SEVENTH FRAMEWORK PROGRAMME SEVENTH FRAMEWORK PROGRAMME: PRIORITY 7.1B LARGE SCALE INTEGRATING PROJECT (IP)	IP project number 247950 Project duration: February 2010 – February 2014 Project coordinator: Joe Gorman Project Coordinator Organisation: SINTEF, Norway Strategic Objective: 7.1.b website: www.universaal.org UNIVERSAAL Universal Open Architecture and Platform for Ambient Assisted Living

Document Type: "Deliverable:" Item Appearing in "List of Deliverables in DoW with delivery date shown in bold "Supplementary Report" As "Deliverable", but delivery date *not* shown in bold. These documents are formally internal to the consortium, but can be delivered on request.		
		Project Deliverable, with independent sub-parts. *Each sub-part forms a coherent whole in its own right, and has been edited and reviewed independently. The sub-parts are integrated in this document, to form the deliverable as a whole.*
		Project Deliverable (single document, no sub-parts).
	X	Sub-part of a Project Deliverable

Document Identification			
Deliverable ID:	**D1.3-E**	Part title:	**Part III: The universAAL Reference Architecture for AAL**
Release number/date:		V1.0 18.11.2013	
Checked and released by:		Sergio Guillen/ITACA-A	

Key Information from "Description of Work" (from the Contract)	
Deliverable Description	Specification of the Reference Architecture. Text, reference figures, UML diagrams. Rules about how to interpret the specification. Includes universAAL protocol specifications, API specifications and ontology.
Dissemination Level	PU=Public
Deliverable Type	R = Report
Original due date (month number/date)	M33/ 31.10.2012

Authorship& Reviewer Information	
Editor (person/ partner):	Rubaiyat Sadat/USIEG
Partners contributing	Rubaiyat Sadat/USIEG, Paul Koster/Philips, Miran Mosmondor/ENT, Dario Salvi/UPM, Michele Girolami/CNR, Venelin Arnaudov/Prosyst, Pilar Sala/UPVLC
Reviewed by (person/ partner)	Sten Hanke/AIT, Joe Gorman/SINTEF

Release History

Release number	Date issued	Milestone*	eRoom version	Release description /changes made
0.1	05.03.2013	PCOS proposed	1	Adapted document from previous version
0.2	06.05.2013	Intermediate approved	2	Added sections about Building Blocks, mapping, compliance and verification
0.3	10.05.2013		3	Update summary and conclusions
0.4	30.05.2013	External proposed	6	Updated building blocks section, change structure and complete missing definitions of RAS
0.5	28.08.2013	External revised	8	Addressed reviewer comments
1.0	18.11.2013	Released	10	TM release

* The project uses a multi-stage internal review and release process, with defined milestones. Milestone names include abbreviations/terms as follows:

- PCOS = ”Planned Content and Structure” (describes planned contents of different sections)
- Intermediate: Document is approximately 50% complete – review checkpoint
- External For release to commission and reviewers;
- proposed: Document authors submit for internal review
- revised: Document authors produce new version in response to internal reviewer comments
- approved: Internal project reviewers accept the document
- released: Project Technical Manager/Coordinator release to Commission Services

Table of Contents

List of Figures

List of Tables

Executive summary

This deliverable describes the AAL Reference Architecture (RA) as developed in the project universAAL.

In defining the RA we follow a Service Oriented Architecture development methodology and notation called SOAML (UML-based SOA). We summarize the business context for the RA as investigated and documented in universAAL (in deliverable D1.1 and D1.2). The business context contains mainly the stakeholders and the value propositions they offer to each other. These value propositions are then mapped to specific services that can be traded among the stakeholders. By mapping these services onto a technological ICT architecture we show how the services can be implemented and deployed in the real world. We provide examples of typical deployments demonstrating simple and more complicated scenarios.

The RA is developed through a combination of bottom-up empirical data (collected during the universAAL consolidation process) and top-down definitions. The first one resulted in the description of abstract building blocks that are the main architectural pieces needed to construct the platform, while the second process has resulted in the definition of the Reference Architecture Services, RAS, offered and consumed by the main platform components.

The RA needs to guarantee some level of compliance among concrete architectures, and at the same time make sure to allow innovation and competition in the real world. This is the main challenge facing us when defining the RA. This document has made the assumption that a RA needs to be as simple as possible and focus on modular services that can be provided by independent businesses. In that respect we have not defined the details of *how* the architecture should implement some functionality but on *what* the architecture should provide.

The users of the RA can be the following:

- Healthcare providers and service institutions who want to invest in a specific AAL technology. The RA will allow then to evaluate the functionality of the intended technology, what it provides and what services it lacks, and whether it will be interoperable with other technologies.
- Platform developers who are aiming at creating a standard platform or who are aiming at migrating their existing platforms to a standard platform. In the same way as service providers, these actors can use the RA to evaluate their own platform, map its functionality to the services defined in RA, and in this way do a SWOT-type of analysis (SWOT is an acronym: Strengths, Weaknesses, Opportunities and Threats) of their own technology.
- AAL application developers can use the RA to build applications that will have a better chance of being portable across specific platforms.

1 About this Document

This document is one of the main parts of deliverable D1.3 that documents the work performed in task 1.4 “Consolidated AAL Reference Architecture specification”. Concretely, this part addresses the specification and design of the universAAL Reference Architecture for AAL (RA). The RA has two inter-related purposes:

1. Describe ICT architectures that are a reflection of the real-world AAL business domain, and that address real-world issues in the AAL business domain.
2. Abstract away specific technologies and solutions, and focus on conceptual, generic features of ICT support for AAL. At the same time the RA should support easy utilization of its concepts in form of specific solutions.[1]

RA is not useful if it is only about abstract concepts with no relation to real-world technologies and solutions. RA interacts with specific solutions and technologies in two distinct ways:

1. **Consolidation**: Experiences from real-world empirical implementations of AAL solutions are important for creating a valid RA. This is done through the consolidation process where specific solutions from earlier projects are analysed.
2. **Instantiation**: Creating new AAL solutions based on our RA, or migrating from existing solutions, is what the instantiation process is about. An example of such a process is provided in this document. A more elaborate case is the Concrete Architecture (CA) developed in universAAL project, and provided in Part IV of D1.3.

1.1 Relationship to other sub-parts of this deliverable

Figure 1 below illustrates how D1.3 part III (this document) is related to other major parts of D1.3.

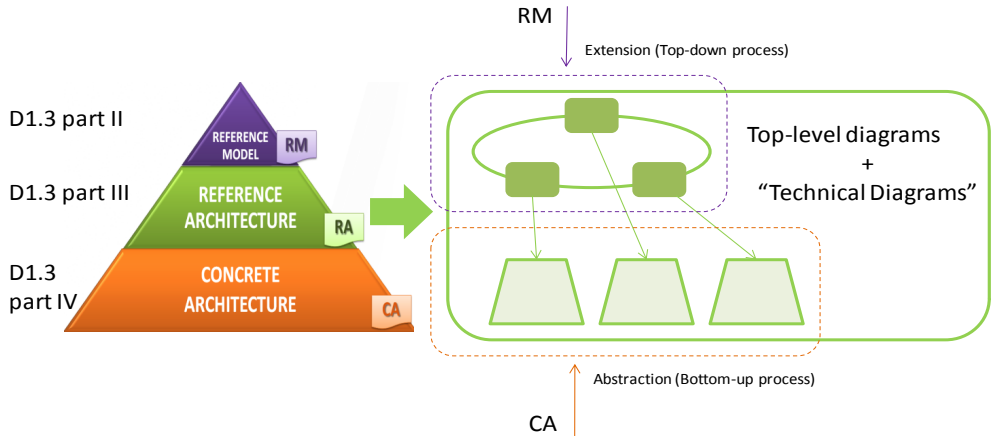

Figure 1: The relationships between the major parts of D1.3

[1] The RA represented in this document can be regarded as the architecture for a class of ICT-based systems, one kind of which is AAL systems. It is necessary to take this approach because AAL systems do not exist in isolation and closely interoperate with other ICT systems and platforms. We see it as a positive aspect of the RA for AAL not becoming "too specific" to the AAL domain, in this way creating unnecessary barriers for interoperability with other platforms.

The pyramid to the left in Figure 1 illustrates abstraction. The upper parts of the pyramid are more abstract (with respect to specific technologies and solutions) than the lower parts. The right side of Figure 2 shows how the three parts are related to each other:

- The green rounded rectangle denotes the RA.
- The Reference Model (RM, part II) provides key AAL domain concepts such as stakeholders, users and relationships that are then supported by the RA.
- The Concrete Architecture (CA, part IV) and consolidation results (part V, not shown in Figure 1) provide the empirical input necessary for creating the RA. This empirical input is in form of real world experience from developing real AAL platforms, on one side, the platforms coming from the input projects, and on the other side, the platform being implemented in universAAL project.

1.2 Relationship to other versions of this Part

D1.3-E part III addresses the following aspects of previous versions:

- D1.3-B part III was mainly focused on the runtime component of the AAL reference architecture. D1.3-C part III was the first attempt to include almost all the major stakeholders in the AAL value network as envisioned in the universAAL description of work, i.e. runtime support, development support, and community/service provider support. D1.3-D part III extended the scenarios with different stakeholders with specific services provided, and D1.3-E refined this to ensure consistency with the Reference Model (D1.3-E Part II)
- D1.3-B part III contained information that was collected from the consolidation process. Much of this information was considered as part of first draft of universAAL Concrete Architecture (part IV of D1.3-C). D1.3-C and D1.3-D part III had a more generalized stance resulting from the top-down approach while D1.3-E combines top-down with the bottom-up approach.
- Related to the point above, D1.3-D part III outlines and describes the preliminary process used to divide AAL architecture into reference and concrete parts. This is one of the more challenging parts of defining reference architecture.
- As part of defining a process, D1.3-C part III has a discussion of success criteria for the AAL reference architecture. D1.3-D part III refines this by addressing the purpose of the reference architecture, and D1.3-E includes also a section about compliance with the reference architecture
- D1.3-B part III used informal diagrams to represent the architecture. D1.3-C part III uses UML diagrams, in particular SOAML which is a profile of UML designed to support the documentation of service oriented architectures, but also deployment model diagrams according to ARCADE methodology. D1.3-E also includes decomposition model diagrams to illustrate the reference architecture abstract building blocks.
- D1.3-D part III has enlisted the specific and complete process of instantiating CA from a given RA related to AAL. No changes have been done in version E
- D1.3-D part III completes the sample scenarios with AAL ecosystem from different stakeholders' point of view. No changes have been done in version E
- D1.3-D part III introduces the authorities and supporters as a role and enlists the services provided by these stakeholders. No changes have been done in version E

1.3 Structure of this document

The AAL reference architecture is presented in Chapters 2-10 as follows:

- Chapter 2 provides an introduction to the terms and purpose of RA, a list of criteria that determine the success of the proposed RA (which are basically describing the concrete goal of RA) and a description of the abstraction process from input projects through consolidation and empirical validation through a proof-of-concept demonstrator.
- Chapter 3 provides an overview of the AAL business context, in particular the value network that is used as the basis for the definition of the RA. The stakeholder groups in this value network and their expectations are discussed here.
- Chapter 4 provides a mapping of the business environment onto an abstract SOA following a simplified version of SOAML [1]. It also presents the results of the bottom-up process in the form of Abstract Building Blocks and its Information Model, that allows describing the structural aspect of the Reference Architecture
- Chapter 5 discusses the behavioural aspects, i.e. how service provision and consumption in the RA happens, using a number of proven collaboration patterns from universAAL and other input projects, in the form of Reference Architecture Services.
- Chapter 6 discusses shortly some deployment and distribution aspects of the RA. In particular how deployment onto smart spaces (AAL Spaces in our RA terminology) is handled, as this requires a high degree of dynamic deployment.
- Chapter 7 provides the description of the instantiation and consolidation process that exists between RA and CA.
- Chapter 8 provides a simple example of an instantiation process including an example AAL service.
- Chapter 9 discusses what methods and tools can be used to evaluate the architecture compliance of any existing, or future, system toward universAAL Reference Architecture.
- Chapter 10 presents the results of the verification and validation processes with regard to RUCs, Reference Requirements and Reference Model, to ensure the consistency and coherence across these project results.

2 Introduction

AAL reference architecture is one of the major deliverables in universAAL. The project participants have long experience in developing high technology AAL solutions and have cooperated in earlier project with the users of such systems in developing AAL solutions. This experience has been the main source of knowledge for developing the RA. The process itself has been long and revisions have been made to the RA. This chapter documents some of the main challenges we have faced and the process we have followed. In the following we provide a short introduction to the working definition of reference architecture. We will then describe some success criteria that are used to evaluate the quality of our RA. We will then describe the process we have followed and the choices we have made along the way.

2.1 Purpose of Reference Architecture

The commonly referred definition of RA is given by OASIS, Open Group and OMG [2]:

"*A reference architecture models the abstract architectural elements (building blocks) in the domain independent of the technologies, protocols, and products that are used to implement the domain.*"

This working definition is used in our work, which we have extended in such a way that our RA also describes the purpose of the RA with a list of expected impact of RA. In using the definition we have consistently focused on the "what" and omitted the "how". The question about how the RA is instantiated and used is answered during the instantiation of the RA into a concrete architecture (see part IV and also Section 7).

During our work we have discussed the purpose of the RA. Although it might seem obvious that we need a RA, it is sometimes not so easy to define the exact purpose of the RA. What we include in the RA and what we exclude can be used as a guideline for what the RA can be used for. In our perception what is useful to include in the RA are the following:

- A set of shared concepts: The basic concepts, such as overall groups of stakeholders and classes of technology, need to be agreed upon. These shared concepts are mainly being defined as part of the reference model in universAAL, with input from the RA activities.
- A set of generic use cases: These are the RUCs, derived by an extensive requirements analysis of the AAL domain. The RUCs define what functional areas AAL systems should provide.
- A set of functions/services: A further specification of the RUCs in terms of services, which allows us to build architectures in SOA manner (Chapter 5).

In addition, we have also deliberately tried to exclude some specific types of RUCs, concepts and services based on the following criteria:

- Any RUCs, concepts or services that will prevent uptake: The list of RUCs, concepts and services could have been much longer. We have tried to keep the lists short and include only what we considered necessary. The longer the list, the more problematic the update will become.
- Any RUCs, concepts or services that will prevent innovation: One important role of the RA is to build a platform for innovation. Any RUC, concept or service that is part of the RA will prevent some form of innovation because it will prevent some stakeholders in doing what they consider innovative. The challenge is to strike a balance in order to maximize both innovation and interoperability.

Apart of our definition of RA we will also need to define a compliance process. A compliance process lies far ahead and implies involving a larger community of practitioners and users who will use the RA and improve it in the process.

An important function of the RA is to highlight the innovative aspects of the proposed AAL platform ecosystem and provide valuable input for the standardization efforts of universAAL partners. The intentions, strategies and progress of the contribution work to the standardization organizations and committees should be described in D1.4 - Guidelines for research in future versions of AAL Reference Architecture and D8.3 - Standardisation usage plan and contributions.

