled it as a Handover Resource Agent. The second system is strongly linked with the *Handover Service* of the AAL Space; for that reason, it is considered a Resource Agent.

For the sake of simplicity, our prototype only implements the AAL Service of continuous health monitoring. Based on the general home care systems, such as the system proposed by Coppeti [25], our AAL Service measures the heart beat signals provided by a sensor or a set of sensors.

## 6.2 General Features

*SmartAndroid+* was implemented by extending the *SmartAndroid* architecture and using the ADT (Android Development Toolkit) [40]. In this case, our AAL Platform can run on top of the majority of the devices with the Android OS, for example, smart phones, tablets, laptops, and mini PCs, as well as sensors and actuators. It can run in two modes: stationary and mobile. In the Table 6.1, we compare the modes in regards to some features available in each one. It is worth mentioning that the Management Services cited in the table comprises the *Resource Location*, the *Resource Register* and the *Resource Discovery.* In general, the stationary mode runs on a typical AAL Space (i.e., smart home and smart office), which usually does not have limitations in battery consumption or availability of resources. On the other hand, the mobile mode was designed to run on mobile devices (i.e., smart phones), for which we have to consider the shortage of resources or power energy.

Table 6.1: Comparison between Stationary Mode and Mobile Mode in regards to some *SmartAndroid+* features.

| | **Stationary** | **Mobile** |
|---|---|---|
| Management Services | X | X |
| Agents set as Indirect Handover | X | X |
| Agents set as Direct Handover | X | |
| Wi-Fi Scanning | X | |
| Identify Last User Space | X | |

As previously discussed in Chapter 5, the *SmartAndroid+* is an extension of the SmartAndroid proposed by the LabTempo [59, 12, 34]. Thus, in this section, we focus our discussions on the components that compose the handover support of our architecture, in particular the *Handover Service*, which is in charge of managing the handover process. Additionally, we describe the components that provide other crucial services to enable the mobility of resource agents: the *SpaceGateway* and the *Resource Location*, including the *Map* structure. *Resource Location* was proposed by the SmartAndroid but modified by us to manage and provide the necessary data to the *HandoverService.*

### 6.2.1 Communication

*SmartAndroid* already features mechanisms of communication between agents in the intelligent environment: the direct with RPC (Remote Procedure Call) and the indirect with Publish/Subscribe. For the *Space Gateway* in the *SmartAndroid+* we employ a server/-

client approach, implementing a multi-threaded TCP/IP server to handle communication with remote spaces and external entities. This allows the *Space Gateway* to act as a continuous service listening for new remote connection requests. The *HandoverService* can process the received messages by implementing an abstract method of the *Space Gateway* class signed as *processmessage(String message).*

As the *SmartAndroid+* was developed over the Google Android Platform [38], the resources of the devices can be accessed by the mechanisms provided by this platform. The *Space Gateway* uses the *WifiManager* to control the Wi-Fi component of the device which the *SmartAndroid+* is running. In particular, the gateway can enable/disable the Wi-Fi scanning and obtain the result of the Wi-Fi scan in the form of a list of *ScanResult* [42] objects. A *ScanResult* provides data about a Wi-Fi network such as the level of the signal, the frequency over the channel, the SSID and the BSSID.

During the start of the gateway, it checks if the *SmartAndroid+* is running in stationary or mobile mode, enabling the Wi-Fi to scan only in stationary modes to save battery power. The *Space Gateway* also features a method signed as *checkIfNetworkInCoverage(String networkSSID)* that looks for the SSID of a given Wi-Fi network in the results of the last Wi-Fi scan. It is further used by the *Handover Service* during the algorithm to discover the last space where the user was.

## 6.3 Handover Service

As discussed in the Section 5.2.3, the *Handover Service* acts as a *Handover Handler* and a *Synchronization Service* in the AAL Platform. Its most important role consists in providing migration of handover agents to another AAL Space (stationary or mobile) by running the handover protocols presented in the Section 5.3. In this section, we explain how the *Handover Service* runs the handover protocols, describing all mechanisms developed to support this. First, we describe the interface between *Handover Service* and the presence detection system of the space and the handover status cycle of a stationary space. Next, we describe the type of messages used by the handover protocols. Finally, we explore the critical methods of the handover protocols: *updateAgents()*, which is shared by all protocols, and *indentifyLastUserSpace()*, which is performed during the direct handover protocol.

### 6.3.1 Service life cycle

The migration of agents seems necessary only when the service coverage of the AAL Platform reaches at least one stationary space or one mobile space. Thus, the coverage area consists of a set of AAL Spaces linked by the intersection of Wi-Fi networks, in particular if those spaces run in stationary mode. For the correct performance of the *SmartAndroid+*, each stationary space should control its handover status and make it available to other spaces. For that purpose, the Handover Service is managed as a *finite-state machine* [14]. The Figure 6.1 illustrates the life cycle of the Handover Service, also indicating the triggers from a presence detection system.

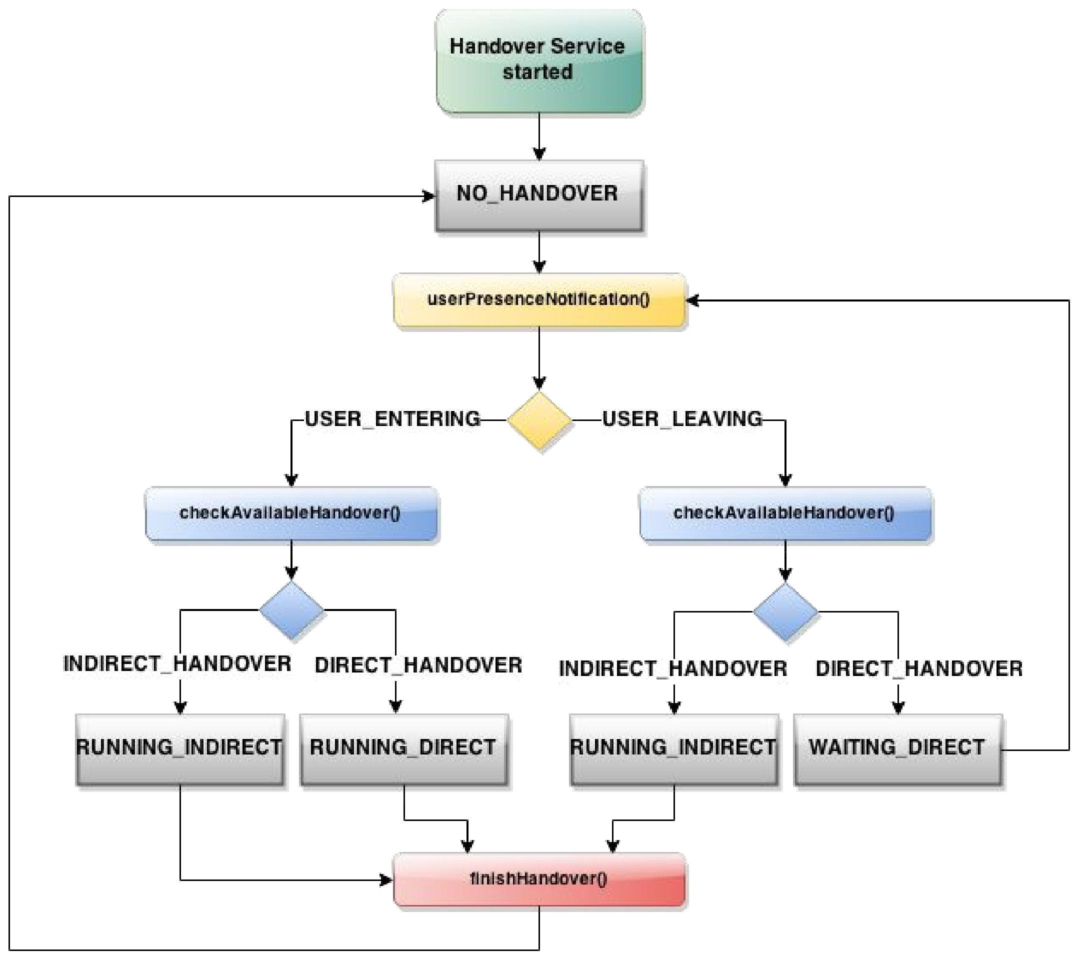


Figure 6.1: The life cycle of the *Handover Service* in a stationary space

When the *Handover Service* starts, it sets the status as `NO_HANDOVER`. This status can be modified only if the presence detection system notifies a user presence event to the *Handover Service*. To abstract the presence detection system implementation, we propose two types of events of the presence detection system that the *Handover Service* can process: `USER_LEAVING` and `USER_ENTERING`. Thus, any presence detection system can notify an event because it notifies according to the pattern supported by the *Handover Service*.

Once notified by the presence detection system, the *Handover Service* calls the method *userPresenceNotification()* to process the received event. If `USER_LEAVING`, it checks the protocols supported by the handover agents in the *Repository*. If it finds an agent set as a direct handover, the service sets the status as `RUNNING_DIRECT` and starts the direct handover protocol. In the case of agents set as an indirect handover, the services set the status as `RUNNING_INDIRECT` and start the indirect handover protocol for the leaving event. In both cases, after running the protocols, the service calls the method *finishHandover()* and closes the cycle by setting the status as `NO_HANDOVER`.

With regard the *userNotification()* step, if the result is `USER_ENTERING`, the *Handover Service* also checks the protocols supported by the agents. If the result is the direct handover, the service sets the status as `WAITING_DIRECT`, which means that the space may be considered the last space occupied by the user in case there is an additional space request to start the direct handover protocol. If indirect handover is the result, the Handover Service sets the status as `RUNNING_INDIRECT` and starts the indirect handover protocol. After finishing the protocol, the *finishHandover()* method is also called and the cycle closes by setting the status as `NO_HANDOVER`. It is worth mentioning that just one more notification from the presence detection system may change the `WAITING_DIRECT`.

### 6.3.2 Protocol messages

The handover protocols for the direct and indirect handover, described in the Section 5.3, have similar steps, but may be in different orders. Taking advantage of this, the *SmartAndroid+* implements a set of message patterns to express these steps of the handover protocols. Employing such a pattern, the *Handover Service* can request, respond and report some process to a remote space (stationary or mobile) using the *Space Gateway*. The Table 6.2 describes all message patterns supported by a *Handover Service*.

In the Section 6.3.2, we demonstrate the relation of the messages with the handover protocol in the implementation level by scenarios designed to evaluate the direct and indirect handover.

Table 6.2: Description of the messages used during the handover protocols

| Message | Description |
|---|---|
| `STATUS_REQUEST` | Request the last handover status of a remote space. |
| `STATUS_RESPONSE` | Respond the status request. The message consists of the handover status and the time (in milliseconds) when the status changed. |
| `AUTH_REQUEST` | Request to the remote space for authorization to start a given handover protocol. |
| `AUTH_RESPONSE` | Respond to the authorization request with the type of the given handover pattern and if the space has permission or not. |
| `START_AGENTS_TRANSFER` | Inform the remote space that agents will be transferred. |
| `UPDATE_AGENT` | Inform the remote space that an agent will be transferred and request to add it to the update queue. |
| `FINISH_AGENTS_TRANSFER` | Inform the remote space that all agents were transferred. |
| `AGENT_UPDATED` | Report the last space, where the agent comes from, that it was updated in the new space. |
| `FINISH_AGENTS_UPDATE` | Report the last space that all agents were updated. |
| `FINISH_HANDOVER` | Request the last space to finish the handover protocol. |

### 6.3.3 Identifying the Last User Space

One of the major contributions of our approach is providing user migration through a service coverage area without employing a mobile device. In other words, the spaces take care of running the handover protocol to transfer the service provisioning from the last space occupied by the user to another space. The user does not carry with him any data about the spaces he occupied nor the context of the services provisioned to him. Bear in mind that the first step of both the direct and indirect protocol is to establish a connection with the space where the user comes from. If the user does not have this information, the problem is to determine where the user was.

