Erina Ferro, Michele Girolami, Dario Salvi, Christopher Mayer, Joe Gorman, Andrej Grguric, Roni Ram, Rubaiyat Sadat, Konstantinos M. Giannoutakis* and Carsten Stocklöw

The UniversAAL Platform for AAL (Ambient Assisted Living)

Abstract: This article describes the UniversAAL platform, an open platform intended to facilitate the development, distribution, and deployment of technological solutions for Ambient assisted living (AAL). The platform is intended to benefit end users (i.e., assisted persons, their families, and caregivers), authorities with responsibility for AAL, and organizations involved in the development and deployment of AAL services. It consists of an extensive set of resources (some are software and some are models/architectures) aimed at these different groups. The resources are classified into three main groups: runtime support, development support, and community support. The article presents the benefits that can be expected from the widespread adoption of the platform. The article also describes progress on prototype implementations of some of the software resources, and the results of initial evaluations of the platform. The work is partially based on results from earlier European Union-funded research projects in the area.

Keywords: Ambient assisted living, open source, execution platform, application development support.

2010 Mathematics Subject Classification: 00A99.

DOI 10.1515/jisys-2014-0127
Received August 29, 2014; previously published online February 4, 2015.

1 Introduction

As the proportion of elderly people in the population continues to increase, so too does the need to create a society where older people can be self-sustaining as long as possible, to maintain a satisfactory quality of life. Self-sufficiency, both in daily activities and in social relations, is the target of research in this sector. Thus, it is fundamental to develop services that help elderly people carry out everyday activities safely and effec-

***Corresponding author: Konstantinos M. Giannoutakis,** Information Technologies Institute (ITI), Centre for Research and Technology Hellas (CERTH), P.O. Box 60361, 6th km Xarilaou-Thermi, 57001 Thessaloniki, Greece, e-mail: kgiannou@iti.gr
Erina Ferro: ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
Michele Girolami: ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy; Department of Computer Science, Universita' di Pisa, Pisa, Italy
Dario Salvi: Life Supporting Technologies Group, ETSI Telecomunicación, Universidad Politécnica de Madrid, Avenida Complutense 30, 28040 Madrid, Spain
Christopher Mayer: AIT Austrian Institute of Technology GmbH, Health & Environment Department, Biomedical Systems, Donau-City-Str. 1, 1220 Vienna, Austria
Joe Gorman: SINTEF ICT, Strindveien 4, Trondheim, Norway
Andrej Grguric: Research and Innovations Unit, Ericsson Nikola Tesla, Krapinska 45, 10000 Zagreb, Croatia
Roni Ram: IBM R&D Labs in Israel, Haifa University Campus, Mount Carmel, Haifa 3498825, Israel
Rubaiyat Sadat: University of Siegen, Hölderlinstraße 3, 57076 Siegen, Germany
Carsten Stocklöw: Fraunhofer IGD, Fraunhoferstrasse 5, D-64283 Darmstadt, Germany

tively and, as far as possible, independently. This leads us to believe that the potential market for Ambient assisted living (AAL) solutions is huge, and that the adoption of a platform that helps developers produce such solutions can be a key factor facilitating the creation of new innovative services. Several developments in the fields of AAL already exist; however, these realizations are often based on proprietary architectures and, in many cases, they do not easily allow integration of external components or devices. This has some negative consequences, such as the fact that effective solutions in different regions in Europe are incapable of connecting both to relevant external systems (such as public health systems and social care systems) and to each other. Also, end users such as elderly people suffer consequences, when, for instance, hardware devices and sensors are not interchangeable. Non-integrability makes customers dependent on a specific vendor for products and services, impeding the acquisition of different devices to extend functionality, or to replace obsolete hardware.

To overcome these obstacles, the EU-FP7 project UniversAAL was set up in 2010, with its main goal being to design and develop an open platform that provides a standardized approach for an easy and economic development of AAL solutions. The UniversAAL platform benefits: (i) the end users (i.e., older adults and people with disabilities, their caregivers, and family members), by making new solutions affordable, simple to configure, and to deploy; (ii) the solution providers, by making it easier and cheaper to create innovative AAL services or to adapt existing ones by using a compositional approach based on existing components, services, and external systems; and (iii) the authorities with responsibility for AAL, by introducing a standardized approach for the development of AAL solutions that reduce vendor lock-in and generate an AAL market that lowers the prices and increases the availability of products. To stimulate an AAL market, UniversAAL has been designed to simplify the installation of UniversAAL-compliant applications by creating *uStore*, an online one-stop-shop for UniversAAL end-user services.

The UniversAAL platform is the result of the consolidation of the best aspects from existing available platforms in order to avoid fragmentation, reuse the functionalities already present in the other platforms, and avoid duplication by providing one common European implementation. An overview of these is provided in Section 2.

The concept of "platform" is used in a very broad sense in UniversAAL, and an overview of this is provided in Section 3. The task of designing a platform that takes account of numerous stakeholders and their requirements, deals with many technical issues, and successfully consolidates input projects is a very complex one. This needs to be approached in a systematic way, and the way we did this is described in Section 4.

The core part of the article is in Sections 5, 6, and 7, which describe the architecture of the UniversAAL platform. Section 8 reports on the work done in the project to evaluate the platform, and the article concludes by summarizing achievements in Section 9.

2 Related AAL Projects

Most of the AAL solutions nowadays available are still prototype solutions that are developed and maintained within different research projects. Often, such solutions are preliminary and lack maturity. There are some mature commercial products; however, these are typically provided as a closed solution.

Among the European research projects, we refer to projects AMIGO [2], MPOW-ER, PERSONA [19], SOPRANO [18], OASIS [15], VAALID [24], and GENESYS [7], which we call the UniversAAL "input" projects. These projects formed part of the e-Inclusion program of the FP6-IST (Information Society Technologies) framework. The projects obtained excellent results, although some weaknesses make them unsuitable candidates for widespread adoption. First, the projects are fragmented: the platforms adopted are based on the architectures and requirements specified within the respective consortium. Often, the requirements of these projects avoid the task of gathering wide acceptance around the architectural design of the platform. Second, the projects are targeted to demonstrate only some particular challenges of AAL. Some notable examples are as follows: show the performance of the designed platform, show the effectiveness of the semantic

interoperability, or highlight the benefits of the integration with heterogeneous hardware. These projects lack an integrated approach for all the technological challenges that arise from the AAL research domain. Third, the projects overlap with each other, in particular offering similar functionalities and addressing similar use cases. Often, we found that the differences between the projects are mostly concerning the techniques they adopt, or the specific technologies adopted (e.g., for sensors). All the previous points illustrate the need for a consolidation effort.

The design of the UniversAAL platform has taken into account the past experience of European Union research in AAL. UniversAAL exploits the best results coming from the existing platform architectures, and addresses two key-goals: form the basis for design of a commercial product and establish a standardized AAL platform.

3 The UniversAAL Platform

The UniversAAL platform provides support in three main areas: runtime support, development support, and community support. Figure 1 shows the main components of each of the areas and its utilization.

The UniversAAL project adopted a holistic approach. It includes runtime support for the execution of services and applications; it offers support for the development of new AAL services (development support); and it offers support for the collaboration among different communities (community support).

Runtime support consists of a set of software components that can be installed on different types of hardware; such hardware is typically adopted in several AAL scenarios. Common hardware devices are smart phones, laptops, and tablet PCs, but also include more powerful server installations. Runtime support is composed of different building blocks:

- The execution environment (EE), which provides a set of core functionalities such as discovery of devices and AAL services, communication capabilities among devices, context and user interaction management, security, remote control, and interoperability.
- The generic platform services, which provide some core services available for all the AAL services (or applications as their software parts) installed on top of the runtime support. Some notable examples are context awareness, [1], and support for security and personalization.