2.2 Expected impacts and goals of the Reference Architecture

We have used the following success criteria which actually derive the expected impact and concrete goals defining the RA:

1. Taking into account all major stakeholders: In the state-of-the-art of AAL systems, one of the problems that have been highlighted as potential barriers for adoption is a lack of presence of business models and value chains that attract certain stakeholders (e.g. service providers).. A RA can help to overcome this problem by demonstrating how the various stakeholders' expectations are addressed so providing a more comprehensive business context in which attractive business models can be derived.
2. Allowing interoperability at a conceptual level: Concepts, functionality and business aspects of AAL systems are under constant development. The RA should capture what is currently agreed upon in the domain in order to enable a conceptual level of interoperability and systematization of knowledge creation by specifying abstract components and building blocks that define the basic framework of AAL systems.
3. Leaving space for innovation: AAL is an innovative area. A lot of resources are being invested in R&D in order to both understands the domain and to understand the possibilities offered by the technologies. RA should not prevent these innovations from happening.
4. Largest possible coverage of existing systems: The RA should demonstrate that it can cover all the demonstrated architectures in the domain in terms of the already developed systems.
5. Different areas of compliance: The RA should allow stakeholders interested in offering or consuming services to select what they need and omit what they don't. The RA should not follow an "all or nothing" approach.
6. Interoperability with related domains: AAL systems are a specific type of mobile and wireless systems, several types of which are used by many present and would-be elderly. It is unrealistic to expect users to switch from other platforms to AAL platforms when they get old. AAL platforms need to be interoperable with a number of similar platforms, and support migration from other platforms.
7. Open for evolution: The RA should be open for changes in form of expanded scope or higher levels of detail in the future as the market matures.
8. Accepted by the stakeholders: The RA should be accepted by the stakeholders in the domain it covers.

Criteria 1-7 have been used extensively in defining this version of the RA. Criterion 7 has up to now been out of scope for us. Nevertheless criterion 8 is the most important success criteria. Our future work includes addressing wider acceptance by exposing the RA to a wider audience than universAAL partners.

2.3 The Abstraction Process: An Empirical Approach

In addition to the software engineering process underpinning SOAML for documenting the RA, we have used an empirical approach for arriving at the RA. This empirical approach includes two interconnected processes inside the project:

- Bottom-up consolidation processes: Knowledge from the input projects were analysed in the beginning of the project to create a first set of specifications for the RA. Most of the input projects had already used their scientific approaches in arriving at their own specifications. This gave us an extensive background material to start with.
- Top-down consolidation process: A task force was set up in the beginning of the project to work with an extensive vision scenario for the project. Parts of this scenario were described in Chapter titled “Example of deployment aspects for a sample ecosystem”. The process resulted in storyboards and architecture descriptions which were used in a prototyping process in the project. This has resulted in a proof-of-concept prototype that demonstrates a number of complex and real world deployments of concrete architectures. In addition, the work on an instantiation process for RA has resulted in an extensive proof-of-concept concrete architecture (see part IV).

3 The business context

This section provides a short overview of the business context of the RA. We describe main stakeholder groups and reference use cases (RUCs) for these stakeholders.

3.1 Stakeholders and expectations considered in the RA

Figure 2 shows a high-level view of a value network[2] that has been used as a basis for creating the RA. The stakeholders are presented as nodes and the provided services of value as arrows. These services are described in more details later in this document, in Section 5.

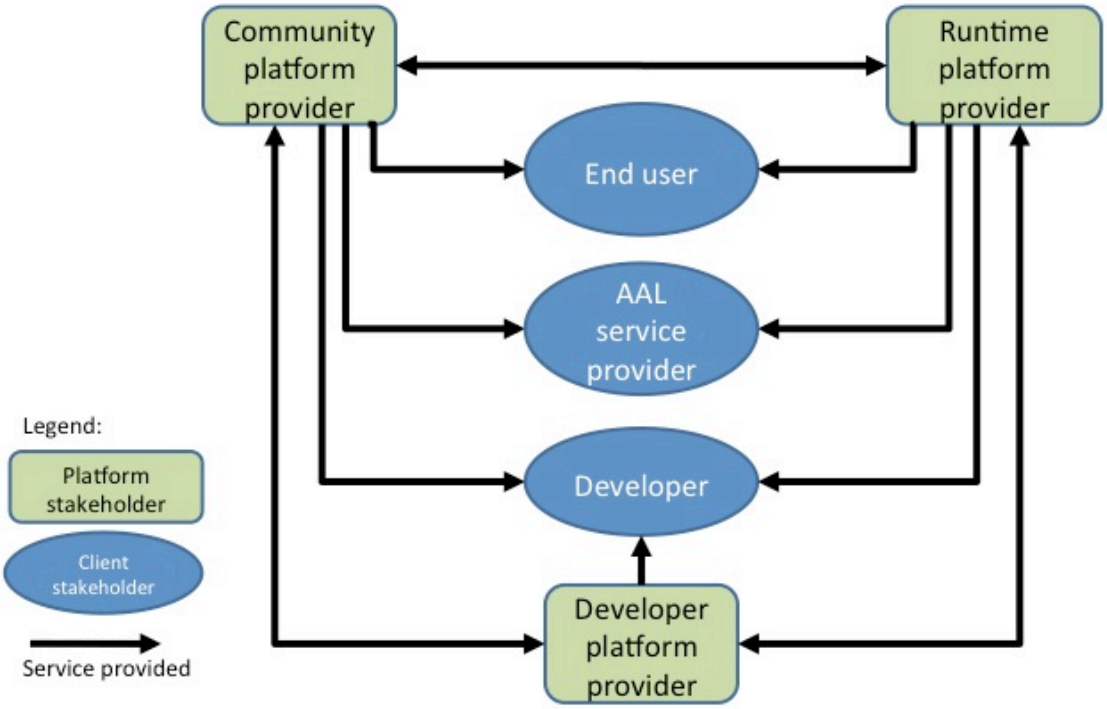

Figure 2: The AAL value network supported by universAAL

We are addressing platforms for AAL. Figure 2 illustrates this by grouping the AAL stakeholders into two groups:

- **Platform stakeholders** (rounded rectangles in Figure 2): These are stakeholders who provide and operate platform components in an AAL environment. We envision a value network where these platform stakeholders will provide services of value to client stakeholders and to each other.
    - Runtime Platform Provider – provides technological support for proper functioning of AAL runtime environments. The AAL runtime environment is where users (the elderly, but also their caregivers) live and interact with the AAL technologies. Typical examples are smart homes.

[2] A value network is a "web of relationships that generates both tangible and intangible value through complex dynamic exchanges between two or more individuals, groups or organizations." Wikipedia.

- Developer Platform Provider – provides software engineering support for AAL developers. AAL services contain a large amount of software, and support for proper development, testing and maintenance of this software is crucial.
- Community Platform Provider – provides technological infrastructure for end users, service providers and developers to build community[3]. The community can be used to trade services, collect user requirements, enable collaboration among service providers etc.

- **Client stakeholders** (ovals in Figure 2): These are stakeholders who buy and use platform functionality provided by the above three groups of platform stakeholders (i.e. they assume the existence of platform functionality). Client stakeholders considered so far in our RA are:
  - End users - non-technical end users such as assisted persons and their caregivers who consume AAL services.
  - Developers - developers of AAL solutions and technologies.
  - AAL service providers - providers of AAL services to the end users

The above classification can be easily mapped onto the stakeholder classification provided by AALIANCE [3]:

- *Primary stakeholders*: These correspond to our end users.
- *Secondary stakeholders*: They correspond to the three platform stakeholders above (if we consider platform in general as a service provided to platform clients). Also AAL service providers are secondary stakeholders.
- *Tertiary stakeholders*: This category can be mapped to our developer category.

Our model of stakeholders can be said to be more towards clarifying the platform ecosystem, and in this way can be seen as a specialization of the AALIANCE stakeholder model. Identifying who builds and operates the AAL platforms, and who the clients of these platforms are, is in our view central for the uptake of AAL services. Platforms can have the positive effect of accelerating service deployment by providing reusable functionality. At the same time, platforms can inhibit innovation by imposing too many constrains on service providers [4]. By explicitly dividing our ecosystem into clients and platform providers we emphasize the services that are provided in the borderline as we will see in the following sections.

The *Authorities* is a special class of stakeholders that represents socio-economical and legal persons/organizations which have an impact on the *AAL Domain*. Their involvement deals with ensuring funding for the services delivered to end-users and with drawing the legal framework in which service provision can reach the target public.. As this group might not directly use the platform and components, they are not shown in Figure 2. However, due to their relevance from the market perspective their interactions with the AAL Platform and other stakeholders have been addressed in this RA.

[3] Note that by community building here we mean mainly online communities. Community building in general is not an ICT problem and is an organizational science problem. Our models and RA address the ICT aspects of building online communities.

The list of stakeholders above is highly generalized. In many specific cases involving specific solutions a more specific list of stakeholders will be required. Deliverable D1.1 in universAAL has a more elaborate list of stakeholders, with examples of specific types of each stakeholder group presented above. For the discussions regarding the reference architecture, the list above, together with authorities, suffices.

After identifying the stakeholders of relevance to the RA, we would like to look into their major **expectations**, i.e. expected benefits that adopting the RA would provide them with. Deliverables D1.1 and D1.2 in universAAL document a thorough concerns and requirements analysis. What is important for us here is to point out the expectations that have been addressed in the RA for each stakeholder group:

- **End users**: Elderly, their families and caregivers are mainly concerned with service quality and usability, which is related to how services are developed (supported by *Developer Platform Provider*, see Figure 2), acquired (supported by *Community Platform Provider*) and operated (supported by *Runtime Platform Provider*). Good development environments will lead to better quality software, and will attract talented developers. Efficient communication with service providers and among service providers will result in services that better meet end user requirements. High quality runtime support will guarantee e.g. fault-tolerance and timely response, and protect privacy. The RA tries to address these expectations by supporting the platform stakeholders in the three central areas shown in Figure 2.
- **Developers**: are mainly concerned with good development environments, a knowledgeable community of developers (addressed by *Developer Platform Providers*, see Figure 2), and access to a market for their software (addressed by *Community Platform Providers*). Our RA supports developers as a major stakeholder and allows developers to participate in a community together with service providers and end users.
- **AAL service providers**: are concerned with efficient communication with their end users (supported by *Community Platform Provider*), fluent interaction with the runtime environment for their services (supported by *Runtime Platform Provider*), and access to talented developers for development and personalization of their services (supported by *Community Platform Provider* and *Developer Platform Provider*). The services provided by our RA allow service providers to deploy, monitor and otherwise manage their services as provided to their end users.

For platform stakeholders in general, the RA addresses a major expectation which is to promote the standardization of the platform ecosystem for AAL technologies[4].

It is these expectations (documented in more details in D1.1) that have guided the development of our RA. These expectations have resulted in a set of reference use cases (RUCs) that play a central role in laying out the functionality for AAL platforms.

3.2 Service capabilities supported by RA

We have so far seen who our major stakeholders are, what major expectations each of them have, and which of these expectations are addressed in the RA. In this section we take the first step to bridge the

[4] We use the word ecosystem to denote a business ecosystem as described by Moore: "*In a business ecosystem, companies coevolve capabilities around a new innovation: they work cooperatively and competitively to support new products, satisfy customer needs, and eventually incorporate the next round of innovations*."[4]

so-far business-related context to what will come in the following sections, i.e. the technical ICT architecture. We will look at how each of the expectations is addressed in form of services exchanged among stakeholders, what we call Reference Architecture Services or RAS.

A first step in this process is to map stakeholders to services they provide to each other. We have already seen the interactions among the stakeholders in Figure 2. The lines among the stakeholders in Figure 2 denote services of value exchanged among the stakeholders. Our RA is concerned with these services and how they can be implemented in a standardized way.

The value network illustrated in Figure 2 is mapped to a SOAML *Services Architecture* diagram in Figure 3 below. In this figure participants (rectangles) denote stakeholders and service contracts (ovals) denote services that are exchanged among stakeholders. Client stakeholders are shown in orange (grey in B/W print), and platform stakeholders are shown as rectangles with thick borders.

According to SOAML [1] services that are exchanged among stakeholders are modelled using a specific type of collaboration called *service contract*. In Figure 3 each of the service contracts (ovals) is a placeholder for all the RAS that are exchanged between the two parties to the service contract. E.g. the service contract between Community Platform Provider and Runtime Platform Provider denotes all the services provided by each of the two to the other. We will see the details of these RA Services in Chapter 5.

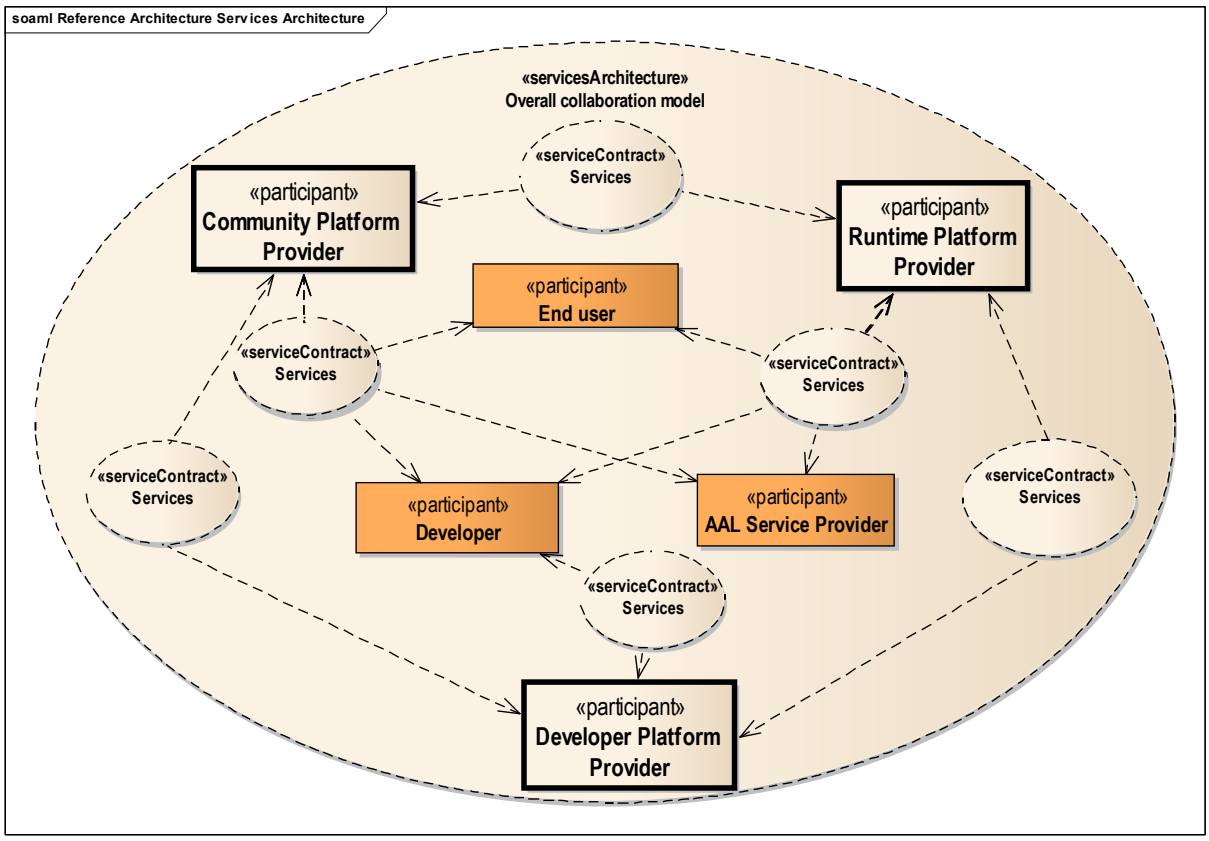

Figure 3: Services Architecture for AAL main stakeholders and their services capabilities

4 The structural aspects (component model)

The following sections provide the mapping between the previously defined business context into RA components, what is called the component model, as a first step to define what the specific services in Figure 3 are, which will be described in detail in Section 5. The participants in Figure 3 are mapped onto UML components that will implement the services. For the sake of simplicity we have provided one top-level component for each of the participants on the overall value network. We will then for each of the identified services define an interface between the two parties to that service, a provider and a consumer interface. Important to note that although we have one component for each stakeholder, in real circumstances (concrete architectures) each service can be provided by a separate business entity, deployed and operated independently. Then we will decompose these top-level components in their inner components that constitute the Abstract Building Blocks.