To determine the last user space to direct the handover, we propose a 2-step solution that in general requests the handover status of all spaces reachable by the current space and that further selects the space with the most recent status time. In particular, when the direct handover protocol starts, the *Handover Service* looks for the spaces with the handover status set as `WAITING_DIRECT`, which means the user was there, and this space is able to perform a direct handover. Then, the first step, which is described in the Algorithm 1, consists of obtaining these available spaces.

```
input  : -
output: A list of available spaces
1  $coverageMap \leftarrow ResourceLocation.getCoverageMap()$;
2  for $space$ : $coverageMap$ do
3      if $SpaceGateway.checkIfNetworkInCoverage(space.networkSSID)$ then
4          $spacesToEnquire(space)$;
5      end
6  end
7  for $space$ : $spacesToEnquire$ do
8      $requestStatus(space)$
9  end
10 $statusResponses \leftarrow receiveStatusResponses()$;
11 for $status$ : $statusResponses$ do
12     if $status.handoverStatus = WAITING\_DIRECT$ then
13         $availableSpaces.add(status.spaceName, status.statusTime$;
14     end
15 end
16 return $availableSpaces$
```

**Algorithm 1:** Get available spaces

The second step - presented in the Algorithm 2 - receive as a parameter a handover pattern. If the handover pattern matches `DIRECT`, the algorithm obtains the available spaces as a list, sorts them by the status time and returns the space name of the first item of the list. To select the most recently status time guarantee that the space was the last one where the user was. If the handover pattern is `INDIRECT`, as our approach supports only one mobile space, the *Handover Service* checks if the mobile space is reachable.

```
input  : Handover pattern
output: Name of the last user space
1  if $handoverPattern = DIRECT$ then
2      $availableSpaces \leftarrow getAvailableSpaces()$;
3      $inverseorder(availableSpaces, 2)$;
4      return $availableSpaces.first().spaceName$
5  end
6  if $handoverPattern = INDIRECT$ then
7      if $SpaceGateway.agentIsReachable(ResourceLocation.getMobileSpace())$
       then
8          return $ResourceLocation.getMobileSpace())$
9      end
10     return $null$
11 end
```

**Algorithm 2:** Indentify the last user space

### 6.3.4 Updating agents

Updating the context of the handover agents is a critical step of the handover protocols. In the Algorithm3, we describe the routine to update the handover agents transferred from the last user space. During the transfer step, the data of the agents (fields and methods) is serialized using the JSON pattern [52] and transferred by TCP messages following the pattern `UPDATE_AGENT` presented in the Section 6.3.2. In general, the update algorithm compares the list of agent data with the list of handover agents available in the *Repository*. When an ID of the agent data matches the ID of a handover agent of the environment, the fields of the given handover agent are updated with the agent data.

```
input  : A list of agents to update
output: -
$agents \leftarrow ResourceRepository.getHandoverAgents()$;
for $agent : agents$ do
    for $agentData : agentsToUpdate$ do
        if $agentData.id = agent.id$ then
            for $field : agent.class.fields$ do
                for $fieldData : agentData.field$ do
                    $field.setValue(fieldData.value)$;
                end
            end
            $agentsToRegister(agent)$;
            $informAgentUpdated(agent.id)$;
        end
        else
            $informAgentUnavailable(agent.id)$;
        end
    end
    $informAgentUpdateFinish()$;
    $agentsToUpdate().clear()$;
    $registerAgents(agentsToRegister)$;
end
```

**Algorithm 3:** Update the handover agents

## 6.4 Heart Monitoring System

After describing the *SmartAndroid+* architecture in the past sections, in this section we focus on implementing an application with handover support. In our reference scenario (Section 2.1), Maria's house has a heart rate monitoring system that can follow her wherever she goes. In other words, she is served by a continuous monitoring system able to analyze her vital signs and to call help in case of emergency. In this context, we developed a heart monitoring system that measures the user heartbeat in conjunction with a wearable sensor, in this case a smart bra. As we are concerned with evaluating the features related with the handover process, our application prototype does not implement the vital signal analysis or the emergency trigger.

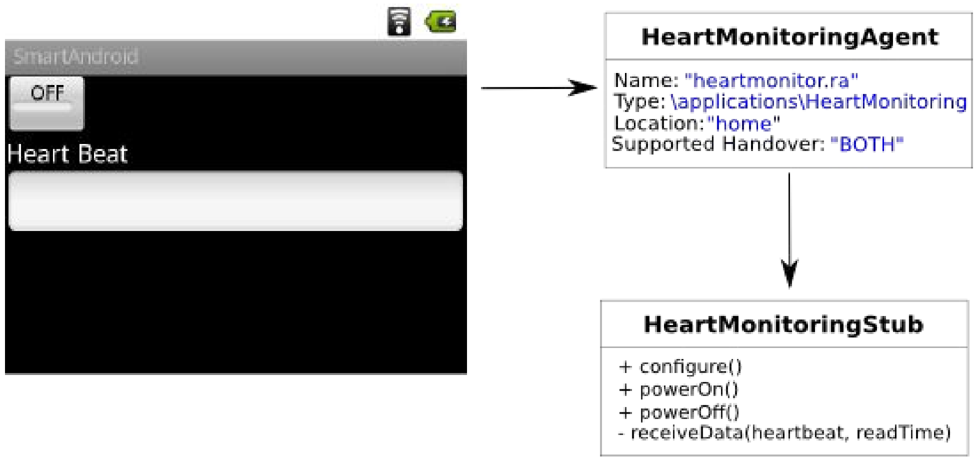


Figure 6.2: Structure of the Handover Monitoring System, which comprises a Handover Agent, a Handover Stub and Application View

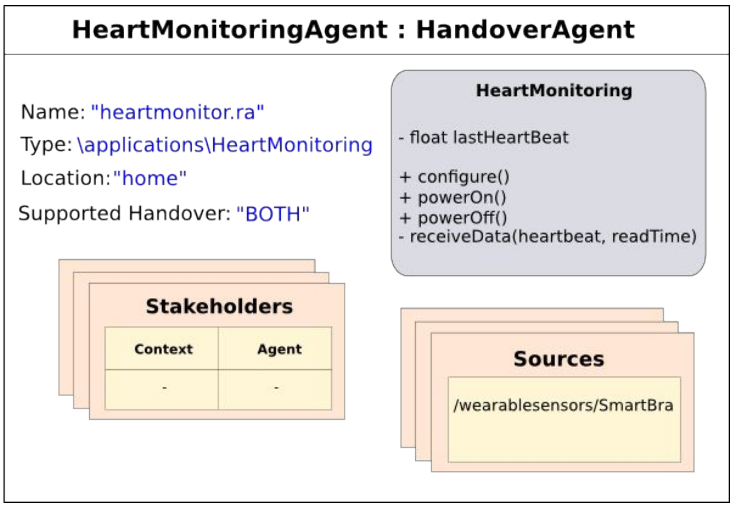

HeartMonitoringAgent : HandoverAgent
Name: "heartmonitor.ra"
Type: \applications\HeartMonitoring
Location:"home"
Supported Handover: "BOTH"
HeartMonitoring
- float lastHeartBeat
+ configure()
+ powerOn()
+ powerOff()
- receiveData(heartbeat, readTime)
Stakeholders
Context
Agent
-
-
Sources
/wearablesensors/SmartBra

Figure 6.3: Agent of the system as a Handover Agent instance

*SmartAndroid* already provides a platform to prototype and test ubiquitous systems [12]. This platform also provides the mechanisms to simulate sensors and applications in the space using the Android Platform. Following the *SmartAndroid* framework, the heart monitoring system, depicted in the Figure 6.2, consists of three components: (i) a handover agent, i.e., the core of the system; (ii) a handover agent stub to establish a communication interface with the other agents in the space; and (iii) a graphical user interface formed by an Android Layout [41] and an Android Activity [39].

The components (ii) and (iii) are important, but we are focused on the component (i) because it comprises the handover agent of the system, the concept proposed by our framework, the *UbiTempo+*. In the Figure 6.3, we illustrate the instance of this system agent. The attribute *handoverSuppport* is set as `BOTH`, which means the system supports not only the direct handover pattern but also the indirect handover pattern. An additional important attribute is the sources for which the system can acquire data. In that case, the monitor can receive data from the sources of the type *br.uff.tempo.werableSensors.SmartBra*. After being registered, the heart monitoring system looks for sources of the given type in the current space. Once one has been found, the system subscribes and configures the source and further starts to receive data from it.

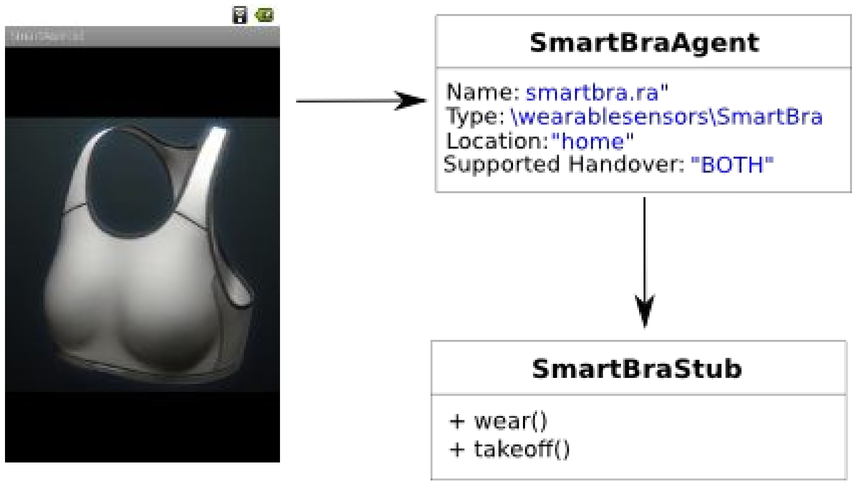


Figure 6.4: Structure of the Smart Bra Simulator, which comprises a Handover Agent, a Handover Stub and Application View

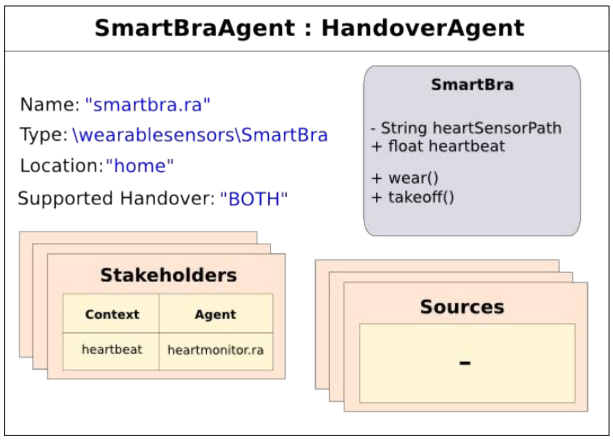


Figure 6.5: Agent of the simulator as a Handover Agent instance

As to the data source, we implemented a smart bra simulator, which is composed of the same structure of the heart monitoring system. The Figure 6.4 illustrates the simulator and their components and, in the Figure 6.5, the handover agent instance of the SmartBra is depicted. To simulate as much as possible the measurement of a user's heart, we use a data set of heart beats provided by the *MIT-BIH Database Distribution* [1] to create a data stream, thus simulating a simple heart rate sensor. In each interval of 500 ms, the agent reads a heartbeat from the data set and sends these data to all subscribed agents. This data streaming is only available when the user is wearing the smart bra. For this purpose, the simulator's activity provides an interface to enable us to set if the user is wearing the smart bra.

## 6.5 Discussion

In this chapter, we described our prototype implementations of the *SmartAndroid+*, and the heart monitoring system. The *SmartAndroid+* is an instance of the *UbiTempo+*, an AAL Platform that provides infrastructure to AAL Services running on top of it. Abstracting the features already provided by the *UbiTempo*, we focused our efforts on explaining the middleware services that feature the handover support in our platform: the *Space Abstraction*, the *Handover Service*, the *Space Gateway* and the *Location Service.* The heart monitoring system is an example of an AAL Continuous Service supported by our AAL Platform.

In regards to devices with limited resources and battery consumption, the *SmartAndroid+* can also be run in the lightweight mode called Mobile Mode. Our platform implementation currently supports only devices with Wi-Fi modules. We believe, however, that in the future, support to other types of wireless networks, such as Bluetooth, may be integrated into the *SmartAndroid+*.

[1]http://ecg.mit.edu/time-series/

# Chapter 7

# Evaluation

In this chapter, we describe how our implementation of the *SmartAndroid+* was tested with respect to its functional behavior and its performance in regards to handover protocols. Next, we discuss the features of our approach, relating them with some scenarios. Finally, we compare the *SmartAndroid+* with the relared platforms discussed in the Related Works (Chapter 3).