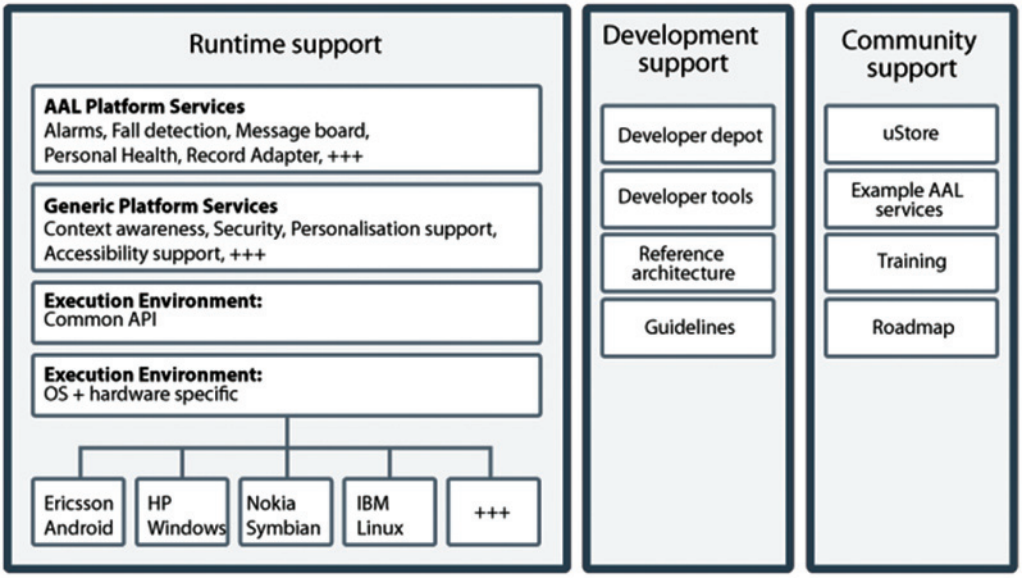

Figure 1: Overview of the UniversAAL Platform and Its Utilization.

– The AAL platform services, which are the set of core AAL services such as messaging, health record, and the fall detection service.

Development support consists of four building blocks that provide some resources for developers. Such resources aim to speed up the design and implementation of new AAL services. Development support provides the following resources:

– Guidelines: the documentation for developers. It provides the guidelines for developing a new AAL service on top of the UniversAAL EE. The guidelines include tutorials for using the main platform features, a description of platform APIs, and documentation for the integrated development environment developed in the project (called AAL Studio).
– Developer tools: a rich set of tools for developers to facilitate the creation of services or ontologies, to check the conformance of the code against some rules, to upload and deploy applications into the uStore, and other essential tools that speed up the development process. These tools are integrated to form a development environment called AAL Studio.
– Developer Depot: the server that hosts the resources needed for developers of AAL applications. It represents the starting point for developers; indeed, it includes the runtime support and development support resources listed above. The Developer Depot provides common repository functionalities such as upload, download, search, and removal of binary and source code resources depending on level of access.

Community support is a collection of resources for the end users (elderly persons and their care providers), including

– Roadmap: a set of guidelines for research and design in future versions of the AAL reference architecture (RA).
– Training: material for training new technical users. It includes documents, presentations, videos, and material for courses.
– Examples of AAL services: a collection of basic AAL services that can be used to show some features of the UniversAAL platform. The set of AAL services currently available includes the following: Agenda and Reminders, an intelligent reminder of daily events; Safety at home, a system that detects potential risks related to the home; AAL efficiency, an application for energy management; and Help when outdoor, a service that provides support in case the user gets lost or is in an emergency.
– uStore: the UniversAAL market where suppliers can publish the services they offer, and end users can search for, buy, and download the AAL services available. The uStore represents a helpful server to advertise AAL services and solutions to the final users or other service providers. The uStore has been designed to be accessible and personalized according to the user's profile; it allows managing complex services including software, hardware, and human resources in their complete life cycle. Moreover, the uStore can be directly accessed from the AAL Studio, to upload new AAL services.

4 Development Approach

This section describes the technical approach that was used to develop the architecture of the UniversAAL platform. The project placed a lot of emphasis on following a structured approach for this because the domain is complex, the variety of stakeholders is wide, the number of requirements is large, and the technical approaches of the "input" projects (see Section 2) somehow need to be consolidated. The approach we selected was to use ARCADE. It is a domain- and technology-independent architectural description framework for software-intensive systems [3]. We also made use of the IEEE 1471 standard for "Recommended Practice for Architecture Description of Software-Intensive Systems" [13].

As described in Reference [3], ARCADE is intended to support software developers to create, understand, and describe the architecture of a complex software system. The framework can be used in the design process

of the architecture of a new system, as well as for documenting an existing system. It provides the structure for documenting the architecture and for addressing quality-related concerns, reusability, and maintainability.

ARCADE bases its fundamental concepts on the IEEE 1471 standard framework "Systems and software engineering – architecture description." IEEE 1471 identifies the assets and the different views the system architecture should cover. In the first instance, it requires the identification of assets, stakeholders, and concerns. The assets are the artifacts (software and standards to be used), the legacy systems, the databases, and generally everything that will provide an input to the architecture.

With respect to the UniversAAL project, the assets come from the consolidation of those AAL projects that have been used as input to our platform. UniversAAL's stakeholders are the intended users and all those entities that could have an influence on the design of the system. UniversAAL's concerns identify the expectations each stakeholder has for the system. Assets, stakeholders, and concerns represent the building block to elaborate the models (use cases, diagrams, etc.), also named "views." Every view addresses a specific "viewpoint." The difference between viewpoint and view is that a viewpoint is a way of looking at a system, while a view is what you see when looking from the chosen viewpoint. A set of views eventually constitutes an architectural description.

In ARCADE, viewpoints are specified [3], whereas views are created when documenting some target system, the UniversAAL platform in our case.

The following sections show the design artifacts (documents and models) that constitute the UniversAAL design for AAL platforms. From the most abstract to the most detailed, the following are presented:

- The stakeholders and their concerns, as defined in the IEEE 1471 framework.
- The scenarios and the reference use cases (RUCs), which describe the typical functionalities of the AAL domain in an abstract and informal way.
- The architectural description, which provides a formal description of the platform from the reference model (RM) and RA, which defines a generic AAL platform, to the concrete architecture (CA), which defines the structure and the design decisions of the specific case of the UniversAAL platform.

The following sections describe each of these artifacts in detail.

4.1 Stakeholders and Concerns

The following categories of stakeholders have been defined for the AAL domain in general:

- Developers: those who develop AAL services. An AAL service can be in the form of hardware or software resources, or a combination of the two of them.
- Deployers: those who install AAL services in the end-user environment (such as health-care services installed at homes or at the hospital).
- End users: the main beneficiaries of AAL; elderly and disabled people as well as their caregivers, and their social environment.
- Assistance providers: the providers of services to the end users. Providers can buy hardware/software infrastructures and optionally the human support to deliver an AAL service.
- Authorities and supporters: those organizations that deal with the socio-economical and legal context of AAL and, therefore, they have expectations on, and indirectly shape, the AAL systems.

Stakeholders will always have some key concerns. In ARCADE, concerns are defined as "functional aspects that are considered to be of such importance that it should be treated separately and be specifically visible in the documentation" [3]. A concern may affect any viewpoint of the architectural description. The following classes of concerns are identified:

- Ethical concerns: deal with how AAL solutions address privacy, how transparent they are, and how they influence people's lives. Subcategories of these concerns are autonomy, freedom, privacy, transparency and consent, security, equal rights and non-discrimination, and termination without adverse consequences.