As an initial step, the following section will outline some of the major information concepts that are used to qualify the provided services. These concepts are mainly related to the AAL service, and how the architecture handles and processes AAL services in general.

4.1 Major information concepts from reference model

As part of specifying any information system one needs to be specific about the information that will be processed by that system. In our case, the system is what we have seen in Figure 3-, whereas the information that will be processed by that system is shown in Figure 4- below. In case of an AAL platform, the major concept is the AAL service and the users involved in developing, providing, acquiring and using that service. This is denoted in the diagram in Figure 4.

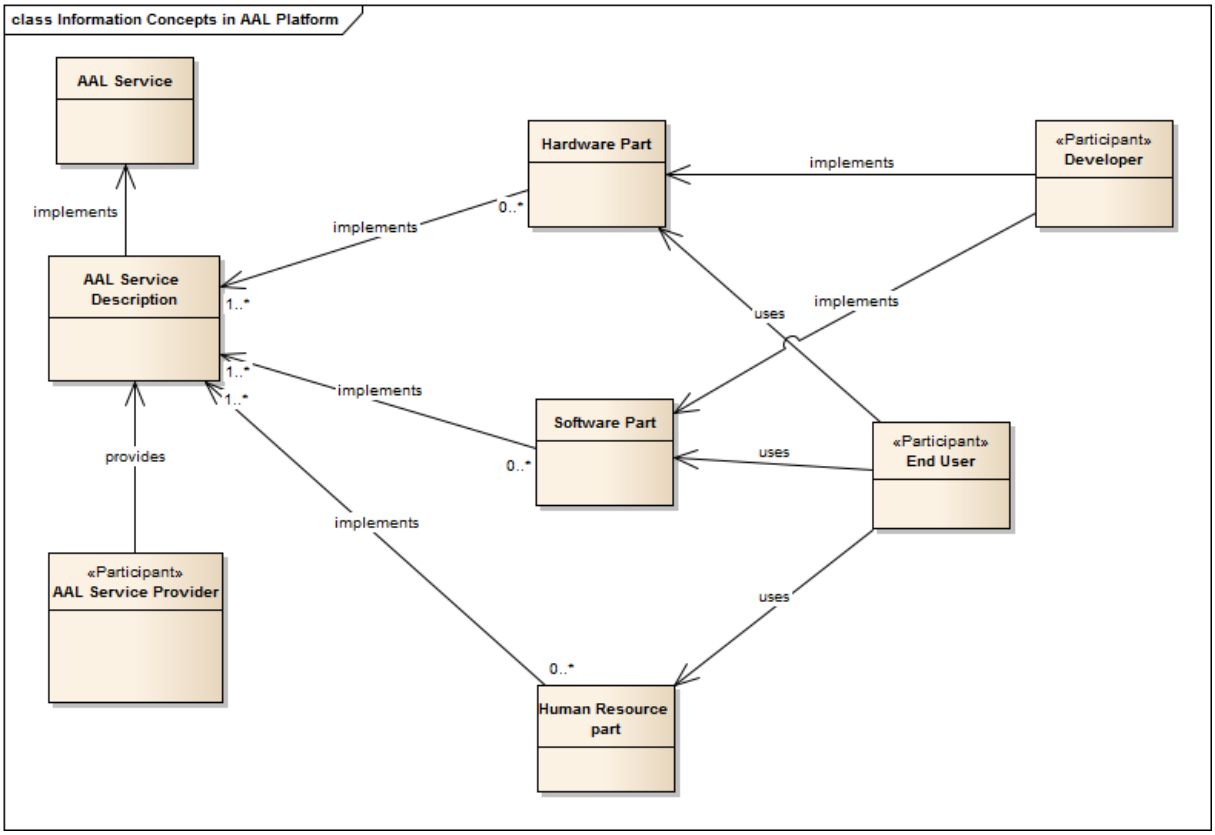

Figure 4: Major information concepts related to AAL service management

Information about an *AAL service* is stored in the platform in form of an *AAL Service Description*. This description contains references to the implementation of the service. In our RA an AAL service is constituted by *parts* that can be software, hardware or human resources. An AAL service might also be constituted by other AAL services (a composed service). In this case the service description will have a reference to the other services' descriptions.

The platform is also concerned with who develops, provides and uses an AAL service. These relationships are illustrated in Figure 4 between participants and information concepts.

4.2 Top-level components

In order to assign a high-level implementation (by a stakeholder) to the service capabilities, we map each of the platform stakeholders -in Figure 3 onto a UML component as shown in Figure 5. In addition we introduce a component called *AAL Service Part* which is the implementation of a part of an AAL service (as introduced in Figure 4).. An AAL service and its parts are third party components, as they are provided by external parties on top of the platform. Nevertheless it is a central component to relate to since a major part of the services provided by e.g. *Runtime Support Platform* is provided to AAL Service Parts in runtime.

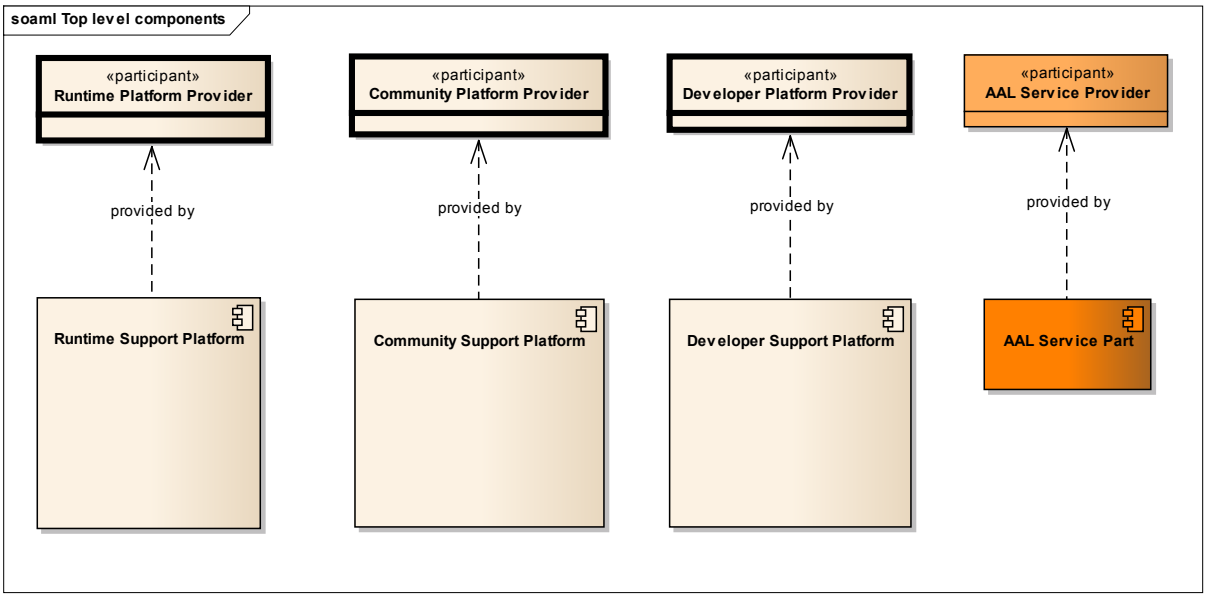

Figure 5: Top level platform components for AAL platform reference architecture

In addition to these top-level platform components (which implement platform services provided and consumed by platform stakeholders), we introduce three top-level tool components as shown in Figure 6. In a service-oriented architecture some component needs to provide the service that the system is promising to provide to the users. In the RA we assign these services to only a few top-level components because we don't want to have a detailed component list (which will be different from architecture to architecture). Each of these top level components represents interactions with client stakeholders[5].

[5] Although the platform components will also have associated tools to be used by the platform stakeholders, these are not yet specified as part of the RA. Our main focus so far has been on tools that are provided to client stakeholders because we believe standardized tools for client stakeholders is of higher priority at this stage.

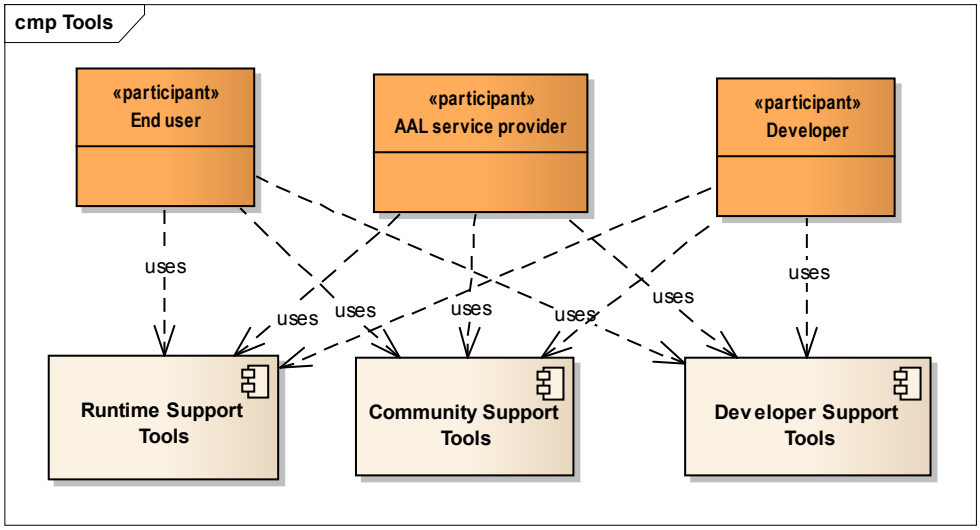

Figure 6: Each client stakeholder is presented a tool component for emphasizing the need for standardized interaction mechanisms

4.3 Abstract building blocks

As presented before, the Reference Architecture is divided into three "double" top level platform components: Runtime, Developer and Community Support sets of Platforms and Tools. The abstraction process from Concrete Architecture to consolidated Reference Architecture involved a mapping of Concrete Architecture building blocks and Reference Architecture building blocks with regard to the aforementioned three top level platform components. The building blocks described here in the Reference Architecture are called Abstract Building Blocks, which will find their decomposition into concrete components in the Concrete Architecture, in Part IV of the deliverable..

Figure 7 presents the above mentioned mapping between these Abstract Building Blocks and the top level platform components, and following are the descriptions of each abstract building block (with a special focus on those of the Runtime Support Platform).

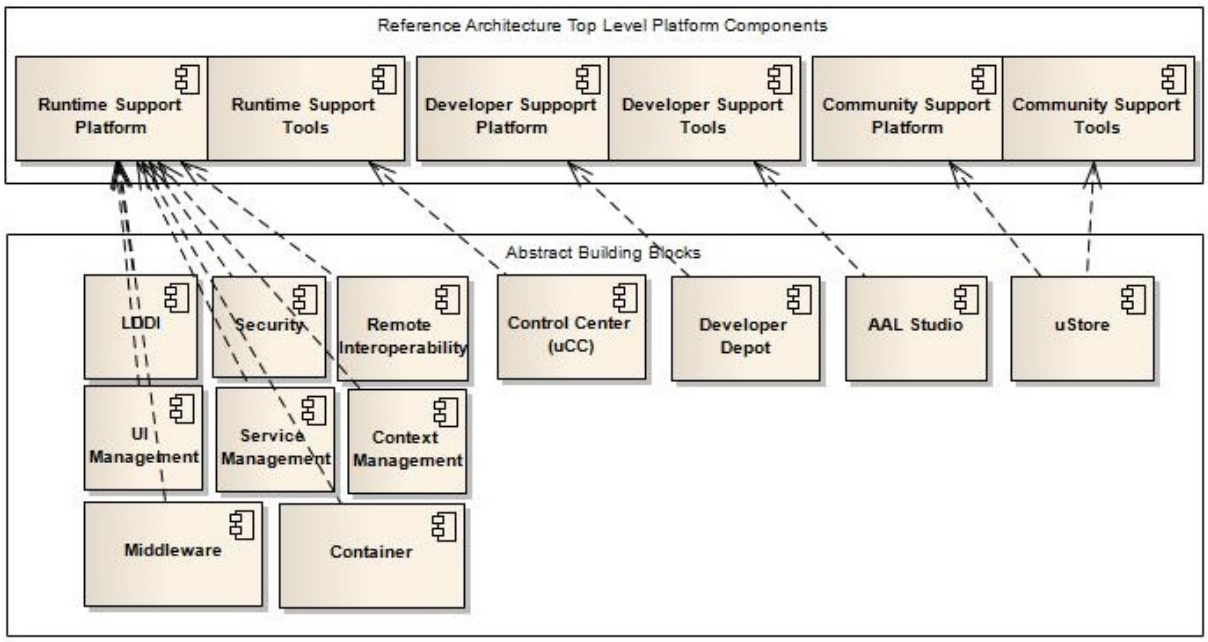

Figure 7: Relationship between Abstract Building Blocks and top level Reference Architecture components

Building block	Description
Control Center (uCC)	The universAAL Control Center (abbreviated as uCC) is uniquely related to Runtime Support Tools building block in RA as it supports the deployment of new AAL services into an AAL environment. This deployment includes the steps that must be done by the deployer (e.g., case manager, technician, or advanced end user) to get the new AAL service (application, hardware, human resources) working in the AAL space. The uCC runs on the Execution Environment, and its three main tasks are installing, configuring and personalising the service. Furthermore, the uCC can support the reconfiguration and uninstallation of AAL services. Also, it provides an interface to easily browse and download the AAL services from the uStore. All these behavioural aspects are consolidated as RAS in section 5
Developer Depot	The Developer Depot provides all resources a developer needs to get started developing AAL applications and plug-ins for the targeted platform. Through the Depot the developer can find and install the development tools (AAL Studio), binary and source code for the execution platform, examples, and documentation of the tools and the platform. Thus Developer Depot is uniquely abstracted as top-level Developer Support Platform where platform-to-platform, platform-to-end-users behavioural aspects are defined as Reference Architecture Services (RASs) in section 5
AAL Studio	The AAL Studio provides a set of development tools that supports different parts of the development process, and that gives the developer easy access to the resources of the Developer Depot. Thus it uniquely builds the Developer Support Tools building block in RA.
uStore	The uStore is the is mapped to both Community Support Platform and Community Support Tool building blocks in RA because uStore gives the end user (assisted person or care giver) a simple way to find and acquire AAL services, and to providers and developers it offers a store where they can upload and distribute their AAL Services. An AAL Service can include not only AAL application (software) and hardware, but also human resources. By acquiring an AAL service, required software (applications and device drivers) will be deployed to the user's runtime platform, access will be provided to required remote software services, and agreements will be made with (local) service providers to reserve required human resources both for deployment and use of the service. For the Community Support Platform building block, the uStore provides facilities to offer their AAL applications to service providers. For the service provider, the uStore gives a management tool where they can find AAL applications offered by developers, make agreements with developers to publish the applications as part of a service, and add the services they offer to the uStore catalog. The uStore serves as the meeting point for all the involved stakeholders where they can share ideas, request new services, ask for help and provide feedback. Details of these behavioural aspects are presented as RAS in section 5

Table 1: All Community, Developer and Tools Abstract Building Blocks

4.3.1 A layered model for the Runtime Support Platform

The Runtime Support Platform is also known as Execution Environment as it comprises the Building Blocks that allow AAL Applications (the software part of an AAL Service) to be executed in the space of the end-user (like the Assisted Person home). 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.