## 7.1 Testbed

To test the functioning and evaluate the competence questions for *SmartAndroid+*,we ran different batches of tests. Scenarios were extracted from Maria's story (Section 2.1) to guide the execution of the user migration by direct and indirect handover protocols. To run the scenarios we prepare two infrastructures, one to simulate two stationary AAL Spaces and their entities and an additional scenario to simulate a stationary AAL Space and a mobile AAL Space with their entities. In Figure 7.1 and in Figure 7.2we illustrate both infrastructures. Indeed, above the agents described in the figures appear the IP Address and RANS of each one. Although each Management Service has its own RANS, we display only the RANS of the *Handover Service* because it plays the main role in the handover process.

The infrastructure to the direct handover comprises: (i) two IEEE 802.11g WLANs configured as the sub-networks of another WLAN; (ii) two Samsung P7500 Galaxy Tab 10.1 tablets running Android OS 3.2; and (iii) four Motorola Moto G XT1032 smart phones running Android OS 4.3. The tablets run the Management Services, in particular the *Handover Service*, of the AAL Platform of each AAL Space. Additionally, in each AAL Space, one smart phone runs the Smart Bra Simulator and an additional smart

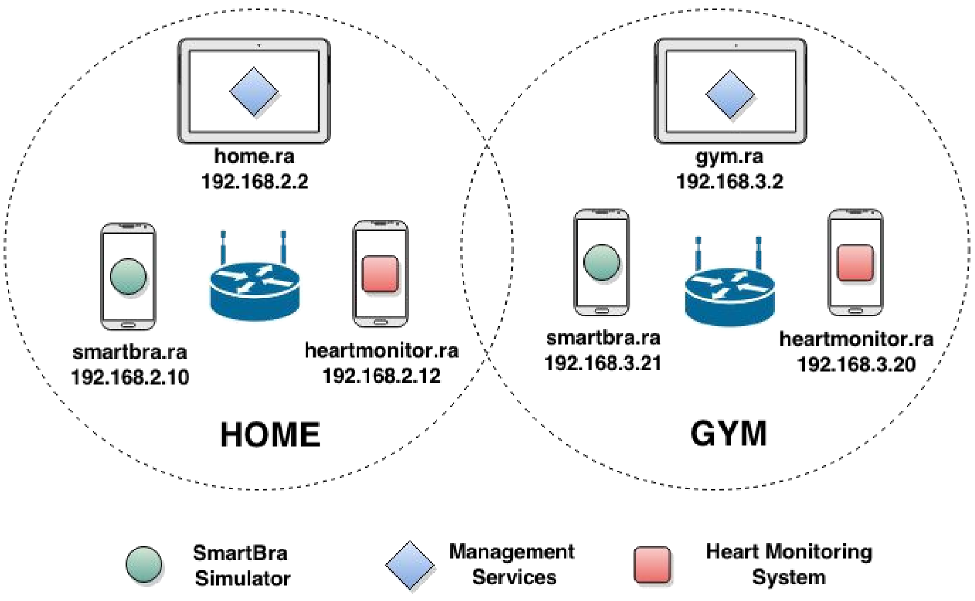


Figure 7.1: Infrastructure to run the Direct Handover scenarios.

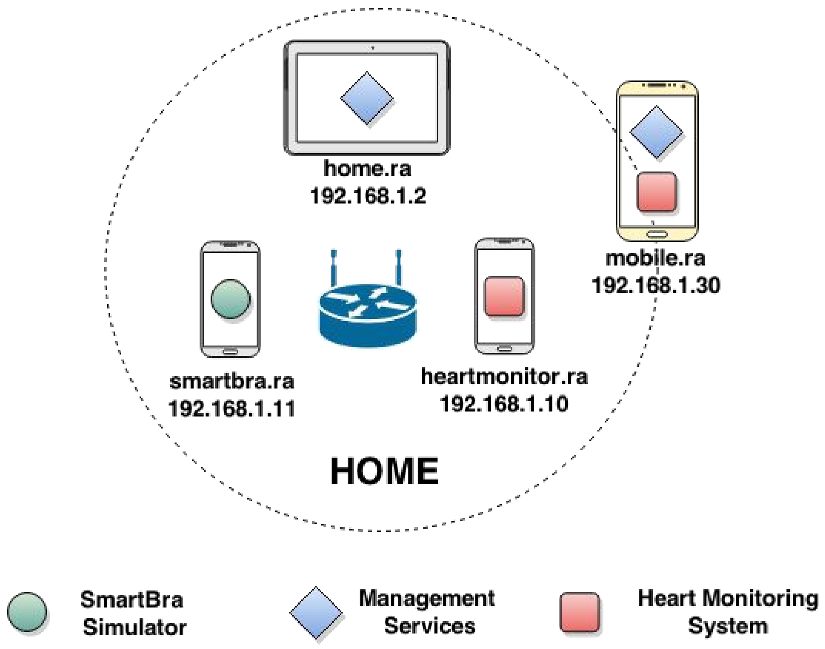


Figure 7.2: Infrastructure to run the Indirect Handover scenarios.

phone runs the Heart Monitoring System.

To run the scenarios of the indirect handover, the infrastructure of the direct scenarios can be reused, simplifying it by removing one AAL Space and configuring the ambient with only one Wi-Fi network. We also reuse one smart phone of the removed AAL Space to run our AAL Platform in Mobile Mode. Thus, we have all entities of the AAL Space Home running at the same time as the direct handover infrastructure with one more entity as a mobile space.

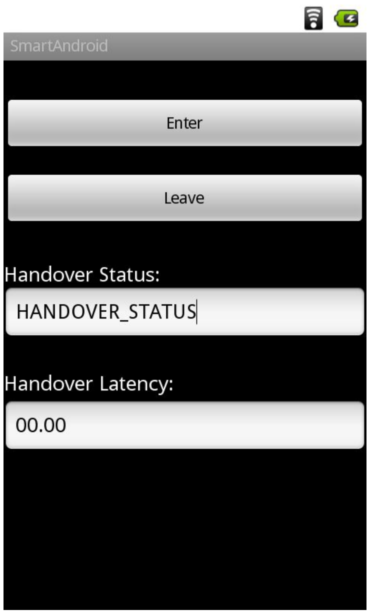


Figure 7.3: GUI of the Handover Test application.

## 7.2 Functional Test

A set of tests was performed to check if our framework met the functional components of the reference architecture to manage user migration through AAL Spaces, as enumerated in Section 2.1. As this operation involved the simulation of stationary and mobile AAL Spaces performing the handover process and the context change occurring within different sequences of events, we implemented the Handover Test, with a simple graphical user to easily trigger presence detection events and interpret the results.

The interface, displayed in Figure 7.3, contains buttons that allowed trigger presence detection events (`USER_ENTERING` and `USER_LEAVING`) and two text fields, one to indicate the current handover status of the space and an additional field to present the handover latency of the last handover process run. The test application measures the time because the presence detection triggers until finishing a handover protocol. We consider this time to be the handover latency. Depending on the test scenario, both AAL Spaces must run the *Handover Test* following a specific sequence of steps, but only the application that started the handover protocol is able to measure the handover latency.

## 7.3 Competence Question

We evaluates the *UniTempo+* by formulating competence questions related to our approach. Generally used to design and to evaluate ontologies, competence questions were proposed by Gruninger and Fox [43] and consist in representing the requirements of a domain problem in the form of questions that a proposed solution must be able to answer. In the context of the AAL Domain, we formulate competency questions about requirements of user mobility presented in the Ambient Assisted Living manifest [74].

For the purpose of demonstrating the conformity of our approach by answering these questions, we deployed the prototypes described in the Chapter 6 in accordance with the following scenarios extracted from Maria's story (Section 7.4): (i) a direct handover triggered by a user entering event (Section 7.4.1); (ii) an indirect handover triggered by a user leaving event (Section 7.5.1), and (iii) indirect handover triggered by user entering event (Section 7.5.1).

Our competency questions focus on evaluating the features of our framework regarding the resources supported and the migration of agents. The following two groups of questions were created:

1. Questions related to the migration of agents
   (a) Can the user leave an AAL Space, enter a blind area and keep being provisioned by any service designed as a Handover Agent?
   (b) Can the user enter an AAL Space from a blind area and keep being provisioned by any service designed as a Handover Agent?
   (c) Can the user move from an AAL Space to another space in the coverage area and keep being provisioned by any service designed as a Handover Agent?
2. Questions related to the support of resources
   (a) Can the AAL Platform handle handover requests from other AAL Platforms and further run the appropriated handover protocol?
   (b) Can the AAL Platform select and transfer any handover agent according to the handover pattern required in the start?
   (c) Can the AAL Platform update the handover agents in the current AAL Space using the agent data transferred from the last user AAL Space?
   (d) Can any system designed as a Handover Agent automatically set a data source in a new AAL Space?

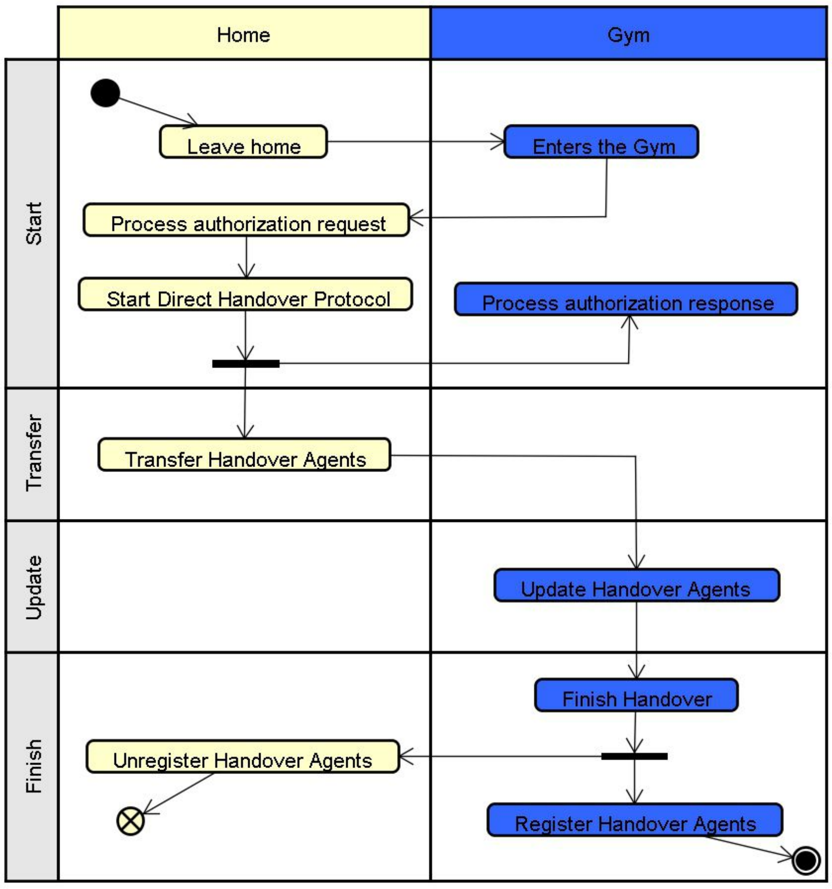


Figure 7.4: Overview of the main events of Scenario 1.

## 7.4 Direct Handover

In this section, we evaluate our framework in regards to the direct handover, i.e., a user migrating from an AAL Space to another space in the service covered area. The main goal of the evaluation is to demonstrate that the heart monitoring system can be continuously provisioned to the user in both AAL Spaces, thus answering **Question 1.c**. To run the application, we create a scenario based on Maria's story (Section 2.1). The scenario also provides features to answer **Question 2.a**, **Question 2.b**, **Question 2.c** and **Question 2.d**. For this purpose, we divide the run of the scenario into four phases: Starting the Handover (Section 7.4.2), Transferring the Agents (Section 7.4.3), Updating the Agents (Section 7.4.4) and Finishing the Handover (Section 7.4.5).

### 7.4.1 Scenario 1: Going to the Gym

Because it was detected that Maria suffers from arrhythmia, the doctor suggested monitoring her heart continuously. Thus, Maria is always wearing a smart bra that measures her heart rate and transfers it through a Wi-Fi network. Additionally, a continuous heart monitoring system was installed in her smart home. The system acquires data from her smart bra and presents the heartbeats on a friendly GUI.

Three times a week, Maria attends a Senior Gym, located in front of her house, to do regular aerobic exercises. Equipped with a smart environment system, the gym provides some services for their attendees such as vital sign monitoring. During the enrollment, Maria received a client id card, having been required to provide information to register the gym in the service coverage of her smart home. Once Maria adds the gym in the covered area, both spaces are able to provide the heart monitoring service to her. Despite the fact that Maria is still wearing the smart bra, at the gym, she can take advantage of an environment equipped with ubiquitous systems. Thus, at the gym, Maria does not have to be concerned about the continuous monitoring prescribed by the doctor.