- Laws and regulations concerns: laws put constraints on the development and use of AAL services. Every stakeholder can have a different point of view with respect to the rules. Typical concerns under this category are fundamental rights, European regulations, and regional regulations.
- Quality-related concerns: stakeholders have concerns about the quality of AAL services and platforms. Typically, two categories are defined: quality of service and quality of experience, the former dealing with performance and the latter dealing with usability aspects.
- Use and deployment concerns: AAL applications should be easy to obtain, install, configure, and also to be maintained and extended. These concerns refer to the aspects of extensibility and reusability, maintainability, configurability and flexibility, deployability, and ability of easy interaction.
- Other concerns: other categories of concerns include accessibility and usability, handling a wide variety of fast-changing needs, business concerns, documentation, training and education, compliance to standards, safety, and energy efficiency.

We derive from these concerns the description of some relevant scenarios and use cases. The next section provides a more detailed description of scenarios and the use cases.

4.2 Scenarios and RUCs

The goal of this section is to describe the scenarios and the RUCs considered for the duration of the UniversAAL project. The scenarios and the RUCs have been defined, exploiting the experience gained from the input projects and from the experience of every partner of the UniversAAL consortium. First, we describe the scenarios and then we derive the RUCs.

A UniversAAL scenario describes the different stakeholders' point of views and how the stakeholders can benefit from different platform artifacts. Moreover, the scenarios provide an effective tool to show the UniversAAL benefits to a non-technical audience. We have identified the following 10 scenarios shown in Table 1. We refer with the term "view" to the ARCADE view definition (see Section 4).

The RUCs are intended for describing the domain to a technically oriented audience. RUCs are grouped according to the three building blocks of the UniversAAL project: runtime, development, and community support (see Figure 1). We have defined 18 use cases; each of them comprises a set of information. In particular, every RUC is described by means of a title, a description, the artifacts that are involved, the rationale behind the use case, what components of the UniversAAL architecture are involved, some real case example scenarios, and the supposed benefits for the stakeholders.

The UniversAAL RUCs have been categorized as follows:

Category 1: A platform for creating innovative and commercial grade AAL services;
Category 2: A platform for creating a market place for AAL services;
Category 3: A platform for supporting developers in innovating the AAL market.

The full list of the RUCs is reported in project deliverable D1.1 [21].

4.3 Reference Architecture Requirements

We define the reference architecture requirements (RARs) for two main reasons. First, the RARs drive the design of the UniversAAL RA. To this purpose, the RARs target mainly to the software architects, to define the guidelines for the design of a specific AAL architecture. Second the RARs help in analyzing the RA. Indeed, RARs can help in understanding the scope of the RA, the design decisions, and the limits or boundaries of the RA. In the following, we report the RAR categories; the full list of RARs can be found in project deliverable D1.2 [22]. The complete list of RARs contains 29 high-level requirements (related to the RA) and 115 techni-

Table 1: The Scenarios.

Scenario	Brief description
IT developer view	It describes how a developer makes use of the runtime support and especially the development support. In particular, how the code is generated, how development tools are used, and how the final application is made available on the uStore.
Elderly person view	It describes the point of view of the final user, e.g., an older adult. Such a user buys an AAL application from the uStore to obtain some form of assistance in daily living activities.
Medical specialist view	It describes the point of view of the caregivers. In particular, the scenario describes how the AAL services can control the patients, how to interact with them, and how to fire alerts about risky situations.
Relative view	It provides the point of view of the informal caregiver. The scenario explains how the caregiver can select an AAL application from the uStore according to the caregiver's feedback and the user comments. The scenario also describes how caregivers can purchase applications.
IT provider view	It shows the point of view of the deployment companies, how they can integrate existing solutions through the uStore and sell the service for installing, configuring, and testing applications to, e.g., insurance companies.
Maintenance provider view	It describes the point of view of a deployer and maintainer of AAL services. It focuses on how a technician can learn about the platform with development support; how he/she can install, test, and configure AAL solutions; and how to adapt the system and to replace the malfunctioning parts.
Retirement residences view	It shows how an assistance provider can evaluate and acquire AAL services. It shows how the provider can benefit from the AAL service and how the feedbacks on the AAL service are returned to the solution provider.
Health insurance company view	It is focused on authorities' and supporters' point of view. It shows how an insurance company buys a solution from a provider by using the uStore and how the benefits of the solutions are tracked, thanks to logs and remote questionnaires.
Social service department view	It shows the point of view of a public authority, particularly how they benefit from promoting open platforms for the assistance of elderly people, how they learn about the platform thanks to the development support, and how the experience is shared with others through the uStore.
Student view	In this scenario, we consider a student as a young developer. The scenario shows how a young developer can learn about the UniversAAL platform, how to reuse the existing AAL services and improve them, and how to gather experience for research in the field.

cal requirements (related to the CA) in 10 different categories that were mapped to concerns and RUCs. The UniversAAL RARs are categorized as follows:

Category 1: Context information and context management;
Category 2: Transparency, privacy, and security;
Category 3: Support for designing human-computer interaction;
Category 4: Service orientation;
Category 5: Interoperability with legacy and remote components and systems;
Category 6: Configuration, personalization, maintainability, and scalability;
Category 7: EE and component and application container for heterogenic devices and operating systems;
Category 8: Shared communication infrastructure;
Category 9: Reliability and safety;
Category 10: Community and developer support platform.

5 The Reference Model

The RM establishes a common understanding of the AAL domain. It is also the starting point for the design of the UniversAAL architecture.

The main characteristic of an RM is that it "consists of a minimal set of unifying concepts, axioms, and relationships within a particular problem domain, and it is independent of specific standards, technologies,

implementations, or other concrete details" [4]. Thus, the UniversAAL RM provides simple definitions of the key concepts that make up the AAL domain, and shows how these fit together. This provides a common basis for understanding the AAL domain as a whole: a common "language." The RM was produced as part of a consolidation process from the input projects; in particular, it defines a new model by combining the concepts from the input projects.

The RM uses a hierarchy of concept maps [26] to provide a structured, but easily understood, way of showing how the different concepts relate to each other. Each concept map is supported by some text providing simple definitions of each new concept introduced in it. Some concepts are expanded in further detail in lower-level concept maps. The top-level concept map is called the "root" concept map. We include it here to illustrate the concept map technique, and to define the highest-level concepts. The full set of concept maps is available in project deliverable D1.3 [23].

The root concept map of AAL RM is depicted in Figure 2. It presents the top level of the AAL domain. The concepts identified are defined as follows:

- AAL stakeholder: an individual, team, or organization (or classes thereof) with interests in, or concerns relative to the AAL domain (e.g., service providers, end users, authorities etc.).
- AAL service: a combination of software, hardware, and human resources that support people who need some assistance in carrying out everyday activities.
- AAL RA: provides a common/standardized view of software architectural elements (building blocks) in the AAL domain.