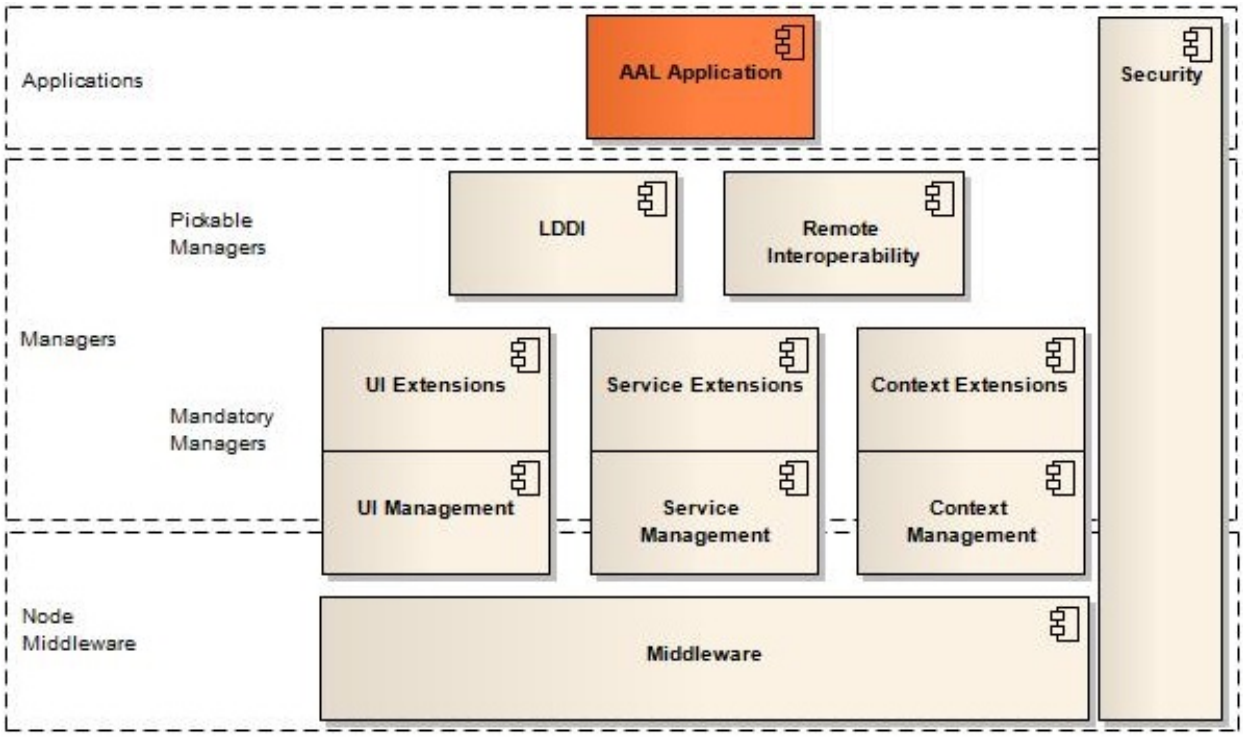

Figure 8: The consolidated decomposition model of the Runtime support platform at the level of abstract building blocks

Building block	Description
Middleware	There must be a footprint of the middleware on those nodes that can host "AAL-aware" software components so that they are able to easily find and share the local representation of the middleware. From another point of view, the possibility of finding and sharing can be understood as a kind of *container* functionality (also known as *virtual hosting*) of the footprint of the middleware that facilitates the integration of its "users" into the distributed AAL system. To hide the distribution, first, nodes must be discovered and enabled to exchange messages. This is also known as support for seamless connectivity between networking-enabled nodes. A major challenge for middleware solutions is the exchange of data. The two questions that have to be handled are: a) what is the data about (domain models), and b) how is it represented (data representation framework). In contrast to data models, the data representation framework belongs to the basics of 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. To provide for a higher level of independence of the middleware users (the communicating parties) from each other, no assumptions about the whereabouts of the parties should be made. In that sense, the middleware is acting as a broker between the communicating parties. This brokerage must handle the exchange of messages in a way that the heterogeneity of technologies used by the communicating parties is absorbed by the middleware.

	The API of this building block introduces a high-level protocol for interoperability over all of the more specific protocols used to realize discovery & peering (D&P) at the device level, and extends the scope of seamless connectivity to the level of end-to-end communication between software components. This API together with the underlying information model to be established by the Data Representation building block should be the only visible interfaces to the world outside the 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, system availability and maintainability of a complex entity of interconnected components.
Context Management	This building block is supposed to provide for adequate communication (both push- and pull-based) between providers and consumers of context data, according to a common model for the representation of such information. Hence, it must *(1)* provide an appropriate brokering for a push-based communication between providers and consumers of context that makes it possible to keep the communication peers independent from each other, (2) guarantee a certain level of persistence both for reflecting the latest state and for making sure that the history of such data does not go lost as long as it might be needed, and *(3)* facilitate access to its own storage through appropriate support for querying the gathered context data. **Extensions:** Context includes shared data representing user preferences and characteristics, so that interaction with the user is personalised, and all components are able to adapt their behaviour to the status and parameters that characterize the user with whom they interact. It will be necessary to provide a model for representing such shared data. Components that provide commonly needed context data are called common providers. The provision of this context, however, can be a matter of "configuration" using the reasoning facilities of Context Management representing the possibility for plugging in such reasoning rules. Additional pluggable components that add new information model items, vocabulary and descriptions which are not included by default in the platform can be added to the Context model.
Service Management	The Service Management building block is supposed to facilitate service-based interoperability. It must handle the registration of services, broker between service providers and consumers for handling service requests (e.g., by matchmaking between received requests and available offers), and provide for service composition and orchestration. For this purpose, a model must be provided that defines how services can be described, how service requests are constructed, and how workflows of composite services should be described. The registration facilities must provide for a secure shared knowledge about the availability of services so that the consumers have a chance to bypass possible faults or change their behaviour when a missing service becomes available. **Extensions:** Common services can be used to speed up the development of different AAL

	services and applications as well as making that development easier. The provision of service, however, can be a matter of "configuration" using the service orchestration facilities of Service Management, representing the possibility for plugging in such common services that exist at a meta level in terms of a script defining a workflow.
UI Management	The UI Management building block addresses the challenges related to explicit interaction between AAL spaces and its human users. It must provide for independence of applications from the concrete I/O infrastructure available in AAL spaces as the latter might differ in its occurrences considerably. It must define a framework for capturing user input and presenting system output to human users thereby brokering between applications and the concrete setup in the AAL Space at hand. Two major challenges in this conjunction are the support for multimodality and guaranteeing the adaptation of presentation to the user situation and needs. The existence of a unified UI model might facilitate the provision of such a UI framework. **Extensions:** Assuming that the separation between presentation and application tiers is achieved by the UI Management, UI handlers can be added, removed or replaced dynamically and freely; hence, they belong to the configuration-specific set of managers. This kind of managers would be responsible for capturing human input (with or without modality fusion) and presenting system output to humans (with or without modality fission) in an application-independent way.
Local Device Discovery & Integration (LDDI)	Not all networking-enabled nodes can be assumed to be "AAL-ready", able to run the middleware or parts of it for communicating with the other networked artefacts. There are often several special-purpose devices, packaged without manipulation possibilities, some even with limited memory or computational capabilities (esp. sensor and actuator nodes from the domain of embedded systems), but still networking-enabled with custom protocols. This building block is supposed to facilitate adding such nodes to the device ensemble that shapes an AAL Space. In order to enable the utilization of the functionalities from such sub-networks, LDDI must overcome the diversity of such heterogeneous systems and devices and help to integrate them based on the common service representation of the middleware layer without much effort. Usually, AAL-aware wrappers or adaptors are used to bridge such subsystems with other AAL-aware artefacts.
Remote Interoperability	A building block responsible for managing platform tasks in the context of interactions between AAL spaces and the outside world. A more precise definition can be provided for this building block based on an analysis of different interaction cases where at least one of the communication parties (no matter if humans or software components) resides within the AAL space and at least one other communication party is outside it. Some common cases include the provision of remote assistance by service providers, remote monitoring, and remote access to services available within an AAL space. This block does not necessarily represent a centralized solution through which all communications with the external world take place. The reality of the existing diversity with regard to wireless and wired communication channels (e.g., DVB-T, ADSL, and UMTS) and protocols available nowadays must be

	respected and if needed a bridging between legacies gateways provided.
Security	The security block has to provide for trust, privacy-awareness, and access control. An artefact 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 artefact at hand does not have any malicious intentions and respects the game rules assumed in the system. After that, the level of freedom of artefacts that have joined the system becomes important. Here, this building block must provide for different mechanisms of access control so that a right balance between the level of trust and the level of freedom in accessing critical data and services can be achieved. One of the most important reasons for the above mechanisms is the protection of the privacy of humans (e.g. to prevent that unwanted disclosure of health details, personal preferences, habits, and lifestyle leads to discrimination, blackmailing or problems in human relations). Therefore, it becomes very important that AAL systems are made privacy-aware. This means that the system must know how such data has been used, which copies of them exist where, which services could or couldn't be used by the users (e.g., due to their own privacy preferences), etc., and be able to provide the users with an overview of this landscape so that they can take corrective actions.

Table 2: All Runtime Support Platform Abstract Building Blocks

4.3.2 Information Model

The information model enabling the communication between the building blocks consists of the following abstract information model elements.

- Operation Profile: A unit of information consisting of the description of a distinct operation that should be registered with the Service Management building block
- Service Request: The content of a message sent to the Service Management building block when outsourced functionality is needed to be provided

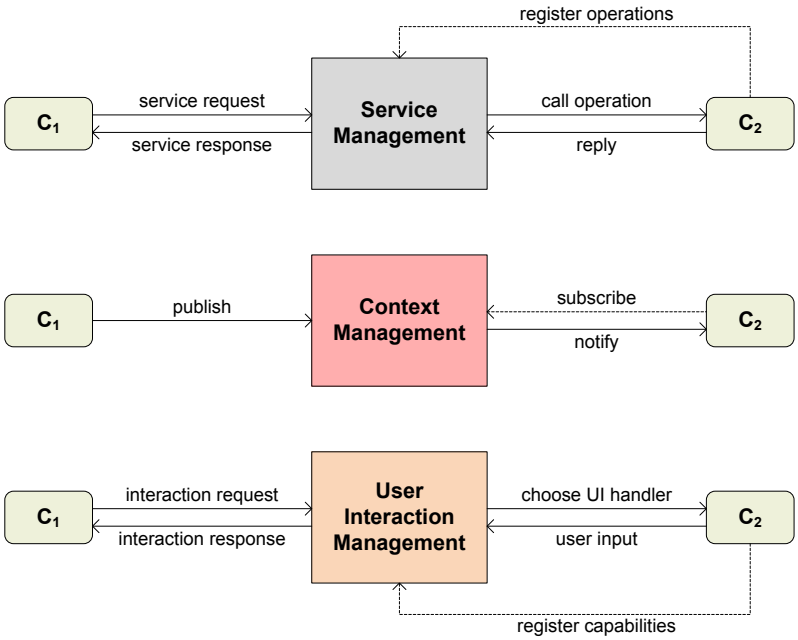

Figure 9: The collaboration model among different building blocks

- Operation Call: The content of a message for invoking a previously registered operation
- Operation Reply: The content of a message in reply to an Operation Call
- Service Response: The content of a message in reply to a Service Request
- Context Subscription: A unit of information consisting of the description of the kind of data in which an event subscriber is interested
- Context Event: The content of a message sent to the Context Management building block when shareable data is available for sharing as well as the content of a message sent by the Context Management building block to a subscriber interested in the shared data
- Interaction Handler Profile: A unit of information describing the capabilities of a component that can use certain I/O channels for handling interaction with human users, i.e. present information to them and capture input from them
- Interaction Request: The content of a message sent to the UI Management building block when the requester needs to present some info to a certain human user and / or needs his or her input as well as the content of a message sent by the UI Management building block to a component chosen for handling an interaction request
- User Response: The content of a message sent to the UI Management building block when a component mandated to handle an interaction request is done as well as the content of a message sent by the UI Management building block in response to an interaction request.

This information model ensures that different abstract building blocks collaborate and this information model is the placeholder for the Reference Architecture Services (RAS) abstraction from Service oriented design approach applied for RAS. For example, in the RAS provided by Runtime support provider *RAS#1.4 'User interaction management'* it is stated that user interaction with the runtime platform is facilitated, e.g. by providing consisting look and feel and adaptability. This includes different information flow among several building blocks, such as Service Request (among different service management blocks), interaction request (among UI block and service management block) and Interaction Handler Profile (among UI blocks). This also justifies the fact that there is (in fact cannot be) **no** one-to-one relationship between RAS and building blocks as a single building block can provide zero to many of the RASes and vice versa.

5 The behavioural aspects (service collaboration patterns)

As explained in section 3.2, the Reference Architecture Services (RAS) are the formalization of collaboration patterns between stakeholders following SOAML methodology. These RAS have been derived from the analysis of the RUCs and provide a comprehensive list of the type of services that any AAL system should implement, however, none of the services, at this point in time, is considered mandatory. Deciding if a RAS is mandatory for a system to be considered AAL needs participation of the whole AAL community; this kind of decisions we hope to achieve it in the course of the standardization process we have initiated in Task 1.5.

The use of RAS is two-fold, it could be used by a platform developer when designing his own system as a guideline of the basic services he should implement, and also it could be used to evaluate existing platforms.

In the following sections we will look into the details of each of the three platform components and its associated tool component. The goal is to identify the high-level reference services (services bubbles in Figure 3) that are provided at four different directions as shown in Table 3. Each type of service will be described in its own section using a table and a diagram.

Each service is labelled with an ID RAS#A.B where RAS means Reference Architecture Service. "A" refers to the Reference Use Case (RUC) category supported (see Section 9.1) and "B" is a running number. These IDs are used as short-cut references to the services.

Table 3: Types of services exchanged in the Reference Architecture Type of service

Types of Services	**Meaning**
Platform-to-Platform	Services that are provided by one platform stakeholder to other platform stakeholders of a different type (e.g. a web service provider requires remote data from a remote database service provider).
Platform-to-Client	Services that are provided by a platform stakeholder to client stakeholders (e.g. a web service provider provides a remote configuration service to a client's smart kitchen).
Peer	Services that are provided by one type of platform stakeholder to other platform stakeholders of the same type (e.g. services exchange among runtime support providers).
Platform-to-AAL Service	Services provided at runtime to operational AAL Services. This category of services is currently relevant for AAL Runtime Support Providers.

The following three sub-sections describe the above types of services for each of the three platform stakeholders.