In the Figure 7.4 we display an overview of the scenario in the form of an Activity Diagram. Each activity represents a main event of the scenario, disposed in the four phases as described previously. Additionally, the lanes represent the AAL Spaces that involved part of the scenario: Home (yellow lane) and Gym (blue lane).

### 7.4.2 Starting the Handover

As previously discussed, the direct handover is applicable if the user moves through the AAL Spaces in the covered area. In addition, the cycle starts when the user leaves his house for the first time, thereby setting the handover status of this AAL Space as `WAITING_DIRECT`. This is the reason that we started the scenario with the activity **Leave Home**. Using a sequence diagram, we illustrate in the Figure 7.5 how the **Home** handles Maria leaving the house. First, the *PresenceDetection* notifies the presence event `USER_LEAVING` to the *HandoverService*, which further queries the handover pattern supported by the agents in *Repository*, which next answers `BOTH`. Next, the *HandoverService* sets its status as `WAITING_DIRECT`, thus becoming available to receive direct handover requests.

The next step of the scenario is Maria entering the Gym, represented by the activity **Enters the Gym**. As depicted in the sequence diagram of the Figure 7.6, when she enters the AAL Space, the *PresenceDetection* notifies the presence event `USER_LEAVING` to the *HandoverService*. Next, it requests from the *Repository* the handover pattern supported by the agents. Once the answer is `DIRECT`, the *HandoverServices* calls the routine to identify the last space occupied by the user, as described in Section 6.3.3, resulting in

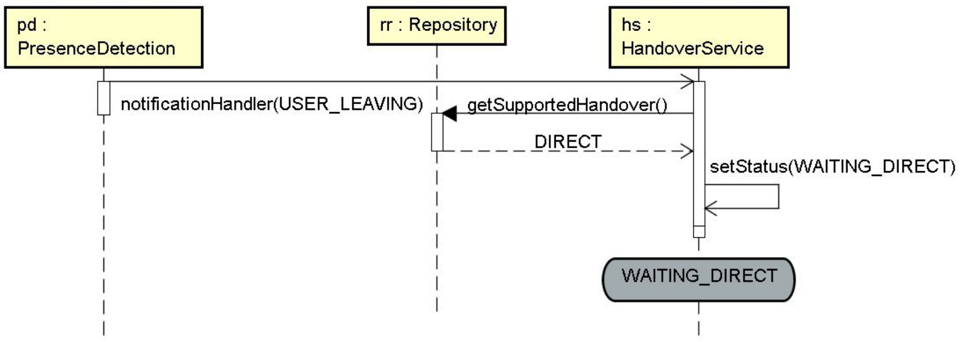


Figure 7.5: Sequence Diagram that demonstrates the interaction between the Management Services during the main event **Leave Home**.

"Home". Next, the *HandoverService* obtains from the *LocationService* the Home's IP address. With this information, the *HandoverService* sends by the *SpaceGateway* an authorization request message to the **Home** as follows: `''AUTH_REQUEST;DIRECT''`.

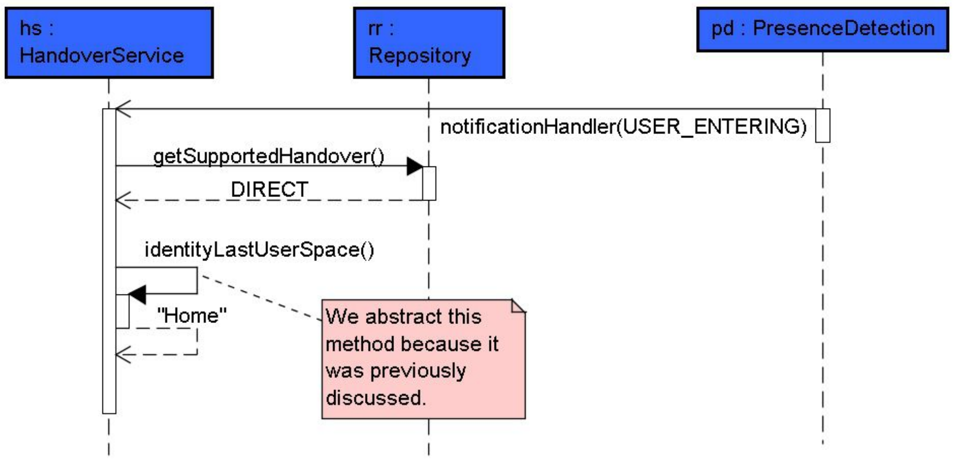
After the authorization request from the **Gym**, there are the main activities: **Process authorization request** and **Start direct handover protocol**. The Figure 7.7 and the Figure 7.8 illustrate the steps for both of them. First, the *SpaceGateway* receives the authorization request message and forwards it to the *HandoverService* to process it. Subsequently, the *HandoverService* checks the authorization by validating from the *LocationService* if the **Gym** exists in the *CoverageMap*. As the return was positive, the *HandoverService* sends an authorization response (`''AUTH_RESPONSE;allowed''`) and starts the direct handover protocol with the **Gym**.

In the last activity of that phase, which is the *Process authorization response*, the **Gym** receives the authorization response (`''AUTH_RESPONSE;allowed''`) from the **Home** by the *SpaceGateway*. Further, it calls the *HandoverServices* to process the received message. As **Gym** has authorization to start the direct handover protocol, the *HandoverService* sets its status to `RUNNING_DIRECT`.

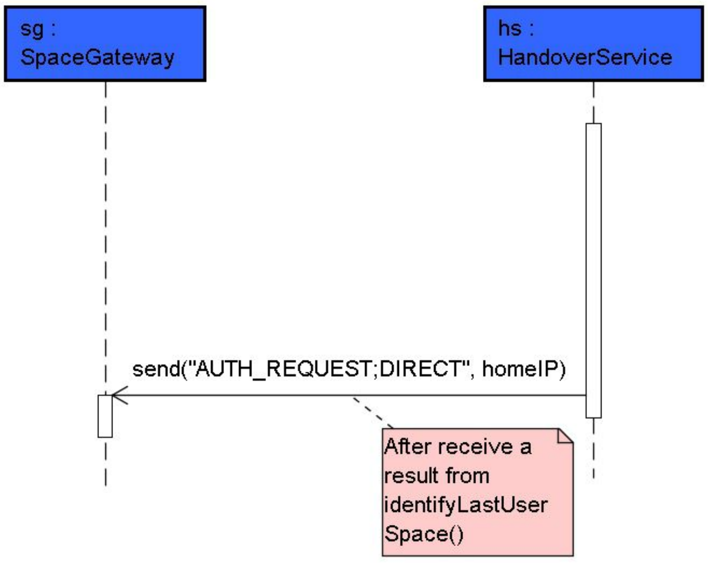
Given the activities grouped in the **Start** phase and described above, we can affirm that an AAL Space powered by the *SmartAndroid+* is able to handle handover requests from another AAL Space and further starts the appropriated handover protocol, thus answering **Question 2.a**.

### 7.4.3 Transferring Agents

The next phase of the scenario consists in transferring the handover agents with direct handover support from the **Home** to the **Gym**. Although the first activity **Transfer**



(a) Part 1: Notifying user presence, getting supported handover and identifying last user space


sg : SpaceGateway
hs : HandoverService
send("AUTH_REQUEST;DIRECT", homeIP)
After receive a result from identifyLastUser Space()

(b) Part 2: Sending an authorization request to the Home

Figure 7.6: Sequence Diagrams to the interaction of Management Services during the main event **Enters the Gym**.

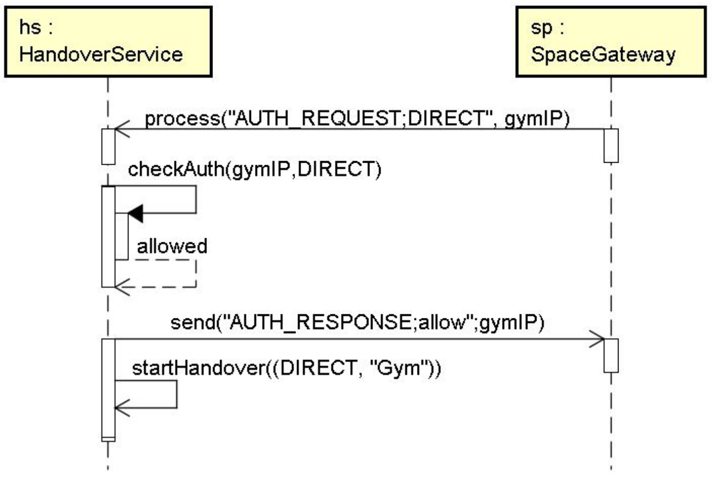

hs : HandoverService
sp : SpaceGateway
process("AUTH_REQUEST;DIRECT", gymIP)
checkAuth(gymIP,DIRECT)
allowed
send("AUTH_RESPONSE;allow";gymIP)
startHandover((DIRECT, "Gym"))

Figure 7.7: The steps of the activities **Process authorization request** and **Start direct handover protocol**

.

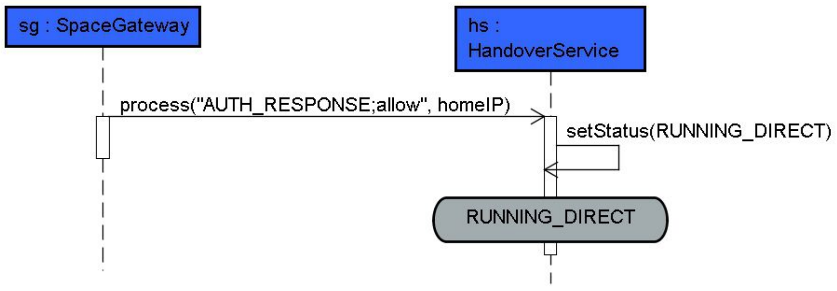
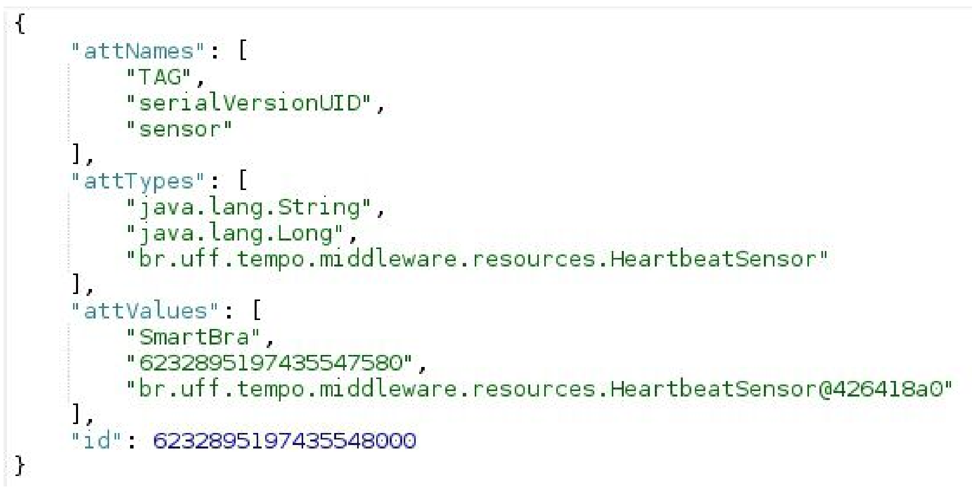

sg : SpaceGateway
hs : HandoverService
process("AUTH_RESPONSE;allow", homeIP)
setStatus(RUNNING_DIRECT)
RUNNING_DIRECT

Figure 7.8: The steps of the activity **Process authorization response** at the **Gym**.

**Handover Agents** appears in the home lane, it reflects directly in the **Gym**. Therefore, we use sequence diagrams to illustrate the activity steps in both AAL Spaces. At home, as depicted in Figure 7.10, for the *HandoverService* requests from the *Repository* the agents with direct handover support and receives a return in the form of a list of *JSONObject* [52]. In the Figure 7.9 we provide an example of an instance of *JSONObject* to a handover agent, which, for the sake of simplicity, we will call agent data.