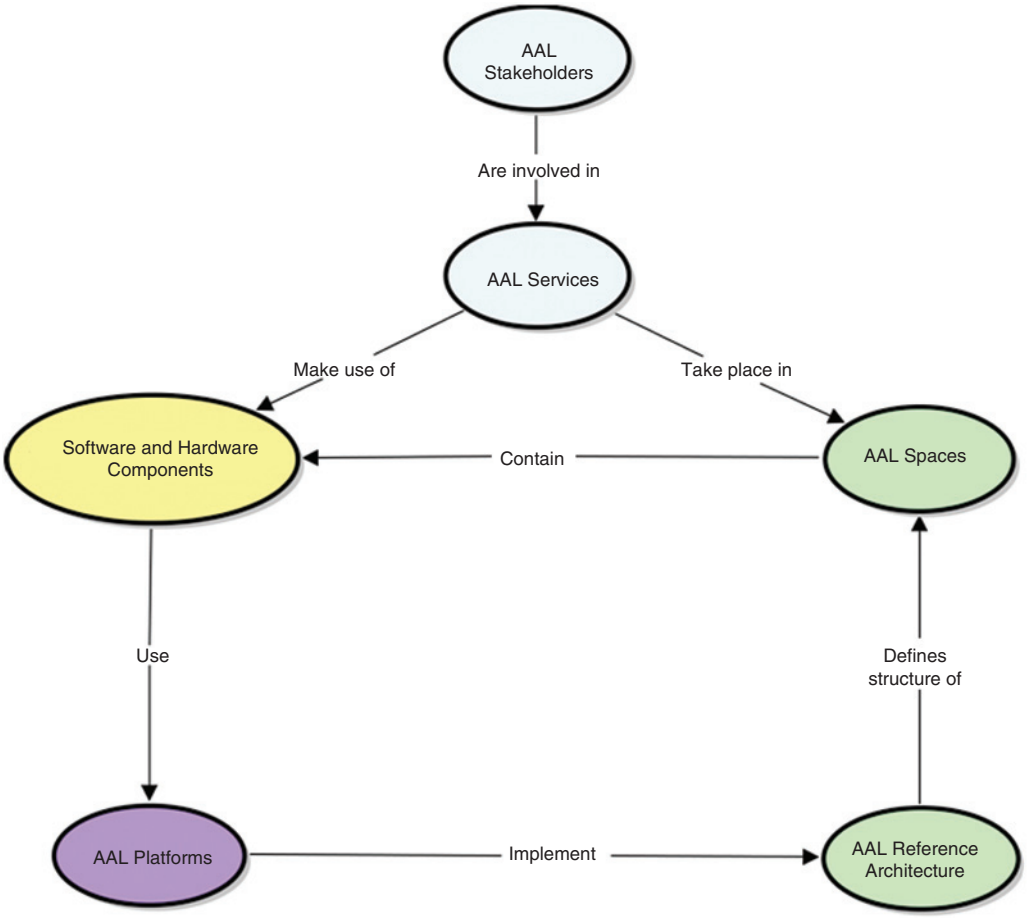

Figure 2: Root Concept Map of the AAL Domain.

- AAL platform: a set of hardware and software artifacts (standardized and compliant with the AAL RA) that help developers and service providers with the provision of AAL services (at all stages in the life cycle, including design, development, deployment, and execution).
- AAL space: environments where people may need assistance (e.g., homes, airports, railway stations, supermarkets, and cars) with defined boundaries of hardware, software, and human resources in the context of, e.g., access control, multimodal user interfaces, etc.
- Software and hardware component: one or more pieces of computer software and/or special devices (sensors, displays, switches, controls for domestic appliances, etc.) that, together with any relevant human resources, are used to provide AAL services.

The AAL domain is all about the provision of AAL services to end users (e.g., assisted persons and/or caregivers). AAL services, which take place in an AAL space, make use of the hardware and software components embedded in it. The cooperation between such distributed components in an AAL space is facilitated by AAL platforms that implement the AAL RA to provide for resource sharing, and let users experience an integrated world easy to interact with based on natural communication.

6 The Reference Architecture

The RA models the abstract building blocks in the AAL domain, independently from the technologies, protocols, and products that are used to implement the domain (SOA-RA) [12]. It is consistent with the RM defined in Section 5.

The RA differs from the RM in that the latter describes the important concepts and the relationships in the domain focusing on what distinguishes the elements of the domain, whereas the RA further elaborates the model, to show a more complete picture that includes what is involved in realizing the model.

The design of the RA is the result of the analysis of the RUCs, the reference requirements, and the RM. We followed two different parallel processes for designing the RA:

- The top-down process: the RA is seen as the "natural" extension of the RM. This approach assures high abstraction, a minimum gap of abstraction with respect to the RM, and, potentially, a model suitable for a larger audience. The main limitation of this process is the partial lack of a technical focus. Such limitation makes the RA more difficult to connect to the real implementation (also named the concrete architecture). The top-down process resulted in a set of collaboration diagrams that show the relationship among the components of the RM. Moreover, this process also provides a service capability analysis by means of the service-oriented architecture modeling language (SOA-ML) notation.
- The bottom-up process: an abstraction process that takes into account the concrete implementation of the input projects. In particular, this process has identified the commonalities and the weaknesses of the input projects. In this case, the result is a set of abstract building blocks that represent the structural aspects of the platform. We have adopted the system decomposition model defined by the ARCADE [3] specification.

The next two subsections report the results we obtained both with the bottom-up and top-down processes.

6.1 Results of the Top-Down Process: Service Capabilities Supported by the RA

The main stakeholders and the concerns have been presented in Section 5. The goal of this subsection is to describe how such concerns are addressed in the form of services exchanged among the stakeholders. The result of the top-down process is named the RA services (RASs).

The value network of the AAL domain is mapped to an SOA-ML service architecture diagram in Figure 3, where the "participants" are the stakeholders and the "service contracts" denote the services exchanged

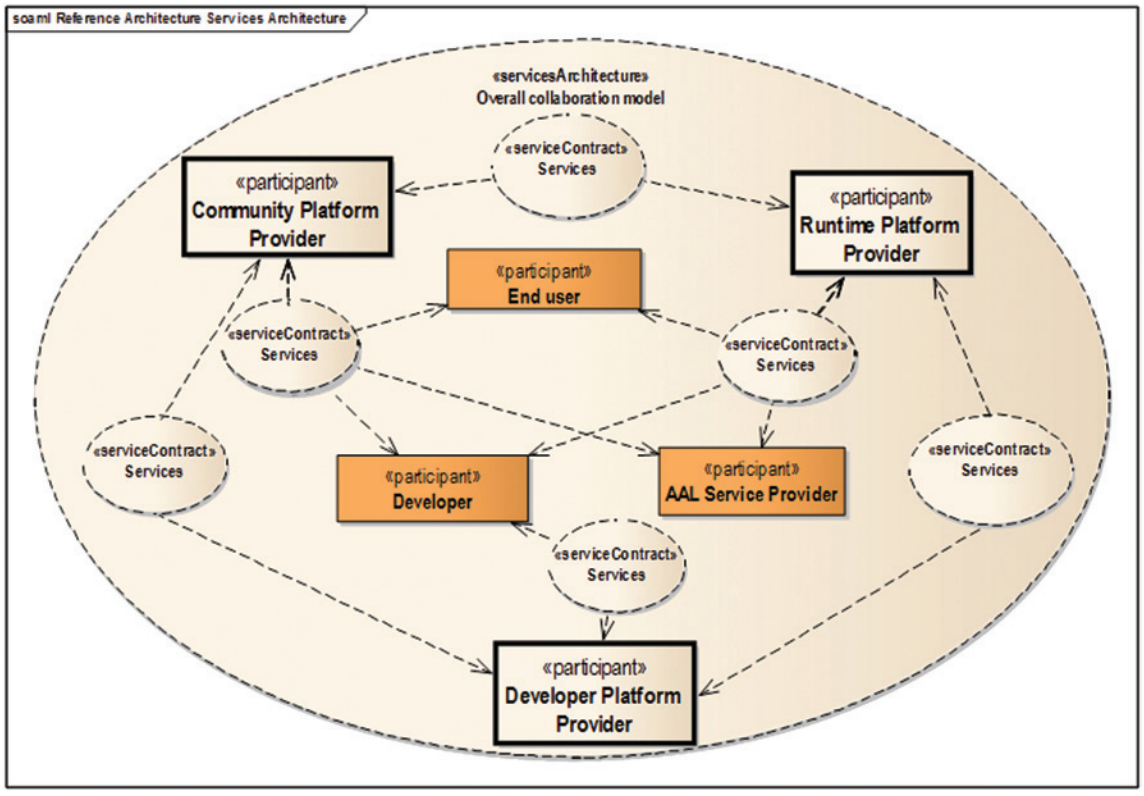

Figure 3: Service Architecture for AAL Main Stakeholders and Their Service Capabilities.

among the stakeholders. Client stakeholders are shown in orange (gray in black-and-white print), and platform stakeholders are shown as rectangles with thick borders.