5.1 Runtime Platform Provider

5.1.1 Platform-to-Platform services

Table 4: Platform-to-platform services provided by Runtime support providers

id	**Service**	**Provided to**	**Description**

id	Service	Provided to	Description
RAS#1.1	Remote maintenance and configuration	Community Support Platform	Allow remote Configuration, management and maintenance of software installed in AAL spaces.
RAS#1.2	Audit data management	Community Support Platform	Allow acquiring usage data about service usage in AAL spaces.
RAS#1.3	Service testing support	Developer Support Platform	Support sandbox testing of services prior to real world deployment.

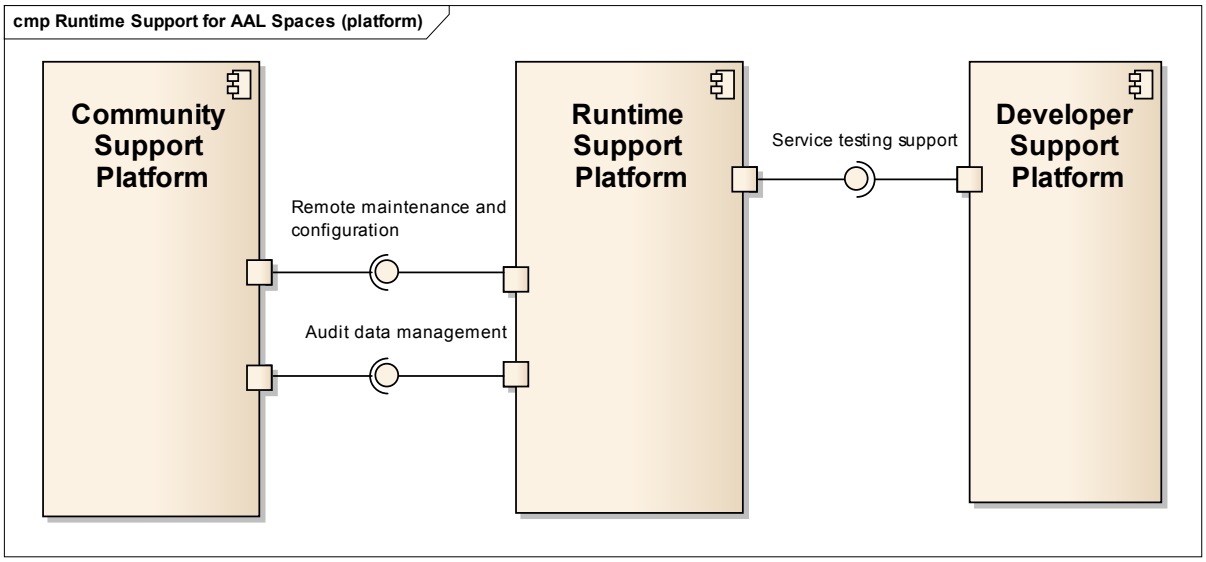

Figure 10: Platform-to-platform services provided by Runtime Support Platform

5.1.2 Platform-to-Client services

Table 5: Platform-to-client services provided by Runtime support provider

Id	Service	Provided to	Description
RAS#1.4	User interaction management	Runtime Support Tool	Facilitate user interaction with the runtime platform, e.g. by providing consisting look and feel and adaptability.
RAS#1.5	Secure communication	Runtime Support Tool	Facilitate secure communication of end user data, e.g. encrypt, sign, authenticate to backend.

Id	Service	Provided to	Description
RAS#1.6	Consent policy specification	Runtime Support Tool	Runtime Configuration Tool can specify his/her consent preferences which denotes who can access his /her data
RAS#1.7	Remote configuration and maintenance	Community Support Tool	AAL service provider can remotely configure the service. In addition they can do the maintenance remotely.
RAS#1.8	Communicate audit trail	Community Support Tool	Audit trail is communicated to the AAL service provider which may contain information such as usage etc.
RAS#1.9	Runtime orchestration of services	Community Support Tool	Service orchestrator is needed for runtime provisioning and deployment of an application through service dependency discovery, service provisioning and service association.
RAS#1.10	Pluggable user interfaces	Community Support Tool	Support the inclusion of new components/interfaces at any time without the need to restart the system.

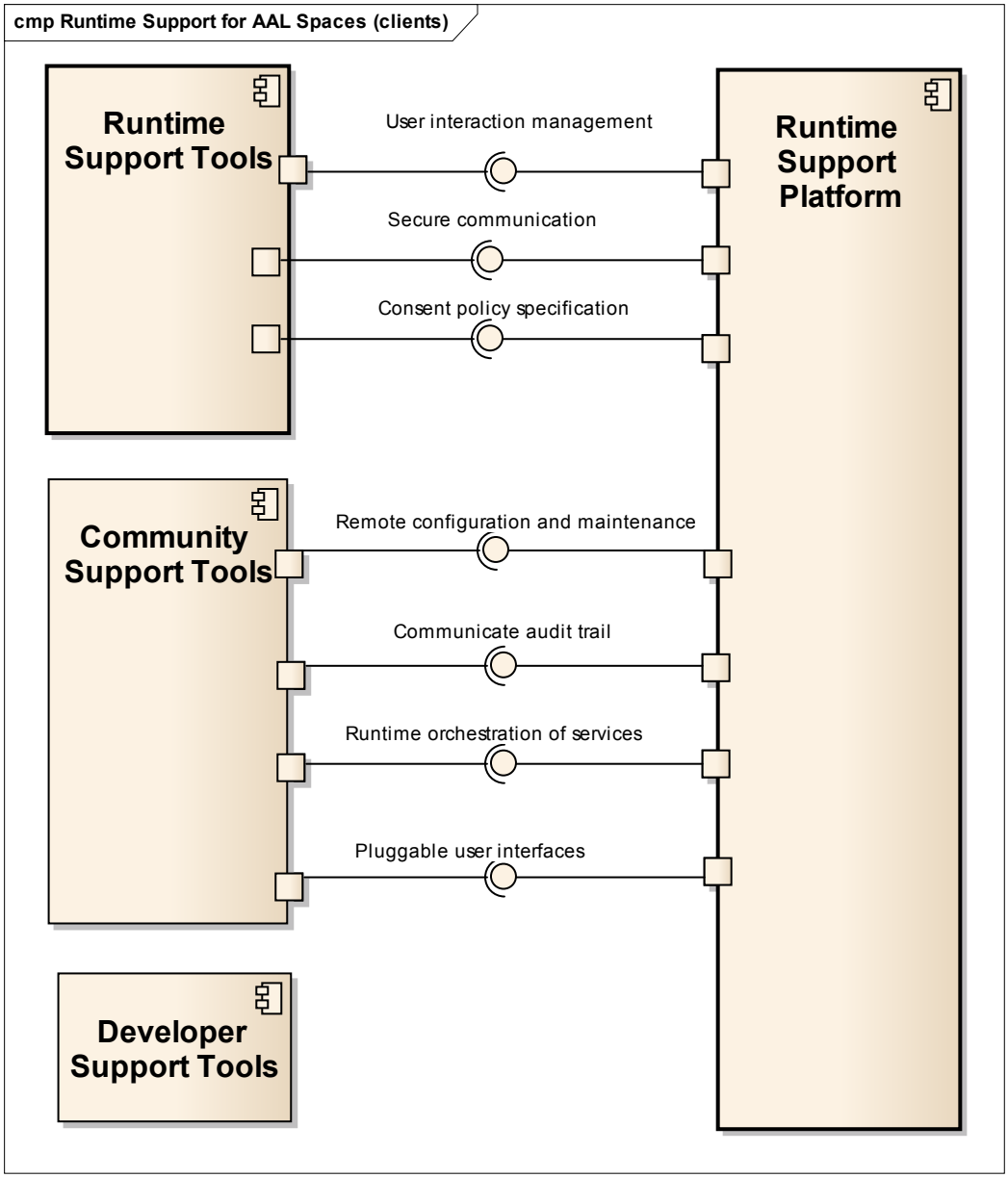

Figure 11: Platform-to-client services provided by Runtime Support Platform

5.1.3 Peer services

Table 6: Peer services provided by Runtime support provider

Id	Service	Provided to	Description
RAS#1.11	Communication	Runtime Support Platform	Allow generic communication among different types and instances of runtime platform.

Id	Service	Provided to	Description
RAS#1.12	Discovery and peering	Runtime Support Platform	Allow discovery and association of other types of runtime platform.
RAS#1.13	Sharing of information	Runtime Support Platform	Allow generic sharing mechanisms for sharing of information with other types of runtime platform.

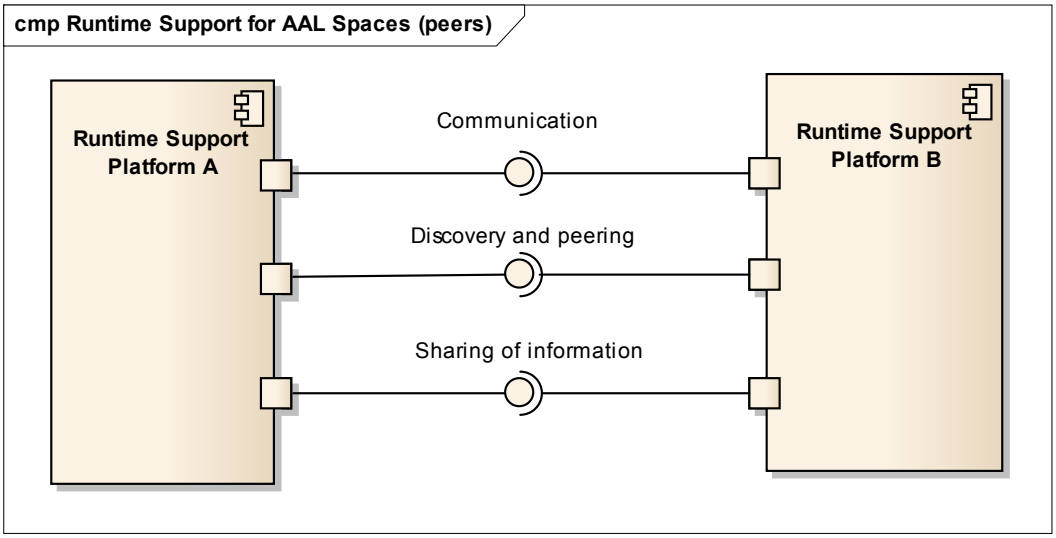

Figure 12: Peer services provided by Runtime Support Platform

5.1.4 Platform-to AAL Service Part

Services provided to AAL service at runtime are a special case for the runtime support part of our platform. These services are described here.

Table 7: Services provided to AAL service at runtime

Id	Service	Provided to	Description
RAS#1.14	Manage communication	AAL Service Part	An AAL Service Part is provided an interface to manage communication with other service parts through Runtime Support Platform.

Id	Service	Provided to	Description
RAS#1.15	Support enrolment into AAL Space	AAL Service Part	AAL service is provided an interface to manage enrolment of parts into an AAL space. AAL service would get information about the AAL space and discover peers.
RAS#1.16	Manage lifecycle	AAL Service Part	AAL service provided an interface to perform functions related to lifecycle management, e.g. updating, testing and activation of parts.
RAS#1.17	Provide secure communication	AAL Service Part	AAL service is provided the capability to receive encrypted and signed information. It can also authenticate the user.
RAS#1.18	Manage context information	AAL Service Part	AAL service is provided the capability to manage context history information obtained the platform.
RAS#1.19	Register reasoning rules	AAL Service Part	AAL service can register rules that can be fired by the Runtime Support Platform.
RAS#1.20	Manage platform behaviour	AAL Service Part	AAL service is given the capability to modify specific platform specific behaviour.
RAS#1.21	Obtain profiling information	AAL Service Part	AAL service is provided the capability to obtain user profile information from the platform.
RAS#1.22	Obtain user consent	AAL Service Part	AAL service can get user consent through the platform.

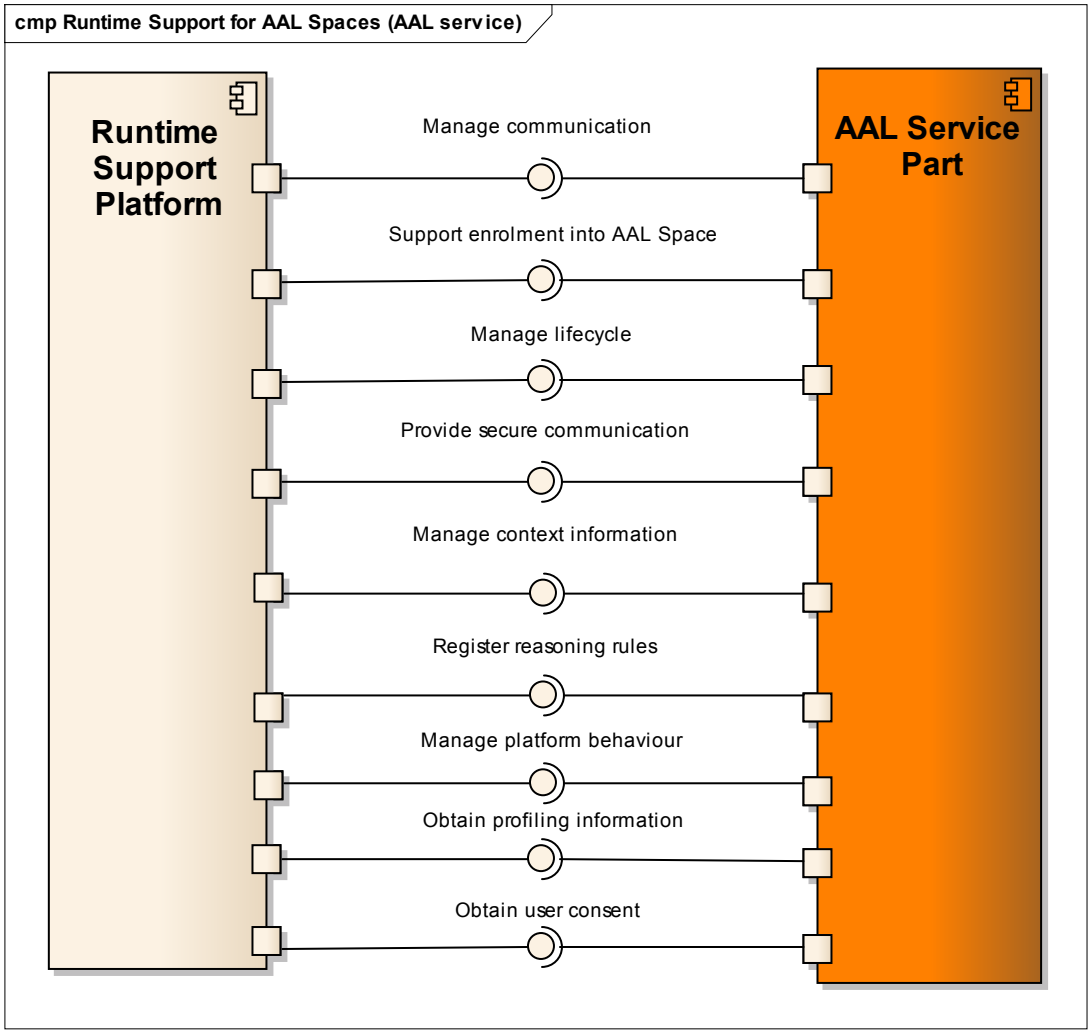

Figure 13: Services provided to AAL Service Part at runtime

5.2 Community Platform Provider

5.2.1 Platform-to-Platform services

Table 8: Platform-to-Platform services provided by Community Support Platform

Id	Service	Provided to	Description
RAS#2.1	Publish development tools	Developer Support Platform	Upload, publish and advertise development tools, training courses, related documentation, etc.
RAS#2.2	Get tool feedback	Developer Support Platform	Get feedback on published tools.
RAS#2.3	Get service ideas	Developer Support Platform	Get feedback about market needs.