Once the list of agents is received, first, the *HandoverService* notifies the **Gym** about the start of the agents transfer with the message `START_AGENTS_TRANSFER`. Next, it transfers each agent data to the **Gym** following the transfer agent message pattern as follows: `‘‘UPDATE_AGENT;JSONObject’’`. Additionally, each agent is added to a list named **agentsToUnregister**. In parallel, as indicated in Figure 7.11, **Gym** receives those messages, processes them and adds each agent data in the list named **agentsToUpdate**, which will further be used in the **Update** phase. When agent data are transferred, the *HandoverService* of the **Home** notifies the **Gym** about the end of the agents transfer



```
{
    "attNames": [
        "TAG",
        "serialVersionUID",
        "sensor"
    ],
    "attTypes": [
        "java.lang.String",
        "java.lang.Long",
        "br.uff.tempo.middleware.resources.HeartbeatSensor"
    ],
    "attValues": [
        "SmartBra",
        "6232895197435547580",
        "br.uff.tempo.middleware.resources.HeartbeatSensor@426418a0"
    ],
    "id": 6232895197435548000
}
```

Figure 7.9: An example of the JSONObject to HandoverAgent.

using the message `FINISH_AGENTS_TRANSFER`, thus finishing the **Transfer** phase.

In this section the activity **Transfer Handover Agents** was described step by step, demonstrating how the *SmartAndroid+* obtains the agents that support a given handover pattern, serializes them and transfers those serialized objects from one AAL Space to another. Thus, we can conclude that it answers **Question 2.b**.

### 7.4.4 Updating Agents

After receiving the message `FINISH_AGENTS_TRANSFER`, the **Gym** starts to update the handover agents in the *Repository* according to the received agent data. It represents the activity **Update Handover Agents** of the scenario. All steps of the algorithm to update the agents are described in the Section 6.3.4. Here, we demonstrate the updating method with a large degree of abstraction, as displayed in the Figure 7.12. In general, this method compares *agent ID* of the **agentsToUpdate** elements with the same attribute of all handover agents registered in *Repository*. When both IDs are equal, the *HandoverService* sets each attribute of the handover agent with its respective attribute in the agent data. Next, the respective handover agent is added in the list **agentsToRegister**. Once the comparison loop finishes, the *HandoverService* notifies the **Home** by the message: `FINISH_AGENTS_UPDATE`.

Considering the steps of the *Update Handover Agents* activity and the algorithm described in the Section 6.3.4, we can affirm that the *SmartAndroid+* is able to update handover agents from the agents data received from another AAL Space. This activity represents one of the most important steps of any handover protocol because it guarantees

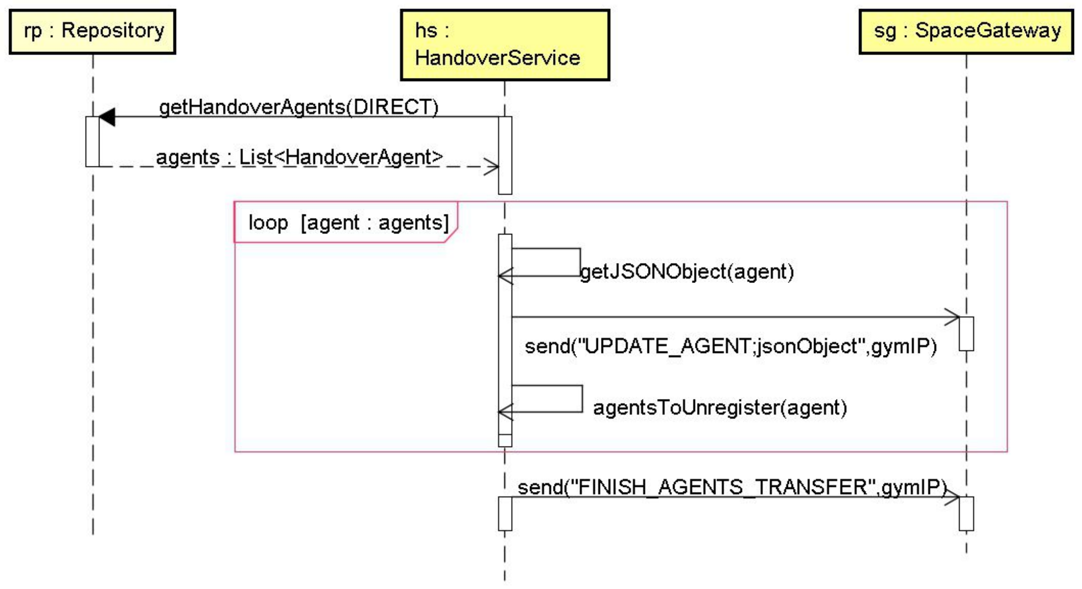

rp : Repository
hs : HandoverService
sg : SpaceGateway
getHandoverAgents(DIRECT)
agents : List<HandoverAgent>
loop [agent : agents]
getJSONObject(agent)
send("UPDATE_AGENT;jsonObject",gymIP)
agentsToUnregister(agent)
send("FINISH_AGENTS_TRANSFER",gymIP)

Figure 7.10: Sequence diagram describing the overflow of the process of transferring agents from the **Home**.

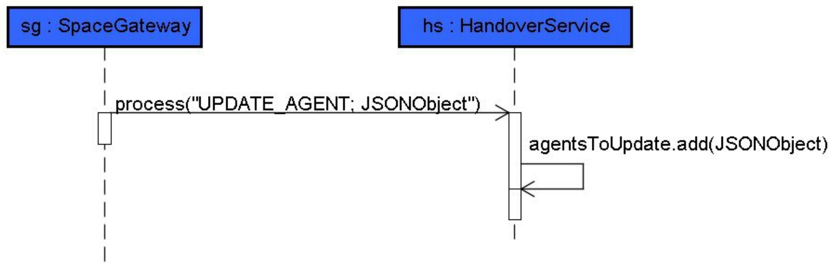

sg : SpaceGateway
hs : HandoverService
process("UPDATE_AGENT; JSONObject")
agentsToUpdate.add(JSONObject)

Figure 7.11: **Transfer Handover Agents** activity at the **Gym**.

that the context and the configuration of the continuous service are preserved in the new AAL Space. In this context, the above points answer **Question 2.c**.

## 7.4.5 Finishing Handover

The last phase of the direct handover scenario starts from the **Finish Handover** activity at the **Gym**. As indicated in the Figure 7.13, after the finished agents update, the *HandoverService* of the **Gym** notifies the *Home* using the message `FINISH_HANDOVER`. Next, two activities run in parallel. At **Home**, the *HandoverService* unregisters the agents in the list **agentsToUngerister**, as depicted in Figure 7.14. On the other hand, at the **Gym**, the *HandoverService* register the agents in the list **agentsToRegister**. It is worth mentioning that, during the register step, the Heart Monitoring System looks for a SmartBra to set as its data source. This feature is possible because a handover agent can manage a list of the type of possible data sources, allowing it to adapt itself in regards to

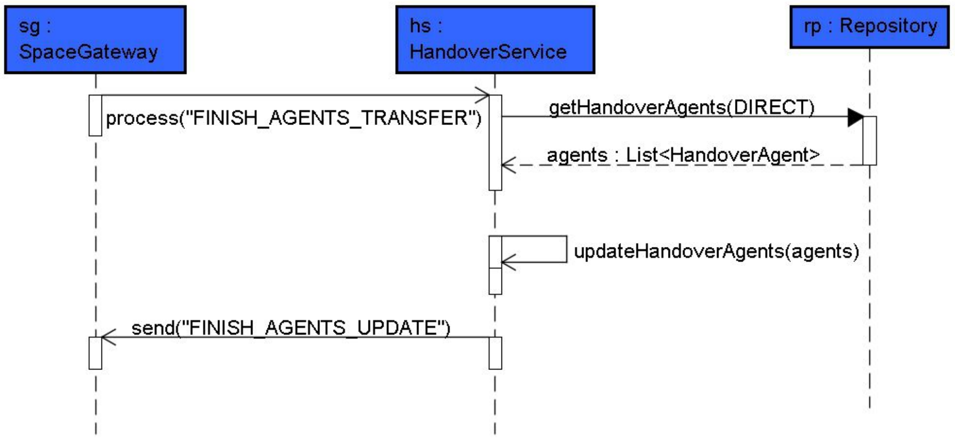

sg : SpaceGateway
hs : HandoverService
rp : Repository
process("FINISH_AGENTS_TRANSFER")
getHandoverAgents(DIRECT)
agents : List<HandoverAgent>
updateHandoverAgents(agents)
send("FINISH_AGENTS_UPDATE")

Figure 7.12: **Update Handover Agents** activity at the **Gym**.

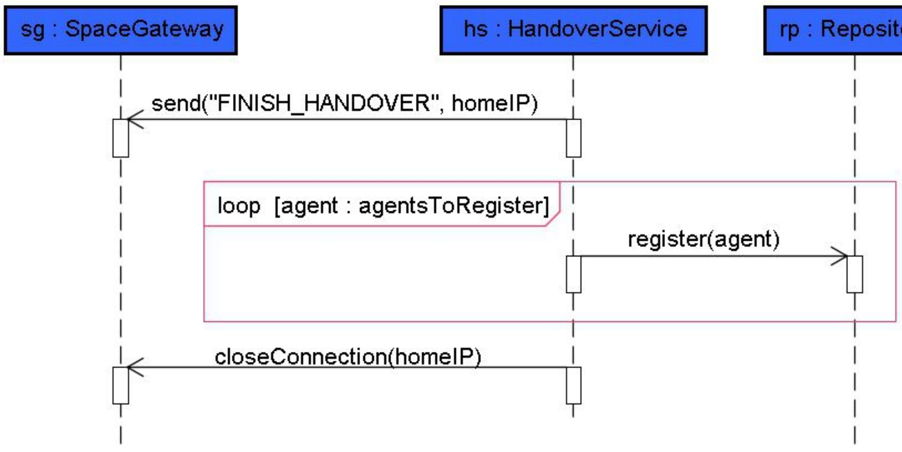


Figure 7.13: Steps of the **Gym** activities **Finish Handover** and **Register Handover Agents**.

data acquisition. Once the routines are finished, the direct handover is finished, and the **Gym** is now responsible for provisioning the heart monitoring system to Maria.

In this section, the last steps of the direct handover protocol were described, thus demonstrating how and when an AAL Space stops to provision a service to another AAL Space that then starts to do it. Additionally, we demonstrate how a system designed as a handover agent automatically sets its data source in a new AAL Space, thus answering **Question 2.d**. Finally, the scenario described all steps of a direct handover protocol, allowing Maria to go to the **Gym** without concerns about her heart monitoring, thus answering **Question 1.c**.

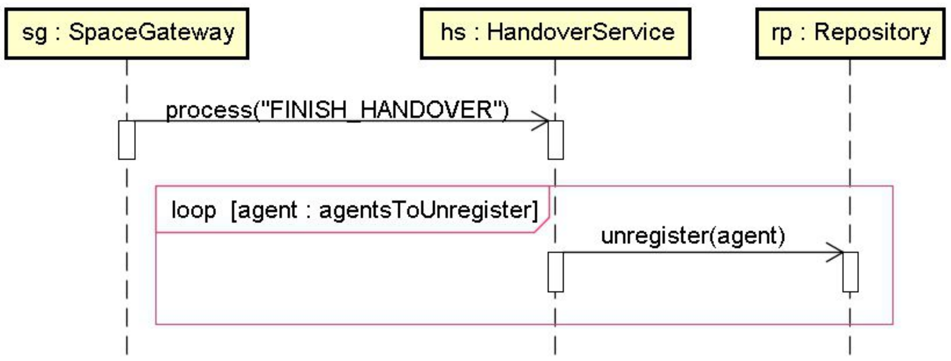


Figure 7.14: **Unregister Handover Agents** activity at **Home**.

## 7.5 Indirect Handover

In this section we evaluate the *SmartAndroid+* in regards to the indirect handover, i.e., a user leaving an AAL Space for an area that is not covered, also called a blind area. We consider the handover protocol triggered not only by a user leaving event but also by a user entering event. The goals of the experiment are to validate that the heart monitoring system can be continuously provisioned to the user in an AAL Space and in a no coverage area considering two perspectives, the user leaving an AAL Space and entering an AAL Space, thus answering **Question 1.a** and **Question 1.b**, respectively. Two scenarios were created based on Maria's story, one to represent the user leaving home and another scenario to represent the user returning home.