According to SOA-ML [20], services exchanged among stakeholders are modeled by using a specific type of collaboration called service contract. In Figure 3, each service contract is a placeholder for all the RASs exchanged between the two parties involved.

For instance, the example in Figure 4 illustrates the services that the runtime platform provides to the community and developer support platforms. The explanation of Figure 4 is given in Table 2.

6.2 Results of the Bottom-Up Process: Abstract Building Blocks of the RA

In this subsection, we provide the model for the runtime support platform (RSP) only. The RSP comprises the building blocks that allow the AAL applications to be executed in the space of the end user (Table 3). For a better understanding of the runtime, it is presented in a relaxed layered model, where each layer can access interfaces provided by layers below it (Figure 5).

7 The UniversAAL CA

As discussed in the previous section, RA is technology-independent model. It can be used to design and implement different concrete implementations. The implementation of the RA is named the CA. The UniversAAL project has defined one open-source CA as the reference implementation of the project.

The main components of the CA are the Developer Depot, the AAL Studio, the UniversAAL control center (uCC), the uStore, and the EE. Figure 6 shows an overview of such components.

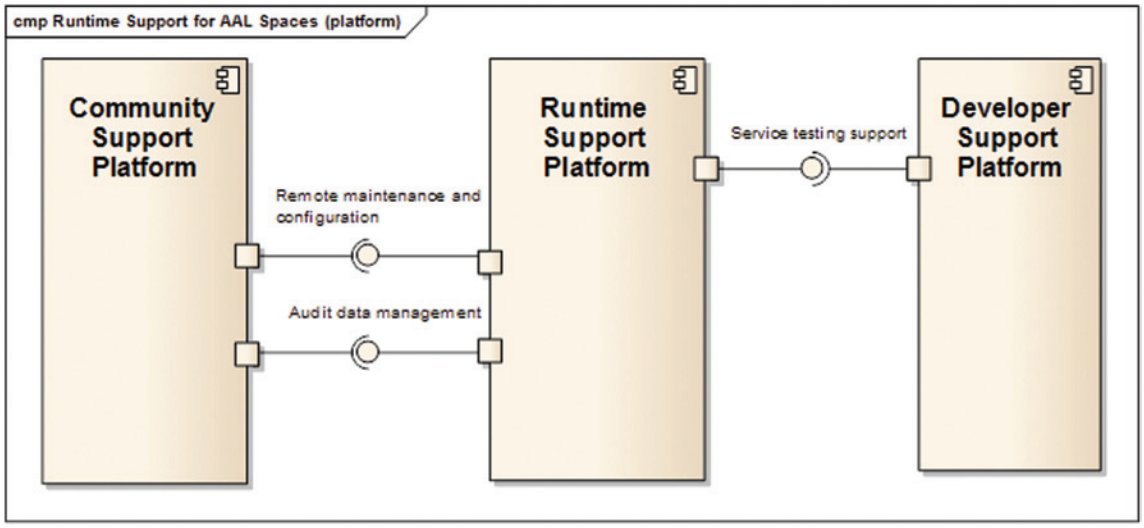

Figure 4: Platform-to-Platform Services Provided by the Runtime Support Platform.

The Developer Depot provides the resources required by a developer to start developing AAL applications. Some examples of resources are the AAL Studio development tools, the binary and source code for the execution platform, and the AAL service examples together with the platform documentation.

The AAL Studio is a customized Eclipse-based development environment for AAL applications. It provides a set of development tools to support the developers during all parts of the development process: design, implementation, test, and deployment. AAL Studio and Developer Depot are bound together. Indeed, we have designed and implemented a specific AAL Studio plug-in to upload or download resources to the Developer Depot.

The UniversAAL EE is a set of software components that are installed on a device or AAL-aware node (such as a smart phone, a PC, or a tablet). The goal of the EE is twofold: first the EE installs updates or removes AAL services (AAL service life cycle management [16]); second, the EE executes one or more AAL services installed on the device.

More precisely, the management of the AAL service life cycle is implemented with the uCC. The uCC is a well-integrated user interface that provides three main features: request installation, configuration, and customization. The end user can use the uCC user interface to select the AAL service available on the uStore and request the installation on its device. End users can also configure and customize the AAL service according to their needs.

The last component of the CA is the uStore. The uStore allows the end users to find and to purchase an AAL service. The uStore provides two interfaces: a Web-based interface and the uCC interface. Hence, the end users can browse the AAL services available on the uStore both for a standard Web browser and from the uCC.

Table 2: Reference Architecture Services (Example).

RAS id	Service	Provided to	Description
RAS#1.1	Remote maintenance and configuration	Community support platform	To allow remote configuration, management, and maintenance of software installed in AAL spaces
RAS#1.2	Audit data management	Community support platform	To allow acquiring usage data about service usage in AAL spaces
RAS#1.3	Service testing support	Developer support platform	To support sandbox testing of services before real-world deployment

Table 3: The Building Blocks.

Building block	Description
Middleware	This building block is the core of every runtime platform; its goal is to hide distribution, supporting seamless connectivity, and to allow the communication among all the elements deployed in the AAL system. It acts as a broker between the communicating parties and hides the heterogeneity of the technologies adopted. The data representation framework belongs to the basics of the middleware software. The selection of a unified data representation framework makes it possible that data from diverse domains can be exchanged using the same middleware. Reliability has to be provided at middleware level and above, to provide the ability of a system or a component to perform the required services under specific conditions for a specified period of time. This includes the failure probability, and the system availability and maintainability of a complex entity of interconnected components.
Context management	This building block provides for adequate communication (both push- and pull-based) between providers and consumers of context data. Data are expressed according to a common model.
Service management	The service management building block facilitates service-based interoperability. It handles service registration, and act as brokers between service providers and the consumers to handle the service requests. As an example, service management finds a match between a received service request and the matching service providers that are available. Moreover, it provides service composition and orchestration features.
User interaction (UI) management	The UI management building block addresses the challenges related to the explicit interaction between an AAL space and its human users. It provides independence of applications from the concrete I/O infrastructure available in AAL spaces, as the latter might differ in its occurrences considerably. It defines a framework for capturing user input and presenting system output to human users. The UI management is documented as a Publicly Available Specification by the International Electrotechnical Commission (IEC) with the reference IEC/PAS 62883 Ed. 1.0 and title "The UniversAAL Framework for User Interaction in Multimedia Ambient Assisted Living (AAL) Spaces."
Local device discovery and integration (LDDI)	This building block facilitates the integration of heterogeneous hardware devices with the UniversAAL platform. Indeed, several AAL applications often require special-purpose devices. Such devices are packaged without the possibility of manipulating the device. Hence, it is required as a specialized building block that is able to integrate such devices in a seamless way.
Remote interoperability	This building block is responsible for managing platform tasks in the context of interactions between AAL spaces and the outside world. Some common AAL applications include the provision of remote assistance by service providers, remote monitoring, remote access to services available within an AAL space, importing services from the outside world, and communication between AAL spaces.
Security	The security block provides trust, privacy awareness, and access control. An artifact that joins an open system like AAL systems must possess a certain level of trustworthiness and, hence, the question here is about the right balance between the openness of the system and the level of control needed. At the level of joining the system, the level of trust needed is that the artifact at hand has not any malicious intention, and it respects the game rules assumed in the system.