Id	Service	Provided to	Description
RAS#2.4	Manage profiles	Runtime Support Platform	Storing and accessing/synchronizing user and AAL space profiles.
RAS#2.5	Download service parts	Runtime Support Platform	Download software to runtime platform.
RAS#2.6	Get runtime support feedback	Runtime Support Platform	Get feedback about runtime support provider.

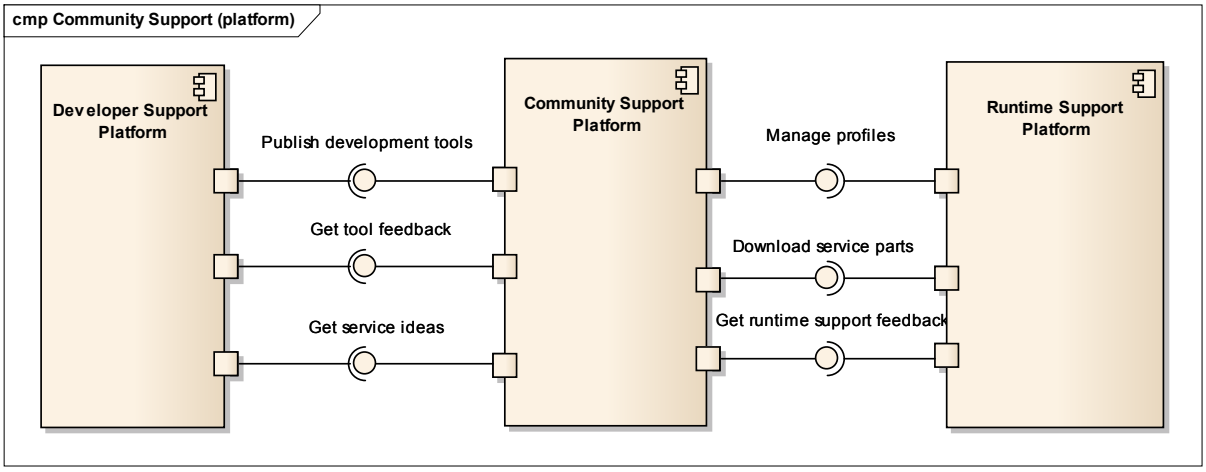

Figure 14: Platform-to-Platform services provided by Community Support Platform

5.2.2 Platform-to-Client services

Table 9: Platform-to-Client services provided by Community Support Platform

Id	Service	Provided to	Description
RAS#2.7	Look for AAL services	Runtime Support Tool	Look for AAL Services based on user needs, preferences and existing environment.
RAS#2.8	Create service request	Runtime Support Tool	Create a request describing new AAL Service or new functionality needed.
RAS#2.9	Acquire AAL service	Runtime Support Tool	Purchase or freely acquire AAL Service according to their status using legal framework (SLA, licences etc.).
RAS#2.10	Provide service feedback	Runtime Support Tool	Provide feedback (comments, ratings etc.) on existing AAL Service or service provider/developer.

Id	Service	Provided to	Description
RAS#2.11	Request maintenance assistance	Runtime Support Tool	Request assistance from AAL Service Providers to install software etc.
RAS#2.12	Communicate inside community	Runtime Support Tool	Communicate with other users inside AAL User community (exchange experience etc.).
RAS#2.13	Publish AAL services	Community Support Tool	Publish AAL services using legal & business framework
RAS#2.14	Explore existing services	Community Support Tool	Explore existing AAL Services & AAL Applications.
RAS#2.15	Communicate with developers	Community Support Tool	Find and manage business connections with developers from AAL Developer Community.
RAS#2.16	Communicate with community	Community Support Tool	Communicate inside AAL Service Provider community.
RAS#2.17	Bundle services	Community Support Tool	Join services of different service providers in packages.
RAS#2.18	Run conformance tests	Community Support Tool	Run conformance testing on software part of AAL Services.
RAS#2.19	Require service certificate	Community Support Tool	Require certificate for AAL Service.
RAS#2.20	Collect user feedback	Community Support Tool	Receive feedback from AAL User Community.
RAS#2.21	Create service request	Community Support Tool	Create a request for new AAL Application from AAL Developer Community.
RAS#2.22	Receive requests	Community Support Tool	Receive requests for new services or updates of existing one from AAL User Community.
RAS#2.23	Upload service parts	Developer Support Tool	Upload software and software updates as AAL applications for free or fee usage.
RAS#2.24	Make business agreements	Developer Support Tool	Make business agreements with service providers to provide uploaded AAL applications as part of AAL services.
RAS#2.25	Collect user feedback	Developer Support Tool	Receive feedback from AAL User Community.
RAS#2.26	Receive requests	Developer Support Tool	Receive requests for new applications or updates of existing ones from AAL User Community and AAL Service Provider Community.

Id	Service	Provided to	Description
RAS#2.27	Communicate inside AAL developer community	Developer Support Tool	Communicate between developers in AAL Developer Community.
RAS#2.28	Run conformance tests	Developer Support Tool	Run conformance testing on uploaded software.
RAS#2.29	Require certificate for AAL service	Developer Support Tool	Require certificate for AAL Application (based on conformance test results).

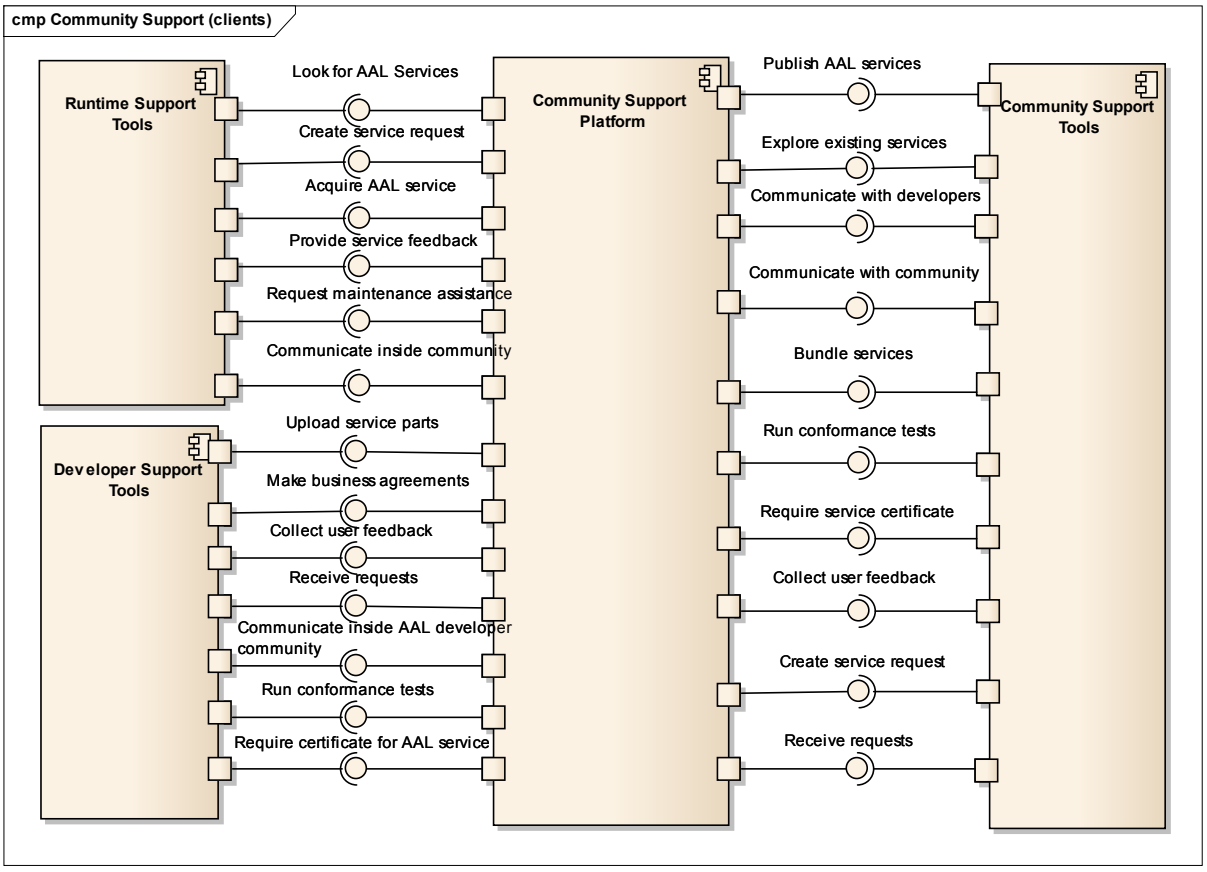

Figure 15: Platform-to-Client services provided by Community Support Platform

5.2.3 Peer services

Table 10: Peer services provided by Community Support Platform

Id	Service	Provided to	Description
RAS#2.30	Explore market	Community Support Provider	Explore proposed AAL Services and interchange information with users from different communities.

Id	Service	Provided to	Description
RAS#2.31	Link to AAL Services	Community Support Provider	Allow purchasing or acquiring of AAL services from another community support tool.

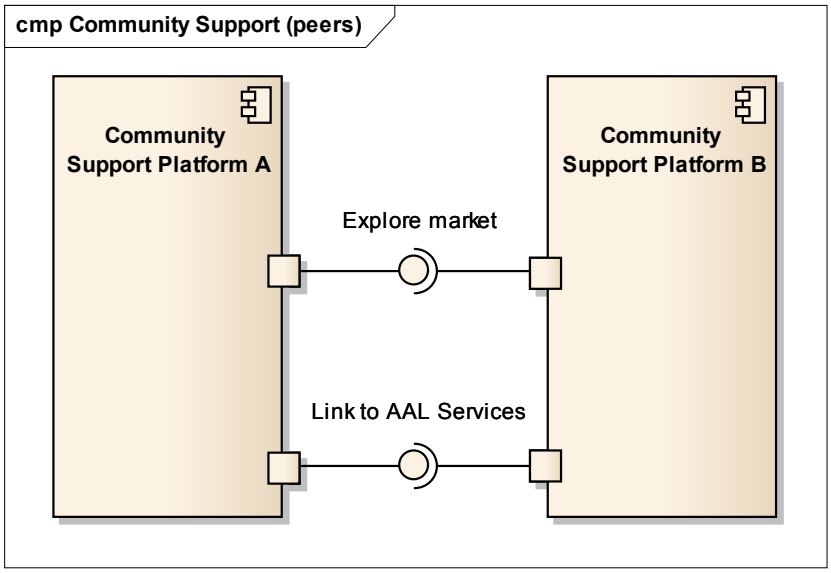

Figure 16: Peer services provided by Community Support Platform

5.3 Developer Support Platform

5.3.1 Platform-to-Platform services

Table 11 Platform to Platform services by Developer Support Platform

Id	Service	Provided to	Description
RAS#2.32	Publish development specification of the platform	Developer Support Platform	Upload and publish the specification of the development platform, tools, related documentation, etc.
RAS#2.33	Share development API	Developer Support Platform	Publish and share developed platform's shared APIs
RAS#2.34	Publicize the developed features	Developer Support Platform	Publicize the features that are newly developed by one platform
RAS#2.35	Host developer support tools	Developer Support Platform	Acts as a host for development tools for other developer support platforms

Id	Service	Provided to	Description
RAS#2.36	Test and verification of other platform specification	Developer Support Platform	Provides test and verification tools for other platform specifications

5.3.2 Platform-to-Client services

Table 12: Platform-to-Client services provider by Developer Support Platform

Id	Service	Provided to	Description
RAS#3.1	Provide IDE	Developer Support Tool	Provide IDE with plug-ins etc.
RAS#3.2	Provide development tools	Developer Support Tool	Allow using wizards, modelling tools, conformance tools, build tools etc. during development process.
RAS#3.3	Develop in community	Developer Support Tool	Develop inside AAL Developer community (Exchange ideas, discuss technical issues etc.)
RAS#3.4	Contribute code	Developer Support Tool	Contribute code to platform
RAS#3.5	Read and provide documentation	Developer Support Tool	Read and provide documentation, guides, tutorials etc.

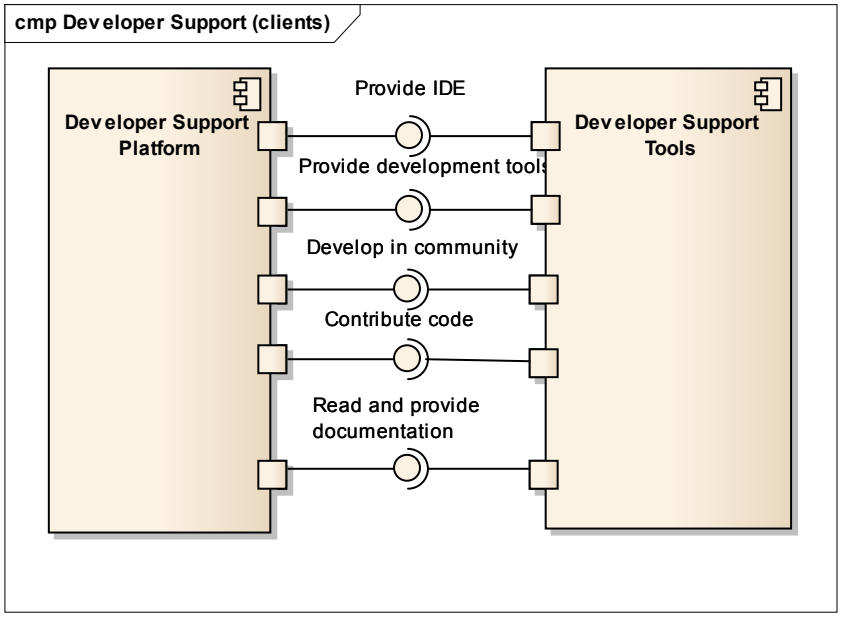

Figure 17: Platform-to-Client services provider by Developer Support Platform

5.3.3 Peer services

Table 13: Peer services provided by Developer support provider

	Service	Provided to	Description
RAS#3.6	Link artefacts	Developer Support Provider	Allow linking of artefacts between developer support tools
RAS#3.7	Exchange artefacts	Developer Support Provider	Exchange code, documentation (about concrete architecture etc.)
RAS#3.8	Provide external interoperability	Developer Support Provider	Provide any needed information, documentation and tools that developers would need in order to interact with another platform

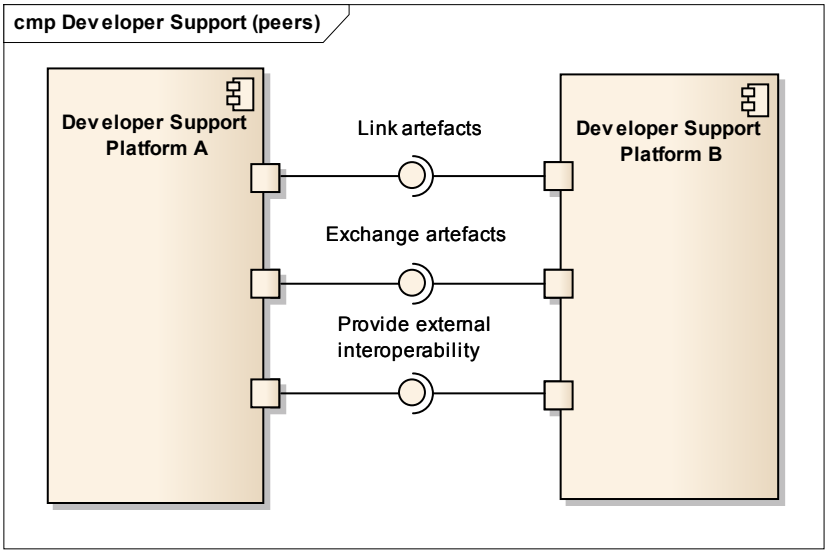

Figure 18: Peer services provided by Developer support provider

5.4 Authorities and supporters (Authoritative service provider)

Authorities need to audit and monitor audit trails to answer basic questions about access to sensitive client data and possible leaks. This is provided by audit service that has a number of components, given in following table.