### 7.5.1 Scenario 2: Going for a Walk

Maria was doing relaxing exercises at the Gym until she came back home at the end of the morning. Four hours later, Maria decided to go for a walk in the park with her friend Juliette. Before leaving, she checked her Smart Bra and obtained her smart phone. Then, she left the house to meet Juliette. They calmly walked for nearly one hour, enjoying the fresh air and the surrounding nature. After this refreshing afternoon, Maria decided to go back home. Once at home, she placed the smart phone on the dining table, sat on the sofa and started to watch her favorite soap opera.

This short scenario continues the story of Maria begun in Scenario 1 (Section 7.4.1). To address these requirements, two indirect protocols must be employed. The first one is an indirect protocol for leaving events, which consists in a stationary AAL Space passing the service provision to a mobile AAL Space after a trigger by a leaving event. The second one, the indirect protocol for entering events, involves a stationary AAL Space receiving

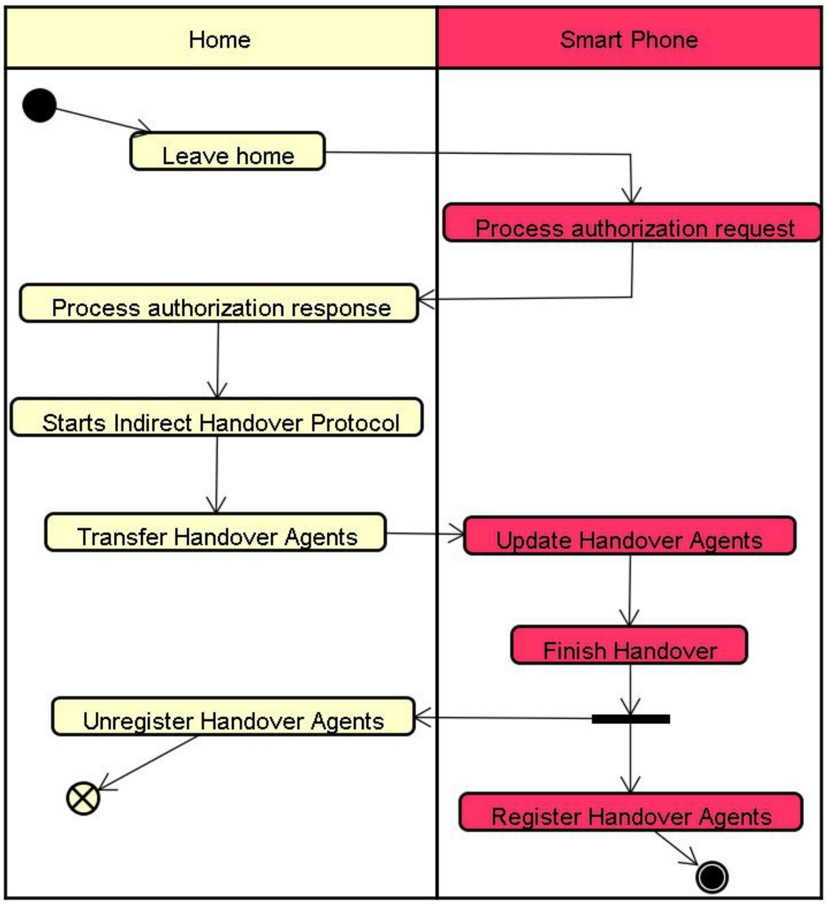

Home
Smart Phone
Leave home
Process authorization request
Process authorization response
Starts Indirect Handover Protocol
Transfer Handover Agents
Update Handover Agents
Finish Handover
Unregister Handover Agents
Register Handover Agents

Figure 7.15: Overview of the main events of Scenario 2 regarding Maria leaving the home.

the service provision from a mobile AAL Space after the trigger of an entering event. Please see Section 7.5.2 and Section 7.5.3.

### 7.5.2 Leaving

In this section, we describe the main events of the Indirect Handover, addressing the above scenario in regards to the leaving event. The two-lane diagram displayed in Figure 7.15 represents the AAL Spaces involved in this scenario, which are Home (yellow lane) and Smart Phone (red lane). Although each handover protocol addresses a specific type of scenario, they share a large part of the activities, sometimes changing only the order. Therefore, we discuss in this scenario only the Activity Diagram for the main activities of the Indirect Handover protocol for the leaving event.

In general, when Maria leaves the house (activity **Leaves Home**, the *PresenceDetection* notifies the event `USER_LEAVING` to the *HandoverService* which further obtains from the *Repository* the supported handover protocol. As the return was `BOTH`, it is the **Home** that controls the handover protocol, starting with its *HandoverService* and obtaining from the *LocationService*, in particular the *MobileMap*, the RANS of the registered mobile AAL Space. Then, an authorization request (``` ``AUTH_REQUEST;INDIRECT'' ```) is sent to the **Smart Phone**, which processes such messages (activity **Process Authorization Request**) and answers with an authorization response message (``` ``AUTH_RESPONSE;allowed'' ```).

Once authorized (activity **Process Authorization Response**), the activity **Start Indirect Handover Protocol** starts, followed by the activity **Transfer Handover Agents**, which consists in the *HandoverService* of the **Home** obtaining all agents set as the indirect handover from the *Repository* and running the transfer routine as described in Section 7.4.3, changing the **Gym** to the **Smart Phone**. In the next step, the activity **Update Handover Agents**, the agents are updated following the same strategy in Section 7.4.4. Next, the *Smart Phone* request to finish the protocol (activity **Finish Handover**) by sending the message ``` ``FINISH_HANDOVER'' ```. In parallel, the *HandoverService* of the **Smart Phone** registers all agents updated and the *HandoverService* of the **Home** unregisters all agents transferred. At the end of these steps, the **Smart Phone** closes the connection with the **Home**, thereby allowing Maria to safely going walk in the park.

We described the necessary steps above to run an Indirect Handover Protocol considering the user leaving a stationary AAL Space, represented in this scenario as a **Home**. This demonstrates that the *SmartAndroid+* is able to continuously provision a service even in blind areas, thus answering **Question 1.a**.

### 7.5.3 Entering

Continuing Maria's story, in this section, we describe the Indirect Handover protocol from the perspective of the user entering an AAL Space, which in this case, is Maria returning home. The Figure 7.16 indicates the protocol steps as an Activity Diagram arranged in two lanes representing the **Home** (yellow lane) and the **Smart Phone** (red lane). This handover protocol starts from the activity **Enters Home**, which comprises the *PresenceDetection* notifying the event `USER_ENTERING` to the *HandoverService*. The HandoverService further obtains the supported handover from the *LocationService* and requests authorization from the **Smart Phone** by sending the message ``` ``AUTH_REQUEST;INDIRECT'' ```.

Next, in the **Smart Phone**, the *ServiceGateway* receives such a message and forward it to the *HandoverService* to further process it (activity **Process Authorization Request**). Then, the **Smart Phone's** *HandoverService* responds with the re-

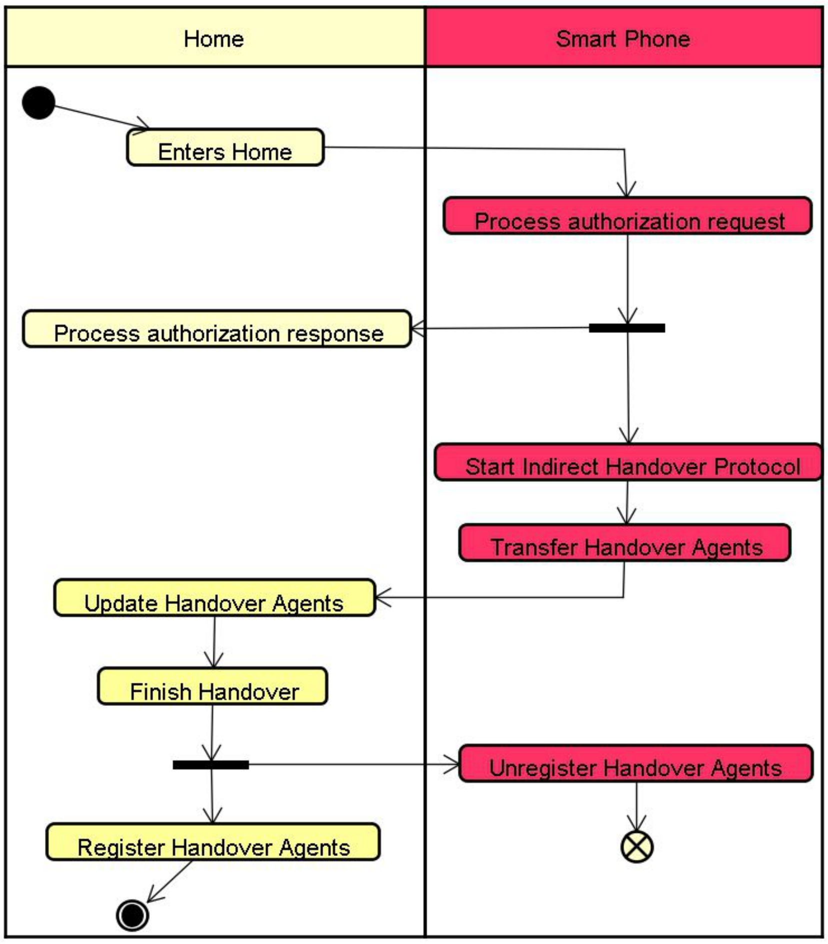

Home
Smart Phone
Enters Home
Process authorization request
Process authorization response
Start Indirect Handover Protocol
Transfer Handover Agents
Update Handover Agents
Finish Handover
Unregister Handover Agents
Register Handover Agents

Figure 7.16: Overview of the main events of Scenario 2 regarding Maria returning home.

quest (``AUTH_RESPONSE; INDIRECT'') and starts the Indirect Handover Protocol. While the **Home** prepares itself to receive the handover agents, the **Smart Phone** starts the activity **Transfer Handover Agents**, which runs in the same way that as homonymous activity described in the Section 7.4.3. This situation is shared with the activity **Update Handover Agents** at **Home**, which initiates after receiving the message ``FINISH_AGENTS_TRANSFER''.

Once all handover agents are updated, the activity **Finish Handover** initiates, notifying the **Smart Phone** and calling the last activity of the scenario, **Register Handover Agents**. While the *HandoverService* of the **Smart Phone** unregisters the agents transferred, the *HandoverService* of the **Home** registers the agents updated. Both activities are also performed in the same way as the homonymous presented in the previous scenarios. In the end, the **Home** is given back the responsibility to provision the heart monitoring service to Maria from the **Smart Phone**.

Given the set of activities displayed in the Figure 7.16 and described in this section, we can conclude that the *SmartAndroid+* is able to guarantee the continuity of a service even when the user comes from a no coverage area, i.e., a blind area. Additionally, the Indirect Handover (considering both perspectives) addresses Scenario 2 *subsec:scenario2* in its totality because our AAL Platform provides mechanisms to pass the service provision to a Smart Phone (mobile space) when Maria leaves the house and to regain this role when she comes back. Given these circumstances, **Question 1.b** was answered.

## 7.6 Performance

A second set of tests was conducted to verify the handover latency of our scenarios. For our performance test, we set up the scenarios in their respective infrastructures, as described in Section 7.1. Scenario 1, installed in the Infrastructure to Direct Handover, was run 10 times. Scenario 2, set up in the Infrastructure to Indirect Handover, was executed 20 times, 10 times from the `USER_ENTERING` perspective. In Table 7.1 we summarize the test results, demonstrating for each simulated scenario the average latency and the confidence interval with 95% of confidence level.

Table 7.1: Handover Latency of the scenarios considering 10 run each.

| | **Average Latency (seconds)** |
|---|---|
| **Scenario 1** | $2.887 \pm 0.217$ |
| **Scenario 2 : Leaving** | $3.887 \pm 0.220$ |
| **Scenario 2 : Entering** | $3.519 \pm 0.265$ |

The handover latency average for all scenarios is $3.431 \pm 0.348$ seconds. During this meantime, we cannot guarantee the service provision to Maria because the handover protocol is running. A latency of $3.431 \pm 0.348$ seconds indicates approximately $6 \sim 7$ heart beats because each one is measured every 0.5 seconds. In the worst case scenario, it may be 7 heartbeats lost during the migration of the user to a new AAL Space or to a blind area. However, this handover latency may be considered tolerable and may not be identified by Maria or her physician. Unlike in mobile networks (discussed in the Section 2.4.2), the migration in the AAL Domain tends to be less frequent. Nevertheless, such tests were performed in environments equipped with a Wi-Fi.