7.1 The UniversAAL EE

The goal of this section is to provide an overview of the features implemented by the EE to execute AAL services on a device.

The EE is composed by several building blocks; the rest of this section describes the functionalities they provide.

7.1.1 Middleware

The middleware acts as a broker between the AAL-aware nodes. In particular, it handles the exchange of messages among nodes and hides the heterogeneity of the hardware devices. It is decomposed by different parts:

– Container: implements the runtime environment for the whole platform. The container provides facilities for managing the life cycle of the software artifacts. As an example, the container allows to install, uninstall, start, or stop an artifact, and to resolve its dependency at runtime. Most of the source code is

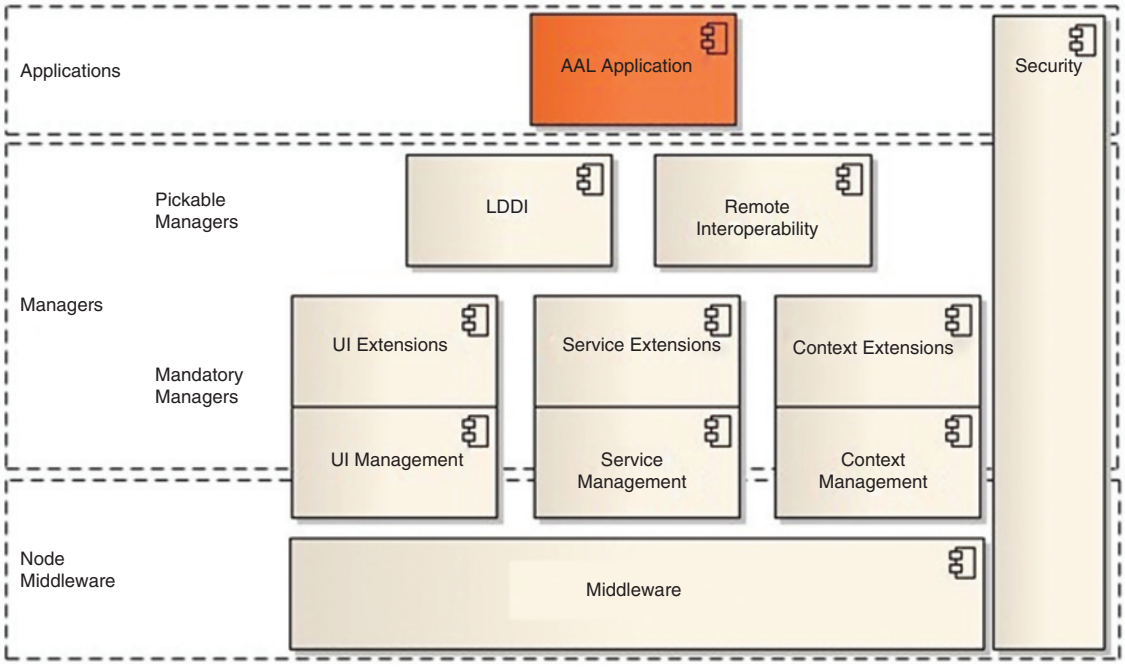

Figure 5: Consolidated Decomposition Model of the Runtime Support Platform at the Level of Abstract Building Blocks.

container independent and can be ported to different containers; as concrete examples, we used Android and OSGI (OSGI Alliance, http://www.osgi.org, last accessed: 24 February, 2014) (i.e., Apache Felix/Karaf and OPS4J Pax).

– Discovery and peering (D&P): the main goal of D&P is to discover and to configure dynamically the AAL-aware nodes, [6, 10]. The UniversAAL CA uses the SLP (Service Location Protocol, RFC 2608) discovery protocol, [9], and the JGroups (http://www.jgroups.org/, last accessed: 24 February, 2014) communication protocol for the exchange of data between nodes.
– Communication: defines how data are transported from the sender to the receivers. The data exchange is implemented according to different communication paradigms: event-based and request/reply modality.

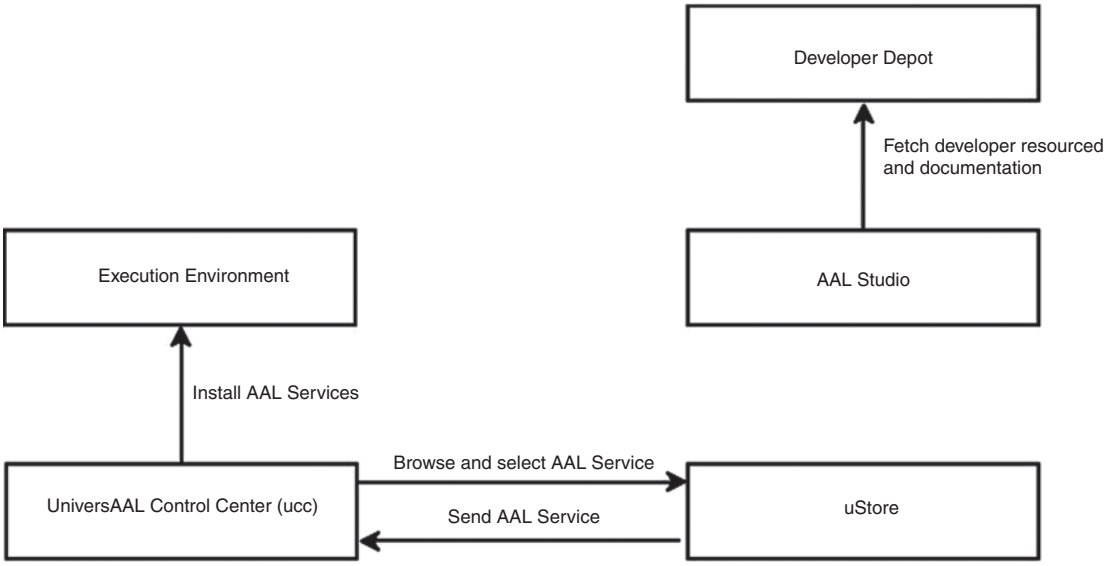

Figure 6: Overview of the Components of the UniversAAL CA.

This component provides the base for the concrete broker used by service management, context management, and user interaction framework.
– Data representation model: defines a unified model for representing data and, thus, provides the foundation for interoperability between components of and across different loadable nodes. It does not describe which data should be handled; rather, it gives a general way of how to represent data so that it is possible to save, transmit, and transform any semantic information that is shared within the system. The functionality can be divided into the following two tasks: (i) representation of data, to provide interoperability between different components of the same loadable node independent of the actual content; (ii) (de-) serialization, to provide interoperability between different loadable nodes by transformation to and from a unique representation that can be transmitted to other nodes in the system; thus, it hides heterogeneity. It was decided to use RDF (Resource Description Framework) and OWL (Web Ontology Language) for representation of data.

7.1.2 LDDI (Local Device Discovery and Integration)

The LDDI building block implements a sensor and abstraction layer (see Reference [5]) to integrate sensors and actuators in the UniversAAL platform. The CA supports a set of sensing technologies: Konnex [17], ZigBee [8], Continua devices via Bluetooth, and a prototypal integration of FS20.

7.1.3 Context Management

This building block deals with the management of the context information (i.e., any kind of information that can represent the situation of the environment and the users). The context information can be exploited by the AAL services, to adapt themselves to the environment or the end-user status. Context management includes the following building blocks:
– Context management: provides for adequate communication (both push and pull based) between providers and consumers of context data, according to a common model for representing such information. Hence, it provides an appropriate brokering for a push-based communication between providers and consumers of context, stores the context information in a context storage, provides access to the storage, and, lastly, reacts to some relevant events that occur in the AAL space. The CA is based on the OpenRDF Sesame framework for context management.
– AAL space manager tools: manages the user preferences, so that the every user can have a personal profile. The AAL space manager tools allow for creation, access to, manipulation, and deletion of the various kinds of profiles (e.g., device profiles, user profiles, AAL space profiles, etc.). It consists of the profiling server and the AAL space server.