Table 14: Audit services provided by Authorities

Id	Audit Service Component	Description
RAS#4.1	Logging	Provides a means to submit audit trail messages. A record/metadata description will provide the client submitting records with a description of the expected record type.
RAS#4.2	Reporting/analysis	Provides capabilities to query the audit trail, both with predefined queries and to submit custom queries
RAS#4.3	Monitoring/alerting	Provides capabilities for real-time alert generation and notification of events or event patterns matching both predefined and custom patterns.

6 The deployment/distribution aspects

So far we have identified a set of reference services divided among three platform stakeholders. For simplicity we grouped the services into large components representing our major stakeholders. This means that we did not care whether the provided services were bundled or distributed. These issues are related to the specific deployment cases. Although part IV of this deliverable provides a number of more complex deployment examples, in this section we provide some generic patterns.

Each of the RA services described in Chapter 5 can in theory be deployed separately and operated by a separate business entity (playing the role of one of our stakeholder types). For instance, one player can specialize in security-related services while another might specialize in remote configuration. Deployment is in this sense a business topic as well as a technical topic:

- Technical aspects: How will the deployed services interoperate? This implies agreeing on where the services will run in the network, and how they will communicate with other services, using what protocols etc. and how internal implementations will be hidden from other services.
- Business aspects: Who will provide each service? Providing a service implies deploying the service, operating it, and making profit by selling the service. It also involves issues of trust and security and other regulations.

The reference architecture needs to accommodate for at least the most common business and technical deployment scenarios. At the same time, the RA should not hinder innovation by supporting only a pre-defined set of deployment scenarios and not allowing for new business and technical models to emerge. This is the main challenge regarding deployment.

As a reference point for presenting the deployment and distributions aspects we will use the simplified deployment scenario illustrated in Figure 19 below. The figure shows three physically distinct *deployment locations* (shown as UML nodes) for each of the platforms. Each of these locations is in charge of providing the services associated with that part of the platform. We assume the three locations are physically distributed and connected using networks.

From a deployment point-of-view, runtime support poses a number of technological challenges and is treated more in depth by the RA. AAL Service Parts run at runtime support provider locations (also called *AAL Spaces*), and are supported by the Runtime Support Platform. It is important to note that Runtime Support Platform might be operated by various actors in real world scenarios:

- In its simplest form, the users themselves might be able to operate the platform. This is the case when e.g. a smart phone is used as the only runtime platform.
- In more complicated cases specialized actors might be hired to operate the runtime platform at e.g. private homes or elderly houses. This might be the case when several sensors and home automation services are in operation.
- A combination of several actors (e.g. users themselves together with hired assistance) might be in charge of operating and maintaining the runtime support.
- In cases where more than one runtime platform exists (see e.g. Figure 20 and Figure 21) multiple consolations of actors might be involved.

The next section demonstrates a number of cases for deployment of AAL services in the Runtime provider platforms. The other two types of platforms (i.e. community support and developer support platforms) will be shortly discussed in Section 6.2.

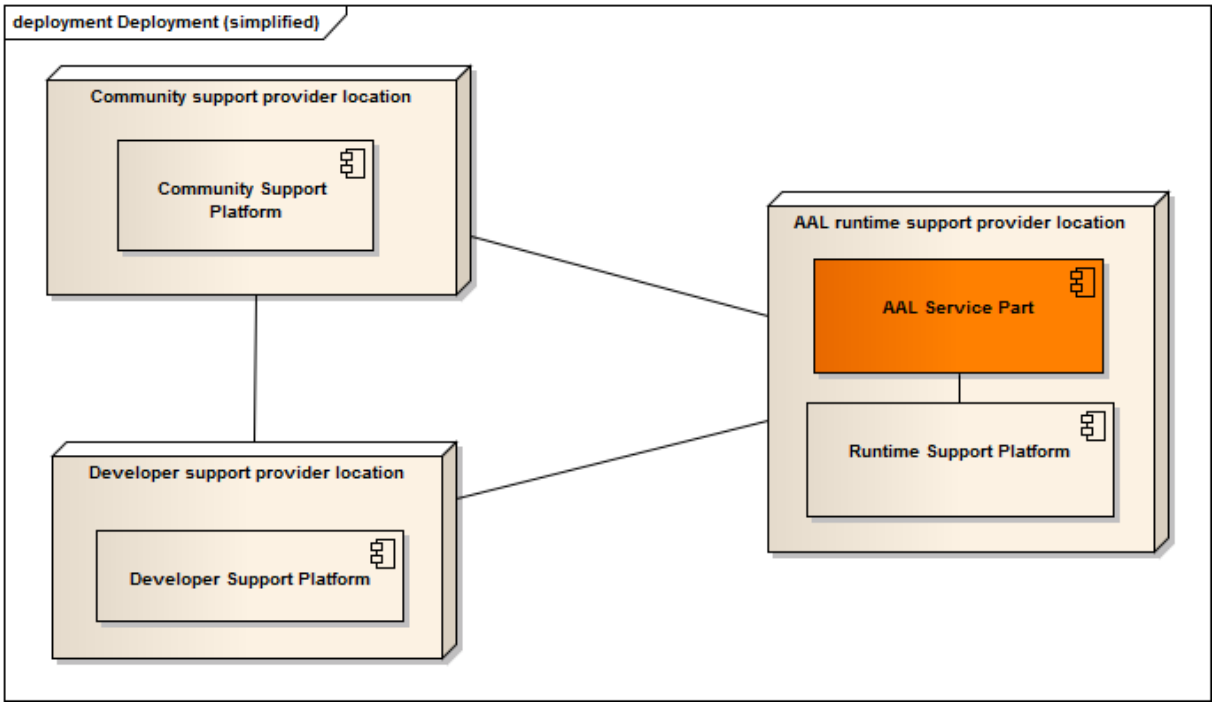

Figure 19: A simplified deployment scenario for RA

6.1 AAL Space deployment scenarios

AAL Service Parts are constituted by various resources that exist in *AAL Spaces*[6], and that communicate with each other in order to provide intelligent support to the inhabitant of these spaces. These could be devices offering some functionality or data (sensors, switches or more intelligent devices), human resources (involved care personnel) or software applications providing some desired support to the End Users. The Reference architecture for AAL should therefore be able to support a variety of scenarios for deployment of the software parts of the AAL Services in AAL spaces (i.e. on top of runtime support). We present a number of these scenarios here.

One step up in the level of complexity from Figure 19 is deployments that allow multiple services run in the same runtime environment. This is shown in Figure 20. This figure denotes scenarios where the End User (e.g. an elderly) is equipped with AAL Runtime platform (e.g. a smart phone) that allows access to a number of services. The scenario in Figure 20 can denote the following configuration:

- The AAL Space can denote the home of an elderly or the elderly's neighborhood. Since the elderly can be mobile, the Runtime Support Platform can also be mobile (e.g. a smart phone running Android OS).
- The Community Support Platform is an implementation that supports a number of services described in Section 5.2 through a single web server.
- The Developer Support Platform might be a private developer site where developers can track service usage patterns among the elderly using their services.

[6] We will use the terms *AAL Space* and *Runtime support provider location* interchangeably as these locations are where the ambient technologies are deployed. While the term *Runtime support provider location* stresses the affiliation of the specific location to one of the major AAL Stakeholders, the term *AAL Space* presents the same as multipart heterogeneous smart environment as introduced in Part II of this document (Reference Model).

- The service parts 1-3 are applications that are downloaded and installed on the AAL Runtime Platform (e.g. a smart phone)[7].
- One or more legacy nodes that are not part of the AAL Service but could be communicated by some Service parts and thus provide valuable contribution to the AAL Service. Although our goal is that our reference architecture should be able to describe all concrete architectures, we reserve a notation for legacy systems that are currently not describable by our RA. This is shown in Figure 21 as a legacy node. A legacy node can always be regarded as a service part running on a runtime environment.

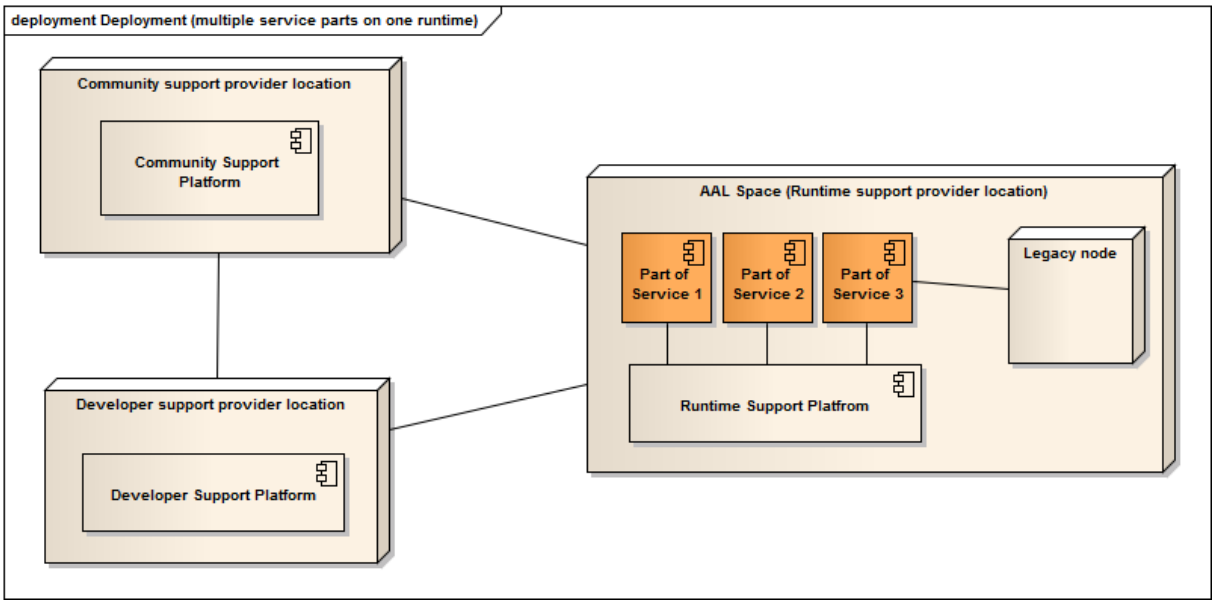

Figure 20: Deployment of multiple services on top of the same Runtime Support Platform

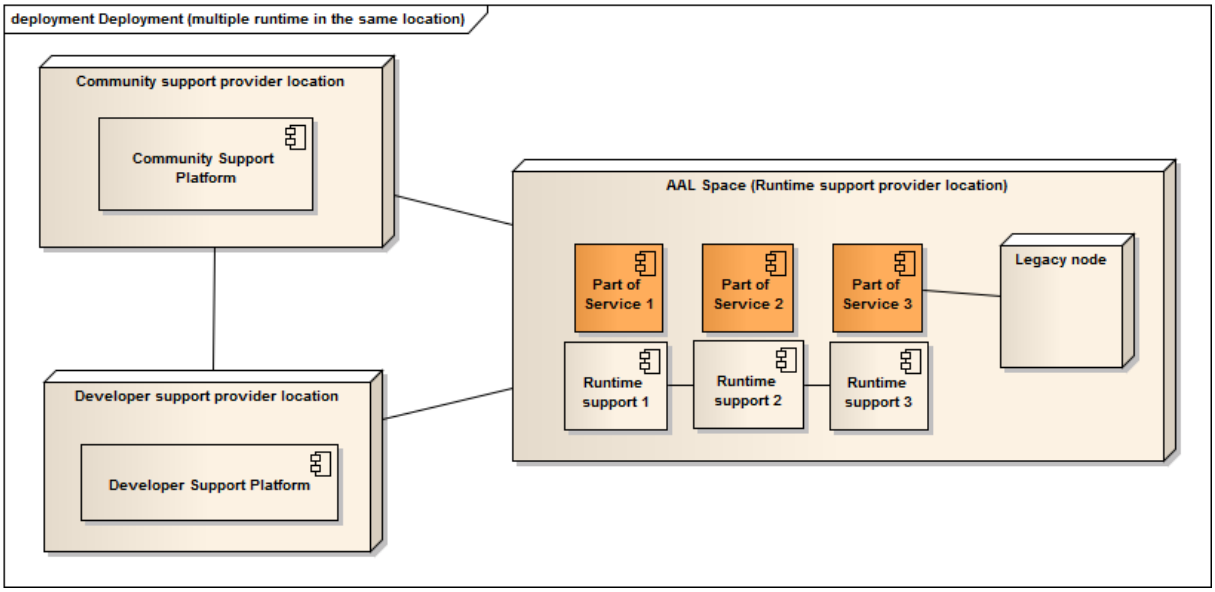

Figure 21: Deployment of multiple services on top of multiple Runtime Support Platforms

[7] Note that the three services here are assumed to consist of each one part.

Figure 21 shows a typical AAL scenario. This figure demonstrates e.g. an elderly person's home (the AAL Space to the right) with a number of specialized services running on top of each their own Runtime Support Platform. These services might represent sensors connected through a proprietary sensor gateway, alarm systems running on top of an embedded OS, smart phones providing some form of GUI, etc. This scenario also illustrates the importance of communication among Runtime Support Platforms and why it is crucial to have a standardize interface for exchanging information among runtimes despite different implementations..

In any realistic scenario an AAL service will involve several end-users (be it an elderly, a care provider, a family member, etc.) in several locations. This also means that the service will have several parts deployed in different AAL Spaces. For instance, a home alarm service will not only involve the home of the elderly, but also the alarm company premises, and possibly the premises of a relative (such as the children of the elderly). In this way the alarm company and the relatives can monitor the alarm at the elderly's home. This scenario is shown in Figure 22 below. In this specific example we can see a case with sufficient and realistic complexity:

- Service 1 is the alarm monitoring service. It might involve parts that run on normal Windows PCs (Runtime support 1). These parts will be deployed at the premises of the alarm company (premise 1 to the top-left of the figure) and at the elderly's home (AAL Space at premise 2 middle-left in the figure).
- Service 2 is the alarm software itself running e.g. on an embedded OS (Runtime support 2) on a device representing the physical alarm. This service will most probably be deployed at the elderly's home (premise 2).
- Service 3 is a communication service (e.g. SMS or email) that receives and forwards messages from the alarm. This service can be deployed on e.g. a smart phone (runtime support 3) in the premises of the elderly and of his/her son/daughter (premise 3 bottom-right in the figure).