These analyses prompt discussion for further works that should create more complex infrastructures, setting up service coverage connected by MAN networks, for instance. Above all, the handover latency identified in the scenarios suggests that it is feasible for a coverage area composed of Wi-Fi sub-networks in a WLAN.

## 7.7 Comparison

In this section we present a systematic comparison among the AAL Service Platforms discussed in Chapter 3 and the *SmartAndroid+*. The comparison criteria highlights, for each platform, six general aspects derived from the the Competence Questions defined in Section 7.3. These aspects can be associated with the main features of an AAL Platform related to continuous service provision discussed in Chapter 4 and 5. The aspects are listed as follows:

- **A1 - Service provision in blind areas:** The platform is able to keep the service provision when the user leaves the covered space towards a blind area and returns. This feature comprises the **Questions 1a and 1b**, which are related to the indirect handover pattern.
- **A2 - Service provision in covered areas:** The platform can transfer the service provision to another AAL Space. This feature is related to the **Question 1c**, which is related to the direct handover pattern.
- **A3 - Management of handover requests:** The platform has mechanisms to receive and to respond handover requests from other AAL Spaces. The **Question 2a** is related to this feature.
- **A4 - Transfer of agents:** The platform features mechanisms to transfer the context of the agents from the current AAL Space to another. The **Question 2b** is related to this feature.
- **A5 - Synchronization of agents:** The platform is able to update the context of the agents received from another AAL Space. The **Question 2c** is related to this feature.
- **A6 - Automatic setting of data sources for agents:** When entering in a different AAL Space, the agents of the platform can find and set a data source available to provide the adequate context data. The **Question 2d** is related to this feature.

In order to classify the AAL Service Platforms according to the main aspects described above, we propose three categories, as follows:

1. **N** : The platform does not address the aspect or does not mention the aspect.

2. **P** : The platform discusses the aspect but does not provide details about how the aspect is tackled.

3. **A** : The platform addresses the feature.

In Table 7.2, the main aspects of the *SmartAndroid+* are compared with those of the related platforms discussed in Chapter 3 considering the categories proposed above. One can notice that while the related platforms address aspects related either to the indirect handover (A1) or to the direct handover (A2), the *SmartAndroid+* addresses aspects of both handover patterns. Additionally, although most of these platforms match the aspects related to support of resources (A3, A4 and A5) only the *Greener Building* and *SmartAndroid+* detail the implementation and application of these mechanisms. Finally, aspect A6 is addressed only by the *SmartAndroid+*.

Table 7.2: Comparison of AAL Platforms presented in Related Works and the *SmartAndroid+*

| | **GatorTech** | **CASAS** | **Persona** | **Uranus** | **Greener Building** | **Smart Android+** |
|---|---|---|---|---|---|---|
| **Service provision in blind areas** | N | N | N | A | N | A |
| **Transference of service provision to another space** | N | P | A | N | A | A |
| **Management of handover requests** | N | P | A | A | A | A |
| **Transference of agents** | N | P | P | N | A | A |
| **Synchronization of agents** | N | P | P | N | A | A |
| **Automatic set data source for agents** | N | N | N | P | N | A |

## 7.8 Discussion

In this chapter, we evaluated the *UbiTempo+* and a prototype implementation, the *SmartAndroid+*, using competence questions formulated to limit our research problem to how to provide continuous services to a user in the AAL Domain. For this purpose, we divided these competence questions into two groups, one focused on the features that support the migration of agents through AAL Spaces and the other related to the necessary resources to start, perform and manage handover protocols.

These questions represent the fundamental points of our approach and are used to evaluate the behavior of our framework regarding the execution of the direct protocol and the two variances of the indirect handover protocol. They also address the framework resources, evaluating the mechanisms for managing handover requests, transferring agents to an AAL Space, and updating recently migrated agents in addition to the flexibility of a continuous system to set up a data source in a new AAL Space. Demonstrating all features challenged by the competence questions, we could prove that *SmartAndroid+* — and *UbiTempo+*, overall — reaches the requirements that are proposed, validating our approach. Additionally, when *SmartAndroid+* is compared with the related platforms discussed in Chapter 3, the original contributions of this work are clarified.

In the next chapter, we discuss the contributions and limitations of the proposed framework, also presenting topics for future research.

# Chapter 8

# Conclusion

In this master thesis a flexible framework for supporting service handover of AAL Services was proposed. First, the handover patterns in the AAL domain were identified, which are: (i) direct, when the user migrates between intersecting AAL Spaces; and (ii) indirect handover, when the user migrates between AAL Spaces with no intersection between their coverage areas. In addition, we formalized protocols to perform direct and indirect handover in AAL Platforms. Next, a functional architecture was proposed, comprising elements to manage services handover in the environment: *Space Abstraction*, *Location Service*, *Space Gateway*, *Handover Handler* and *Synchronization Service*. This architecture can be combined with a predefined AAL Platform in order to add service handover support, reusing mechanisms to design, develop and test ubiquitous systems. In order to validate our proposal, a flexible framework, which is an extension of the one proposed by the LabTempo [59, 34, 12], was implemented. For the sake of simplicity, we call the reference framework *UbiTempo* and our approach *UbiTempo+*.

*UbiTempo+* was designed to provide mechanisms to transfer the provision of a service from the current AAL Space to another. It comprises middleware core services to handle communication with remote AAL Spaces and to perform migration of agents and manage the service coverage area. Moreover, the flexible framework aims at simplifying the design and implementation of AAL Services with requirements of continuity through different spaces.Additionally, *UbiTempo+* aims at expanding the service coverage of an AAL Platform by providing mechanisms to interconnect AAL Spaces, enabling them to enlarge the covered area as much as possible. Migration through the covered areas does not require the user to carry any device to store information about the AAL Services (e.g. context, status etc) since the AAL Spaces run the appropriated handover protocols for transparently performing the user migration. Finally, we propose the use of mobile devices, such as smart phones, to provide service coverage in blind areas, i.e., areas out of the range of any AAL Space. Combining both approaches might reduce the issues of continuous service provision in AAL scenarios.

In the AAL domain, continuous services can be used to monitor the health status of the user, as discussed along this thesis and in particular in Maria's Story (Section 2.1). If a health monitoring system can be made available in any environment enabled by the *UbiTempo+*, the user may feel safer and more independent to perform daily activities outside the house, such as walking on the park, going to supermarket and hanging out with friends. For example, in our reference scenario, Maria seems to have a busy lifestyle, attending a senior gym three times a week and walking in the park every day.

The concepts of the *UbiTempo+* were instantiated as a middleware called *SmartAndroid+*. This middleware has such name because it is an extension of *SmartAndroid*, the reference implementation of *UbiTempo* [59]. Our middleware provides an API for developers and a set of core management services, allowing them to design and develop AAL Services that should attend handover requirements. *SmartAndroid* implemented four core management services: Register Service, Discovery Service, Name Service and Location Service. The *SmartAndroid+* reuses those four services adding two more: Handover Service and Space Gateway.

Nevertheless, our present flexible framework proposal and implementation have some limitations. First of all, only one handover operation is supported at a given time. Secondly, the migration from a mobile space to another is not supported by the framework. As consequence, it is not possible to transfer the provision of a service to another mobile device (e.g. tablet or another smart phone) when a smart phone runs out of battery. Finally, in our test scenarios the only communication technology we considered was Wi-Fi network, although there is a wide-range of types of sensors and actuators that have only Bluetooth connections available.

## 8.1 Future Works

Several topics that could possibly be tackled in future works are identified. At first, to support more than one user is the natural evolution of the framework, allowing it to wide the range of possible real world scenarios. For this purpose, the *Handover Service* should be able to manage information about the users at the space such as his position and the services provided for him. Furthermore, a user (e.g. person, resident, patient etc) may be represented as either a *Resource Agent* or a *Handover Agent*. As consequence, in the middleware level, the core services would not be adapted. It is important to mention that the presence detection system installed in the space also must be able to handle more than one user, which also implies to adapt its interface with the *Handover Service*.

Another aspect to be approached in the future is the use of a concrete and real world AAL system to test our framework, in particular some application implemented using our middleware, the *SmartAndroid+*. Deploying an application designed using our framework and running upon our middleware for a real world scenario could bring great advantage, indicating problems to be corrected in our implementation or new features to be added. This accomplishment would enable a practical analysis of the non-functional attributes of the framework, yielding their improvement and favouring a discussion about handover in AAL, which is still an unsolved problem.

In our approach, AAL Spaces must have in their repository of agents, which is called the *Repository*, an instance of the agents that will be migrated through the covered area. It represents a risk for the service's continuity, increasing the possibility of service interruption whenever the user crosses AAL spaces where the service agent is not available. In order to reduce the risk, an strategy can be to transfer the agent instance instead of only the agent data during the handover protocol. Thus, any AAL Space would be able to provide the service. One approach to solve this issue may be to adapt the SmartAndroid+ to support Linux Containers [71, 35], applying some containers engines such as Docker [62], for instance. Other ideas are discussed in [20, 7].

Although privacy was not mentioned in our scenario, adapting the *Security Management Service* [59], which was already proposed by *UbiTempo*, can be considered an important improvement to the *UbiTempo+*. The idea is to employ a more secure authorization mechanism and communication channel between stationary spaces. For instance, applying some data encryption strategies in the messages exchanged by the spaces or deploying SSL (Secure Socket Layer) support in the *Space Gateway*, considering any external connection (e.g. spaces, third party services and medical services).

Finally, a direct way to improve our work is by making our protocol more robust such as dealing with communication problems during the handover process, i.e., message loss and disconnections. In our approach, we assumed that the communication was reliable. As such, we did not include confirmation messages in our protocol, and hence, the loss of any message can cause a handover operation to be discontinued, with no warning being sent to the clients.

# Bibliography

[1] Online etymology dictionary.

[2] Aarts, E. *Ambient Intelligence in HomeLab.* Philips Research, 2002.

[3] Aarts, E.; Wichert, R. *Ambient intelligence.* Springer, 2009.

[4] Acampora, G.; Cook, D. J.; Rashidi, P.; Vasilakos, A. V. A survey on ambient intelligence in healthcare.

[5] Aloulou, H.; Mokhtari, M.; Tiberghien, T.; Biswas, J.; Phua, C.; Lin, J. H. K.; Yap, P. Deployment of assistive living technology in a nursing home environment: methods and lessons learned. *BMC Medical Informatics and Decision Making 13*, 1 (2013), 42.

[6] Argoneto, P.; Renna, P. Multi-agent architecture. In *Innovative Tools for Business Coalitions in B2B Applications.* Springer, 2011, pp. 31–47.

[7] Augusto, J. C. Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence. In *Intelligent Computing Everywhere.* Springer, 2007, pp. 213–234.

[8] Baker, N. Zigbee and bluetooth: Strengths and weaknesses for industrial applications. *Computing and Control Engineering 16*, 2 (2005), 20–25.

[9] Balasubramaniam, S.; Indulska, J. Vertical handover supporting pervasive computing in future wireless networks. *Computer Communications 27*, 8 (2004), 708–719.

[10] Baldoni, R.; Di Ciccio, C.; Mecella, M.; Patrizi, F.; Querzoni, L.; Santucci, G.; Dustdar, S.; Li, F.; Truong, H.-L.; Albornos, L., et al. An embedded middleware platform for pervasive and immersive environments for-all. In *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops' 09. 6th Annual IEEE Communications Society Conference on* (2009), IEEE, pp. 1–3.

[11] Barish, G.; Lester, P.; Saltzman, W. R.; Elbogen, E. Beyond sensors: Reading patients through caregivers and context. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (New York, NY, USA, 2014), UbiComp '14 Adjunct, ACM, pp. 1273–1277.

[12] Barreto, D. Interface de prototipagem e gerenciamento de aplicações pervasivas. Dissertação de mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, 2013.

[13] Bertamini, F.; Brunelli, R.; Lanz, O.; Roat, A.; Santuari, A.; Tobia, F.; Xu, Q. Olympus: An ambient intelligence architecture on the verge of reality. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on* (2003), IEEE, pp. 139–144.

[14] Bose, S.; Kannan, A. Detecting denial of service attacks using cross layer based intrusion detection system in wireless ad hoc networks. In *Signal Processing, Communications and Networking, 2008. ICSCN'08. International Conference on* (2008), IEEE, pp. 182–188.

[15] Brumitt, B.; Meyers, B.; Krumm, J.; Kern, A.; Shafer, S. Easyliving: Technologies for intelligent environments. In *Handheld and ubiquitous computing* (2000), Springer, pp. 12–29.