7.1.4 Service Management

This component deals with the definition and implementation of the UniversAAL service infrastructure. It is composed of the following two building blocks:
– Service management: handles the registration of AAL services and acts as broker between service providers and service consumers. In particular, it implements a mechanism for matching the service requests with the services available. The CA is based on OWL-S (Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, last accessed: 24 February, 2014) for representation of services.
– AAL space orchestrator (ASOR): provides support for service composition using an appropriate scripting language. ASOR extends the composite processes of OWL-S by utilizing the whole functionality available in AAL spaces.

7.1.5 User Interaction (UI) Management

The UniversAAL platform offers a UI framework for user interaction that separates the content exchanged between the user and the application from its actual presentation. The UI framework [14] consists of the following components:

- UI bus: provides distributed methods for accessing UniversAAL instances in relation to the user interaction.
- UI model: contains ontological representations of the most common user interaction components, i.e., the building blocks for sending messages to end users and retrieving feedbacks that are shared among the bus peers. It enables a modality-neutral description of user interfaces, inspired by the standard W3C XForms.
- UI caller: represents an application; it sends requests to the UI bus and manages responses from the user.
- UI callee: also called UI handler: presents the request to the end user. Each UI handler uses a different set of modalities, e.g., GUI, speech, or gesture.
- Dialog manager: used by the UI bus to manage the system-wide coordination of each user's dialog through dialog priority and adaptation.
- Resource manager: manages various media resources (e.g., images) that can be used by UI handlers.

7.1.6 Remote Interoperability

The remote interoperability building block implements the interactions between the AAL space and the world outside the space. This includes the interoperability between humans and software residing in the AAL space, on one side, and humans and software outside the space, on the other side. It is composed of the following building blocks:

- AAL space gateway: addresses the challenges related to the interaction between AAL spaces and the external world. The functionality can be divided into two components: (i) providing the necessary interfaces for accessing AAL space resources from outside the AAL space and importing Web services from the outside world; (ii) providing reverse access from AAL spaces to the outside world utilizing Web service technologies. Additionally, it provides connectivity between two different AAL spaces for remotely accessing bundles and data of an AAL space.
- Remote administration: contains tools that allow the remote access and management of an AAL space (and its nodes) that can be offered either internally or outside the space, to manage the data by the service providers.
- Remote interoperability tools: tools that provide support for interoperability issues. Currently, only one tool is available, that is the Ontology Repository; however, this tool is not executed in the runtime platform but is hosted in a separate server and accessed by the platform at the AAL space. The objective of this tool is to provide storage, access, visualization, and usage of ontologies specifically implemented for the UniversAAL interoperability requirements. It is based on NCBO Bioportal (https://bioportal.bioontology.org, last accessed: February 24, 2014).

7.1.7 Security

The security component provides for trust, privacy awareness, and access control. One of the most important reasons for the above mechanism is the protection of the privacy of humans. As a notable example, the security component aims to prevent unwanted disclosure of health details, personal preferences, habits, and lifestyle leading to discrimination. It includes the following components:

- Container security: makes use of the security mechanisms of the underlying container, e.g., Java, OSGI, and Android.

- Authentication framework: defines a general framework to authenticate end users, with a concrete example for authentication with login ID and password.
- Session management: manages the validity of an authentication over a location or device at a given moment in time.
- Consent (or privacy policy) manager: stores the patient consent information according to HL7 CDA R2 Consent Directives.
- Encryption: encryption of data that is exchanged between nodes and IHE Document Encryption for the encryption of data at rest.
- Policy-based permission matchmaking: defines and tests all operations an application performs on the three brokers. As compared with the node-level container security, this component provides security on space level.

8 Evaluation Process

This section describes the evaluation process that we followed during the UniversAAL project with what regards architectural aspects of the platform. Several evaluations were performed during the execution of the project, and feedback has been collected on several occasions. Some of the more important results are presented in the following.

8.1 Evaluation of the Scenarios and the RUCs

Scenarios and RUCs have been evaluated in four rounds. In a first round, a questionnaire was sent to project members to assess the clarity of the description and the feasibility of the presented concepts within the project's lifetime. Also, suggestions for improvements were collected in free text fields. The questionnaire was answered by 16 people including software architects and developers. In the same round, personal interviews were set up with experts external to the project. In total, six groups of stakeholders were involved, with 14 users in three countries (Austria, Denmark, and Spain). The results of this evaluation round indicate a general satisfaction concerning the intention of the platform, although many suggestions for improvement were also provided and some unrealistic scenarios were simplified.

A second evaluation round was conducted about RUCs. In this evaluation round, the RUCs were compared with the components of the RA to assess how much the design of RUCs and components matches, and to identify possible defects and lack of coherence. The RUCs were analyzed by the developers and the software architects of the project; in particular, eight persons joined the evaluation. The evaluation was formalized by means of a spreadsheet containing the RUCs on the rows and the components of the architecture as columns. In each cell, experts had to insert a value, from 0 to 10, expressing how much the component's functionalities supported the RUC. In another sheet, comments were collected about the components or the use cases. Thanks to this evaluation, the level of compliance between use cases and actual implementation was improved, both by lowering the requirements in some use cases and by fixing better functional requirements in the architectural design.

After the improvements were completed, a third evaluation round was launched. RUCs and scenarios were reviewed by some members of the UniversAAL project in a detailed analysis. Many small issues were identified related to linguistic deficits and partly missing explanations. Finally, a fourth evaluation round was conducted only on the scenarios with external experts. The objective was to validate the way UniversAAL provides solutions to help solve the problem of independent living through the analysis of the scenarios. A questionnaire was submitted to five developers, two service providers, eight assistance providers, one informal caregiver, one formal caregiver, and three assisted persons from three countries. The study showed that the scenarios are generally well written and positively received by the participants. The overall idea of UniversAAL, the concept of providing tools for the development of AAL services, and the idea of the uStore

as a shop for AAL solutions were very well received. Some shortcomings were identified about privacy and business models.

8.2 Evaluation of the RM and RA

The RM and the RA were evaluated with relevant stakeholders by collecting experts' opinions. The idea of the evaluation was to get insight into how well the UniversAAL RA fit with other AAL platforms. To this purpose, the full set of the RASs was sent to other AAL platform providers who were asked to indicate whether such a type of service is represented in their platform. Comment areas were provided to gather further details. As an adjunct purpose, we collected concrete suggestions about the RM, on possible improvements, as well as whether there were important concepts missing in the model.

Six experts from both industry and research institutions provided feedback. Four alternative AAL platforms were mapped to the RASs of RA, and the analysis of the results showed that around 36% of RASs were also implemented in competitors' platforms. More specifically, it seems that the runtime support category is the most commonly represented, indicating that development support and community support are secondary or ignored in the other platforms. Looking at the comments, it was deductible that experts considered some RASs too specific for a platform or of responsibility of the application developers. Summarizing the responses on the RM, it was possible to see that its coverage of the domain was considered as high, and no major concepts were missing. Most of the received feedback was on minor issues and some general concerns, e.g., about the complexity of the platform.

8.3 Evaluation of the Reference Requirements

The reference requirements have been internally analyzed by project members to check to what extent the requirements are being met in the RA and also in the CA and the reference implementation. One SW architect and eight developers were asked to quantify the level of conformance between the requirements and the CA and to add free comments. The analysis of the results showed that, for the most part, the requirements are suitable for the domain and the project itself, the level of details provided by the technical requirements is sufficient, there is a good match between the RA and the reference requirements, and the conformance between the design and implementation is mostly good.

9 Summary of Achievements

The design framework presented in this article proved suitable for the creation of a complex platform like the one of UniversAAL. The development of the example services showed that having a platform allows avoiding programming several common functionalities of distributed, ambient technologies, and easily program services without having to cope with technology-dependent aspects like networks, sensor protocols, non-interoperable data models, different kinds of user interfaces, and adaptation to disabilities, among others.

UniversAAL had a strong focus on standardization activities. In the following, activities related to the architecture and design are highlighted. The architectural models presented hereby aim at becoming an official standard. At the moment, some efforts are in progress for standardizing some aspects of the UniversAAL reference design. An example is the participation in and contribution to IEC SG 5 on AAL, which aims to manage and coordinate AAL standardization work in IEC technical committees to establish and achieve interoperability and interconnectivity of AAL systems [25]. It provides the opportunity to standardize AAL solutions from UniversAAL and other projects. The UniversAAL framework for user interaction has obtained the official IEC PAS status for its specification. It is documented as a Publicly Available Specification by the International Electrotechnical Commission (IEC) with the reference IEC/PAS 62883 Ed. 1.0 and title "The

UniversAAL Framework for User Interaction in Multimedia Ambient Assisted Living (AAL) Spaces" [11]. In the same technical committee (IEC TC100), some of UniversAAL's use cases are an input to the standardization of use cases related to AAL (IEC/TR 62907/Ed. 1), IEC, IEC/TR 62907 Ed. 1.0, *Use cases related to ambient assisted living (AAL) in the field of audio, video and multimedia systems and equipment*.

UniversAAL has also contributed to data protection by specifying an IHE Document Encryption profile for data in the health-care and AAL domain. Consequently, the specification was implemented and feedback was provided to IHE. It has obtained the trial implementation status and passed the IHE Connectathon validation. Furthermore, UniversAAL has contributed to the Zigbee API and Device Abstraction Layer in the OSGi Alliance resulting in the requirements definitions (RFP 142 and RFP 147) and resulting in the technical solutions (RFC 192 and RFC 196) expected in 2014.

For exploitation through standardization of the RM and RA, a detailed plan has been elaborated. The key idea is to support various stakeholder groups in different activities, ranging from the researchers and developers (baseline), to authorities (regulatory, policy), to service providers (process, cope), and finally to follow-up standardization activities (technical).

The UniversAAL project was completed in January 2014, and a fully operational version of the platform here described was produced. All technical results (including all software) and documents are openly available via the project's Web site at www.universaal.org. The platform is now being applied in large-scale pilots being conducted in the ReAAL project (see http://www.cip-reaal.eu/home/).

Funding: This work has been cofunded in the framework of the UniversAAL project: UNIVERsal open platform and reference Specification for Ambient Assisted Living, FP7 Programme, FP7-ICT-2009-4-Objective 7.1b "Open Systems Reference Architectures, Standards and ICT Platforms for Ageing Well," contract no. 247950. Web site: www.universaal.org.

Bibliography

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles, Toward a better understanding of context and context-awareness, *Handheld and Ubiquitous Comput. Lect. Notes Comput. Sci.* **1707** (1999), 304–307.
[2] Amigo Short Project Description, January 16, 2004, http://www.hitech-projects.com/euprojects/amigo/publications/IST-004182%20Amigo-IP%20short%20project%20description.pdf. Accessed 20 January, 2015.
[3] ARCADE, *An Open Architectural Description Framework*, http://www.arcade-framework.org. Accessed 20 January, 2015.
[4] M. A. Boston, *Reference Architecture Foundation for Service Oriented Architecture*, Committee Draft 02, http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf. Accessed 20 January, 2015.
[5] S. Chessa, F. Furfari, M. Girolami and S. Lenzi, SAIL: a Sensor Abstraction and Integration Layer for Context Aware Architectures, in: *34th EUROMICRO Conference – Special Session on Software Architecture for Pervasive Systems (SAPS)*, pp. 374–381, 2008.
[6] W. K. Edwards, Discovery systems in ubiquitous computing, *Pervasive Computing, IEEE* **5(2)** (1998), 70–77.
[7] GENESYS project, http://www.genesys-platform.eu/. Accessed 20 January, 2015.
[8] K. Gill, Y. Shuang-Hua, Y. Fang and L. Xin, A zigbee-based home automation system, *IEEE Transact. Consum. Electron.* **55(2)** (2009), 422–430.
[9] E. Guttman, C. Perkins, J. Veizades and M. Day, *Service Location Protocol, Version 2*, IETF RFC 2608, 1999.
[10] M. Handley and V. Jacobson, *SDP: Session Description Protocol (RFC 2327)*, IETF.
[11] IEC, IEC/PAS 62883 Ed. 1.0, *The universAAL Framework for User Interaction in Multimedia Ambient Assisted Living (AAL) Spaces*.
[12] S. Jones, Toward an acceptable definition of service [service-oriented architecture], *Software* **22** (2005), 87–93.
[13] M. W. Maier, D. Emery and R. Hilliard, Software architecture: introducing IEEE Standard 1471, *Computer* **34** (2001), 107–109.
[14] G. A. M. Medrano, D. Salvi, P. A. Jimenez, A. Grguric and A. M. T. Waldmeyer, Separating the content from the presentation in AAL: the universAAL UI framework and the Swing UI Handler, in: *4th International Symposium on Ambient Intelligence, ISAmI*, 2013.
[15] OASIS Project, http://www.oasis-project.eu/. Accessed 20 January, 2015.

[16] R. Ram, F. Furfari, M. Girolami, G. Iban Ez-SaNchez, J. LaZaro-Ramos, C. Mayer, B. Prazak-Aram and T. Zentek, UniversAAL: provisioning platform for AAL services, in: *ISAmI – 4th International Symposium on Ambient Intelligence*, pp. 105–112, Springer International Publishing, Switzerland, 2013.
[17] M. Ruta, F. Scioscia, E. Di Sciascio and G. Loseto, A semantic-based evolution of EIB Konnex protocol standard, in: *IEEE International Conference on Mechatronics (ICM)*, pp. 773–778, 2011.
[18] A. Sixsmith, S. Muller, F. Lull, M. Klein, I. Bierhoff, S. Delaney, P. Byrne, R. Savage and E. Avatangelou, *A User-Driven Approach to Developing AAL Systems for Older People: The SOPRANO Project*, 2010.
[19] M. Tazari, F. Furfari, J. P. Lazaro Ramos and E. Ferro, The PERSONA service platform for AAL spaces, in: *Handbook of Ambient Intelligence and Smart Environments (AISE)*, pp. 1171–1199, Springer, USA, 2009.
[20] I. Todoran, Z. Hussain and N. Gromov, SOA integration modeling: an evaluation of how SoaML completes UML modeling, in: *15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, pp. 57–66, 2011.
[21] UniversAAL Deliverable D1.1, http://universaal.org/images/stories/deliverables/. Accessed 20 January, 2015.
[22] UniversAAL Deliverable D1.2, http://universaal.org/images/stories/deliverables/. Accessed 20 January, 2015.
[23] UniversAAL Deliverable D1.3, http://universaal.org/images/stories/deliverables/. Accessed 20 January, 2015.
[24] VAALID project, http://www.oasis-project.eu/. Accessed 20 January, 2015.
[25] VDE, The German AAL Standardization Roadmap, http://www.dke.de/de/std/AAL/documents/german%20aal%20standardization%20roadmap.pdf, 2012. Accessed 20 January, 2015.
[26] J. J. Villalon and R. A. Calvo, Concept map mining: a definition and a framework for its evaluation, in: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 3, pp. 357–360, 2008.