[16] Camarinha-Matos, L. M.; Afsarmanesh, H. Design of a virtual community infrastructure for elderly care. In *Collaborative Business Ecosystems and Virtual Enterprises*. Springer US, 2002, pp. 439–450.

[17] Cardoso, L. Integração de serviços de monitoração e descoberta de recursos a um suporte para arquiteturas adaptáveis de software. Dissertação de mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, 2006.

[18] Chalmers, D.; Sloman, M. A survey of quality of service in mobile computing environments. *Communications Surveys & Tutorials, IEEE 2*, 2 (1999), 2–10.

[19] Cook, D.; Schmitter-Edgecombe, M.; Crandall, A.; Sanders, C.; Thomas, B. Collecting and disseminating smart home sensor data in the casas project. In *Proceedings of the CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research* (2009).

[20] Cook, D. J.; Augusto, J. C.; Jakkula, V. R. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing 5*, 4 (2009), 277–298.

[21] Cook, D. J.; Crandall, A. S.; Thomas, B. L.; Krishnan, N. C. Casas: A smart home in a box. *Computer 46*, 7 (2013).

[22] Cook, D. J.; Das, S. K. How smart are our environments? an updated look at the state of the art. *Pervasive and mobile computing 3*, 2 (2007), 53–73.

[23] Cook, D. J.; Youngblood, M.; Das, S. K. A multi-agent approach to controlling a smart environment. In *Designing smart homes.* Springer, 2006, pp. 165–182.

[24] Copetti, A. *Monitoramento Inteligente e Sensivel ao Contexto na AssistÃªncia Domiciliar Telemonitorada.* Tese de Doutorado, Instituto de Computacao, Universidade Federal Fluminense, Niteroi, RJ, Brasil, 2010.

[25] Copetti, A.; Leite, J.; Loques, O.; Neves, M. F. A decision-making mechanism for context inference in pervasive healthcare environments. *Decision Support Systems* (2012).

[26] Coronato, A. Uranus: A middleware architecture for dependable aal and vital signs monitoring applications. *Sensors 12*, 3 (2012), 3145–3161.

[27] Das, B.; Seelye, A. M.; Thomas, B. L.; Cook, D. J.; Holder, L. B.; Schmitter-Edgecombe, M. Using smart phones for context-aware prompting in smart environments. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE* (2012), IEEE, pp. 399–403.

[28] Das, S. K.; Cook, D. J. Designing smart environments: A paradigm based on learning and prediction. In *Pattern Recognition and Machine Intelligence.* Springer, 2005, pp. 80–90.

[29] Degeler, V.; Gonzalez, L. I. L.; Leva, M.; Shrubsole, P.; Bonomi, S.; Amft, O.; Lazovik, A. Service-oriented architecture for smart environments (short paper). In *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on* (2013), IEEE, pp. 99–104.

[30] Degeler, V.; Lazovik, A. Architecture pattern for context-aware smart environments. *Creating Personal, Social and Urban Awareness through Pervasive Computing* (2013), 108–130.

[31] Demiris, G.; Hensel, B. Technologies for an aging society: a systematic review of smart home applications. *Yearb Med Inform 3* (2008), 33–40.

[32] ElHelw, M.; Pansiot, J.; McIlwraith, D.; Ali, R.; Lo, B.; Atallah, L. An integrated multi-sensing framework for pervasive healthcare monitoring. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on* (2009), IEEE, pp. 1–7.

[33] Endler, M.; Nagamuta, V. General approaches for implementing seamless handover. In *Proceedings of the second ACM international workshop on Principles of mobile computing* (2002), ACM, pp. 17–24.

[34] Erthal, M. Uma proposta para a interpretacação de contexto em ambientes inteligentes. Dissertação de mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, 2013.

[35] Felter, W.; Ferreira, A.; Rajamony, R.; Rubio, J. An updated performance comparison of virtual machines and linux containers. *technology 28* (2014), 32.

[36] Fengou, M.-A.; Panagiotakopoulos, T. C.; Fengos, S. L.; Lazarou, N. G.; Lymberopoulos, D. A new telemedicine framework handling the emergency room overload. In *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on* (2010), IEEE, pp. 1–4.

[37] Georgalis, Y.; Grammenos, D.; Stephanidis, C. Middleware for ambient intelligence environments: Reviewing requirements and communication technologies. In *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments.* Springer, 2009, pp. 168–177.

[38] Google. Android Platform. http://goo.gl/M9zDTb, 2015.

[39] Google. Android Platform - Activity. http://goo.gl/n73ocE, 2015.

[40] Google. Android Platform - Android Development Kit. http://goo.gl/l421dm, 2015.

[41] Google. Android Platform - Layouts. http://goo.gl/EPkwxV, 2015.

[42] Google. Android Platform - ScanResult. http://goo.gl/mn6Ffz, 2015.

[43] Grüninger, M.; Fox, M. S. Methodology for the design and evaluation of ontologies.

[44] Han, D.-M.; Lim, J.-H. Smart home energy management system using ieee 802.15. 4 and zigbee. *Consumer Electronics, IEEE Transactions on 56*, 3 (2010), 1403–1410.

[45] Hanke, S.; Mayer, C.; Hoeftberger, O.; Boos, H.; Wichert, R.; Tazari, M.-R.; Wolf, P.; Furfari, F. universaal–an open and consolidated aal platform. In *Ambient assisted living*. Springer, 2011, pp. 127–140.

[46] Hashimoto, A.; Mori, N.; Funatomi, T.; Yamakata, Y.; Kakusho, K.; Minoh, M. Smart kitchen: A user centric cooking support system. In *Proceedings of IPMU* (2008), vol. 8, pp. 848–854.

[47] Helal, A.; Cook, D. J.; Schmalz, M. Smart home-based health platform for behavioral monitoring and alteration of diabetes patients. *Journal of diabetes science and technology 3*, 1 (2009), 141–148.

[48] Helal, S.; Chen, C. The gator tech smart house: enabling technologies and lessons learned. In *Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology* (2009), ACM, p. 13.

[49] Helal, S.; Mann, W.; El-Zabadani, H.; King, J.; Kaddoura, Y.; Jansen, E. The gator tech smart house: A programmable pervasive space. *Computer 38*, 3 (2005), 50–60.

[50] IBGE. *Projeção da população do Brasil por sexo e idade, 1980-2050: revisão 2008.* Estudos e pesquisas: Informação demográfica e socioeconômica. IBGE, 2008.

[51] Jaschinski, C. Ambient assisted living: Towards a model of technology adoption and use among elderly users. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (New York, NY, USA, 2014), UbiComp '14 Adjunct, ACM, pp. 319–324.

[52] JSON. The Java Script Object Notation. http://www.json.org/, 2014.

[53] Kassar, M.; Kervella, B.; Pujolle, G. An overview of vertical handover decision strategies in heterogeneous wireless networks. *Computer Communications 31*, 10 (2008), 2607–2620.

[54] King, J.; Bose, R.; Yang, H.-I.; Pickles, S.; Helal, A. Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on* (2006), IEEE, pp. 630–638.

[55] Kleinberger, T.; Becker, M.; Ras, E.; Holzinger, A.; Müller, P. Ambient intelligence in assisted living: enable elderly people to handle future interfaces. In *Universal access in human-computer interaction. Ambient interaction*. Springer, 2007, pp. 103–112.

[56] Lee, C.; Nordstedt, D.; Helal, S. Enabling smart spaces with osgi. *Pervasive Computing, IEEE 2*, 3 (2003), 89–94.

[57] Leijdekkers, P.; Gay, V.; Lawrence, E. Smart homecare system for health tele-monitoring. In *Digital Society, 2007. ICDS'07. First International Conference on the* (2007), IEEE, pp. 3–3.

[58] Living, A. A. Ambient assisted living joint programme. *DOI= http://www. aal-europe. eu/projects/AALCatalogueV3. pdf,(last checked 30.11. 2011)* (2011).

[59] Mareli, D. Um framework de desenvolvimento de aplicações ubíquas em ambientes inteligentes. Dissertação de mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, 2013.

[60] Mareli, D.; Erthal, M.; Ferreira, D. B.; Loques, O. Um framework de desenvolvimento de aplicações ubíquas em ambientes inteligentes. In *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (2013), pp. 643–656.

[61] Mazza, M.; Lefèvre, F. A instituição asilar segundo o cuidador familiar do idoso. *Saúde e sociedade 13*, 3 (2004), 68–77.

[62] Merkel, D. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal 2014*, 239 (2014), 2.

[63] Montenegro, S.; Silva, C. A. B. d. Os efeitos de um programa de fisioterapia como promotor de saúde na capacidade funcional de mulheres idosas institucionalizadas. *Rev Bras Geriatr Gerontol 10*, 2 (2007), 161–78.

[64] Murphy, A. L.; Picco, G. P.; Roman, G. Lime: A middleware for physical and logical mobility. In *Distributed Computing Systems, 2001. 21st International Conference on.* (2001), IEEE, pp. 524–533.

[65] Nagamuta, V. *Um Arcabouço para Composiçao, Teste e Simulaçao de Protocolos de Handover Suave.* Tese de Doutorado, UNIVERSIDADE DE SAO PAULO, 2006.

[66] Nehmer, J.; Becker, M.; Karshmer, A.; Lamm, R. Living assistance systems: an ambient intelligence approach. In *Proceedings of the 28th international conference on Software engineering* (2006), ACM, pp. 43–50.

[67] Parra, J.; Hossain, M. A.; Uribarren, A.; Jacob, E.; El Saddik, A. Flexible smart home architecture using device profile for web services: A peer-to-peer approach. *International Journal of Smart Home 3*, 2 (2009), 39–56.

[68] Poslad, S. Ubiquitous computing smart devices, smart environments and smart interaction, 2009.

[69] Ranganathan, A.; Chetan, S.; Al-Muhtadi, J.; Campbell, R.; Mickunas, M. Olympus: A high-level programming model for pervasive computing environments. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on* (2005), IEEE, pp. 7–16.

[70] Román, M.; Hess, C.; Cerqueira, R.; Ranganathan, A.; Campbell, R. H.; Nahrstedt, K. A middleware infrastructure for active spaces. *IEEE pervasive computing 1*, 4 (2002), 74–83.

[71] Rosen, R. Linux containers and the future cloud. *Linux J 240* (2014).

[72] Stanford, V.; Garofolo, J.; Galibert, O.; Michel, M.; Laprun, C. The nist smart space and meeting room projects: signals, acquisition annotation, and metrics. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on* (2003), vol. 4, IEEE, pp. IV–736.

[73] Taylor, L.; Titmuss, R.; Lebre, C. The challenges of seamless handover in future mobile multimedia networks. *Personal Communications, IEEE 6*, 2 (1999), 32–37.

[74] Tazari, M.-R.; Furfari, F.; Ramos, J.-P. L.; Ferro, E. The persona service platform for aal spaces. In *Handbook of Ambient Intelligence and Smart Environments*. Springer, 2010, pp. 1171–1199.

[75] Tiberghien, T. *Strategies for context reasoning in assistive livings for the elderly.* Tese de Doutorado, Institut National des Télécommunications, 2013.

[76] Vallée, M.; Ramparany, F.; Vercouter, L., et al. *A multi-agent system for dynamic service composition in ambient intelligence environments.* na, 2005.

[77] Viterbo, J.; Endler, M.; Baptista, G. A two-tiered approach for decentralized reasoning in ambient intelligence. *Intelligent Systems, IEEE 25*, 5 (Sept 2010), 54–60.

[78] Viterbo, J.; Mazuel, L.; Charif, Y.; Endler, M.; Sabouret, N.; Breitman, K.; Seghrouchni, A. E. F.; Briot, J. Ambient intelligence: Management of distributed and heterogeneous context knowledge. *CRC Studies in Informatics Series. Chapman & Hall* (2008), 1–44.

[79] Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (1991), 94–104.

[80] Wolf, P.; Schmidt, A.; Klein, M. Soprano-an extensible, open aal platform for elderly people based on semantical contracts. In *3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'08), 18th European Conference on Artificial Intelligence (ECAI 08), Patras, Greece* (2008), Citeseer.

[81] Wu, C.-L.; Liao, C.-F.; Fu, L.-C. Service-oriented smart-home architecture based on osgi and mobile-agent technology. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 37*, 2 (2007), 193–205.

[82] Zeng, Q.-A.; Agrawal, D. P. Handoff in wireless mobile networks. *Handbook of wireless networks and mobile computing* (2002).