The scenario in Figure 22 shows the importance of not only communication among the various platform stakeholders, but also among the same stakeholder types (here runtime support providers). We can imagine that a lot of service specific communication will happen outside the platform. However, the value of a standardized platform becomes clear when multiple service providers need to communicate with each other in a standardized way (even without knowing about each other).

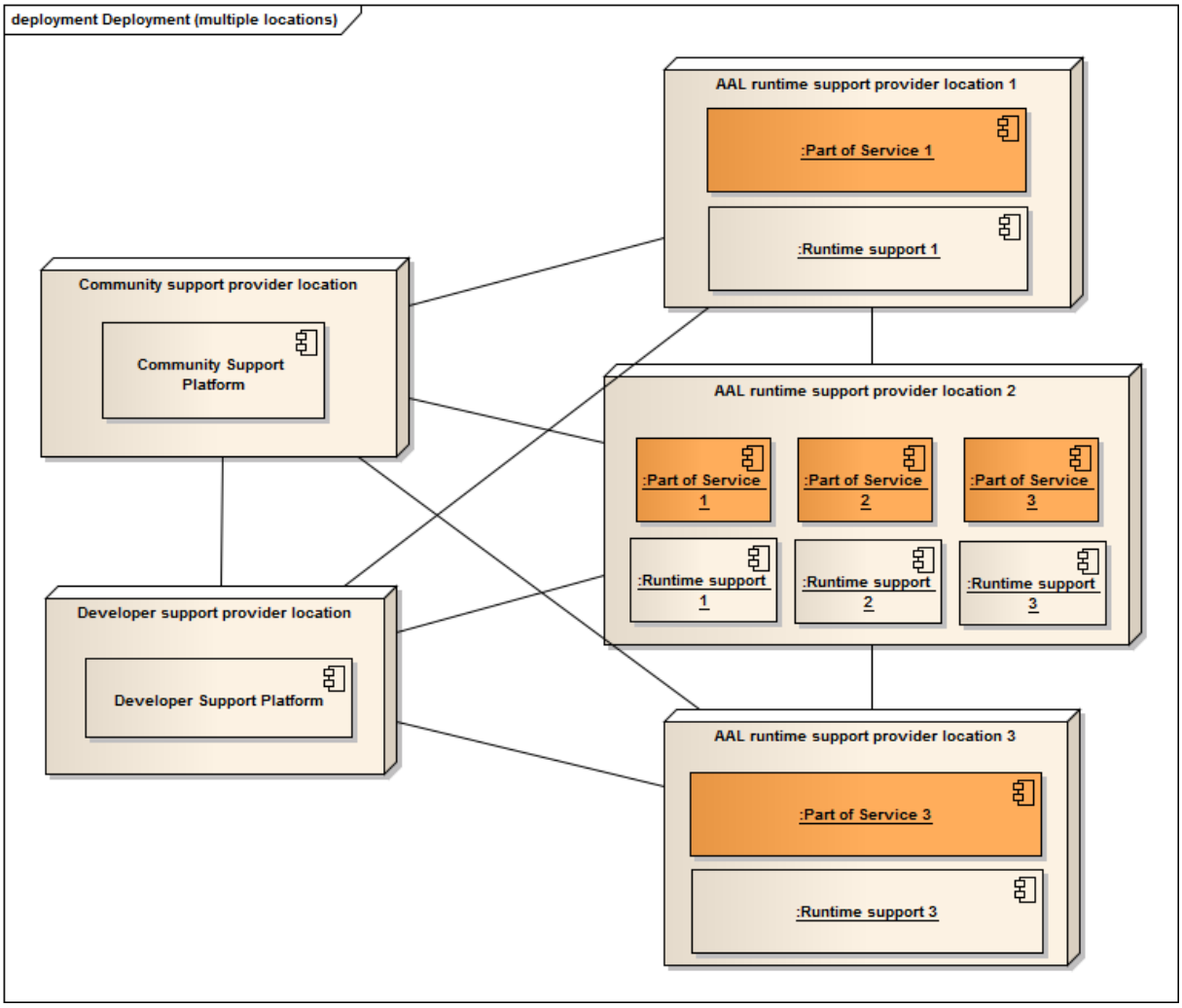

Figure 22: Scenario showing multiple services deployed at multiple locations on top of multiple runtime supports

6.2 Deployment of other platform components

We have focused in the deployment aspects relevant for the runtime support. This does not mean that the other two platform stakeholders will not demand more complex deployment opportunities than represented in the above cases. Other possible deployment cases can consider:

- Multiple community and deployment support providers. We can already see a number of online support communities for developers (e.g. sourceforge, github). We can see the same emerge for the AAL domain.
- Community and developer support providers that provides only a subset of the services described in or RA. Our RA has been defined as an exhaustive set of services. It is clear that not all stakeholders will be interested in all the services. We can see communities of platform service providers emerge in the future.

6.3 Example of deployment aspects for a sample ecosystem

This section provides examples of how the deployment aspects of the reference architecture described so far can be mapped onto a specific ecosystem. The examples are provided as a storyboard. More details on how the concrete architecture for such an ecosystem might look like are provided in part IV of this deliverable. Deliverable 1.1 contains more elaborate scenarios and storyboards describing some

of the features of the storyboard here in more details. The usage of RA for a sample ecosystem is viewed from two different perspectives.

a) Platform stakeholders' perspective

b) Client stakeholders' perspective

In the following subsections, two different ecosystems are described to reflect the above mentioned stakeholders who can use RA to achieve their respective goal.

6.3.1 Platform Stakeholder Perspective

Today AAL service providers in Norway have a large offering of services. These are often services that specialize in areas such as monitoring systems, alarm systems, information provision to the elderly, daily cleaning and care, and coordination of staff. These services are often provided using proprietary technologies, and operate in isolation.

As a first step in streamlining the AAL market in Norway, *uStore Inc.*, a company specializing in AAL, decides to set up a community support portal for AAL service providers. uStore Inc. decides in the outset to comply with the European AAL reference architecture. To begin with, the uStore portal acts as knowledge exchange online community. The services from the RA that are supported are the following:

- Collect feedback from users and service providers
- Collect ideas from users for new services

uStore Inc. markets the portal towards users, service providers, and developers. The portal becomes a success. There are no other online sites in Norway to discuss AAL services. When the discussion about service ideas becomes more structured, uStore launches services that allow

- Service providers to publish information about AAL services they offer
- Users to register a personal profile that will allow personalized access to the site
- Users to provide feedback to specific AAL services

At the same time uStore Inc. signs an agreement with a Norwegian online payment company, *Betal Inc*, to provide payment services to uStore registered service providers. Betal is a partner of the Norwegian State Pension and is specialized in supporting Norwegian citizens in financial issues related to health services.

By now uStore has more than a hundred thousand registered users. It has become the de facto directory for searching and acquiring AAL services. It is used by consumers and by municipalities, and by SMEs who provide services to the municipalities. Gradually a market for privately acquired AAL services has also emerged thanks to uStore. The service-oriented approach of uStore is praised by municipalities and users because they can force service providers to provide complete and packaged services to them, with clearly defined terms of service. End users are also happy because they can have immediate access to feedback provided by other users about the different services. And they can of course provide their own feedback. Integration with Betal AS is also considered one key aspect of the solution.

AAL services are technologically advanced services including not only human resources but also lots of software and hardware. uStore does not support the technical aspects of the services they have in their directory. Only business-related terms of service specifications are supported, together with audits and payments by Betal. By now, due to the growth in the number of AAL services per user, technological issues start to emerge. Users complain about solutions not being interoperable, or that

they have to interact with and configure lots of devices all the time. On the other hand, maintenance of technical components has become a considerable cost for the service providers and for the municipalities. Users are not happy.

At the yearly Norwegian AAL Conference, uStore Inc organizes a service provider's workshop to discuss the matter. During this conference the community decides to start a new company to take care of these technical expectations. The new company *Install AS* is set up as a cooperative by uStore user community and gets the mandate to provide the following services (in line with AAL RA):

- A one-stop portal for technical discussions related to AAL service provision
- A directory of AAL services which complements the uStore business-oriented directory with technical information about provided services
- Support for remote installation and maintenance of software
- Support for remote and mass configuration of deployed services

Install portal is mainly targeted at technically-oriented users. Because of compliance with the AAL RA (e.g. RAS#3.2 which allows using wizards, modelling tools, conformance tools, build tools etc. during development process) the portal is easily linked to uStore's portal. Browsing a service in uStore one can now click and receive more technical information about service parts and information about how services work. In addition, integration with uStore portal allows for one-stop shopping and download of simple AAL services similar to what is currently available in commercial application markets. End users are also provided with self-service functionality for simple maintenance, such as upgrading software on their devices at home. These simple steps eliminate some of the costs related to sending installers to people's home for every minor upgrade or maintenance.

In this way Install AS creates a national de facto standard for AAL service parts. In order to be available in Install portal, services have to be packaged in a specific way, signed, made available online etc. In order to assist SMEs with these tasks, Install AS starts cooperation with *AALSource Inc*, an online developer community for programmers of AAL services. AALSource already provides a comprehensive developer's toolbox online, including source code management, build management, forums etc. Together they plan a migration of AALSource into AAL RA and start providing the following services:

- Online resources related to best practices for AAL service development
- Integration with user forums on uStore portal so that developers can have instant access to user feedback on their services
- Integration with Install portal to allow direct upload of modules
- A set of tools based on the popular Eclipse development platform that helps developers in their development of AAL service parts
- A set of code libraries in order to promote standardization of some service aspects
- Single-sign-on to AALSource, Install and uStore portals

The developments described here demonstrate a typical deployment of AAL Reference Architecture. The resulting ecosystem is shown in Figure 23. In this figure each UML node symbol denotes one business actor's location. The component instances in each node denote the part of the RA functionality deployed in that premise. Some aspects relevant to this ecosystem and the usage of RA:

- The ecosystem is flexible and so should the RA be. The RA does not make any demand on the existence of an actor, but supports the actor when it is present.

- The main stakeholders defined in the RA can be represented (partly) by various actors. For instance, the RA Stakeholder Community Platform Provider is represented by three different actors (uStore Inc, Betal AS, and Install AS) together providing the desired functionality.

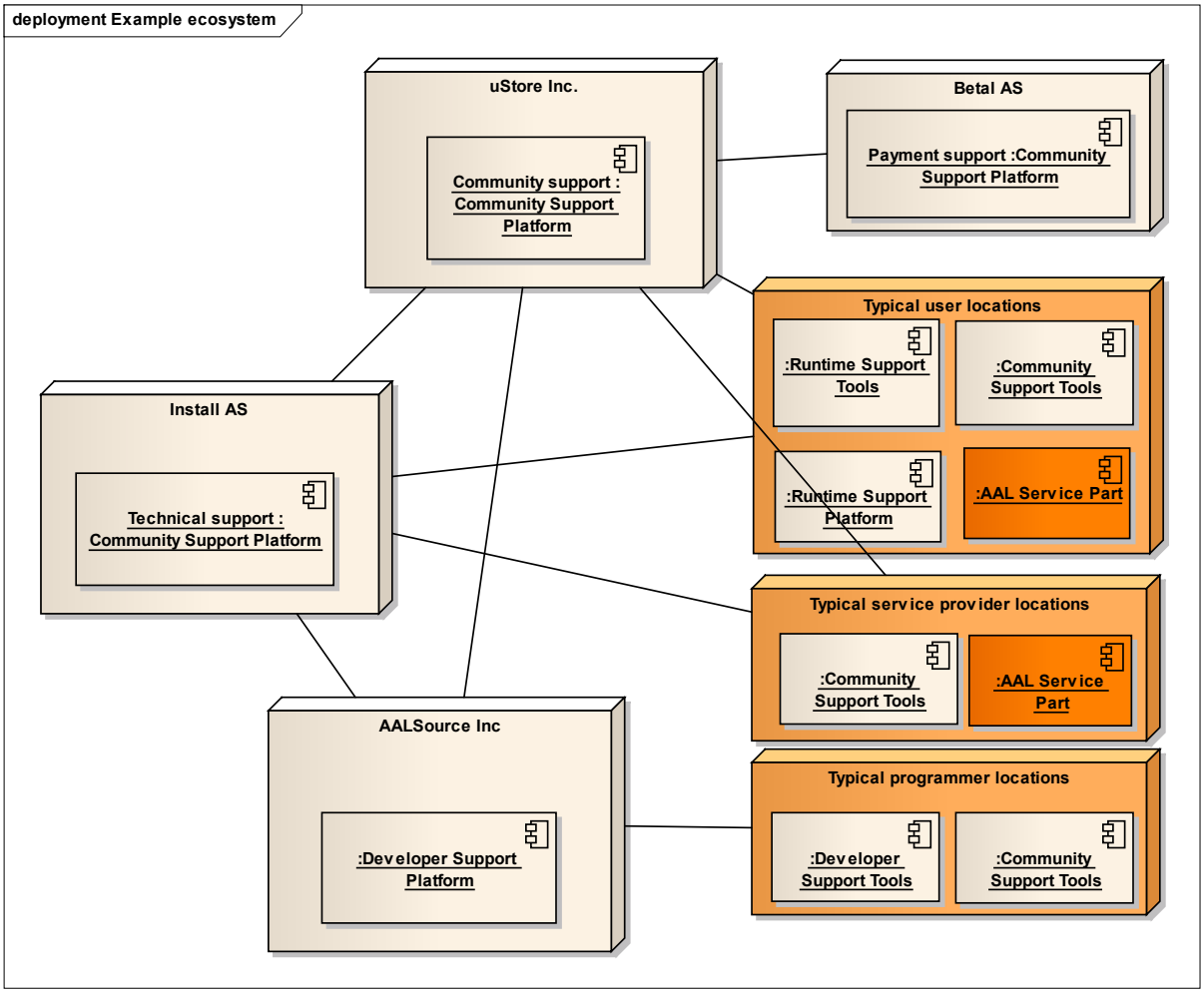

Figure 23: The ecosystem involving the actors described in the storyboard

6.3.2 Client Stakeholder Perspective

In this section we provide an example of reference architecture deployment from a client perspective. Peter Nitter is an elderly person who lives alone in Vestfold, Norway. His son Jack Nitter has a job in Oslo and lives there. Jack worries about his father, since he has a problem following a healthy diet after the death of his wife and moreover he has hypertension and high cholesterol level. In order to assist his father in his independent living, Jack started to look for AAL services for him. He learnt about the portal of *uStore Inc.* that offers AAL services in such areas as monitoring systems, alarm systems, information provision to the elderly, daily cleaning and care, and coordination of staff.

Jack found a complete Nutrition Service Package at *uStore Inc.* that includes kitchen sensors for cooking activity monitoring, provided by *SensorTera Inc.,* and related software services, provided by IT company *InnovIT,* for analysing cooking activities and providing nutritional advices based on a nutritionist's guidelines and its questionnaires. Jack liked the Nutrition Service Package as it just looked like one complete seamlessly integrated service, since its parts were developed using *Install AS* national de facto standard for AAL service parts. He had decided to acquire this service for his father,

so he requested the service via *uStore Inc.* portal and paid for it with the help of *Betal AS* payment service provided at the portal. The users of *uStore Inc.* portal get attracted and start using it, since

- it is a one-stop portal to find various AAL services;
- AAL services are easy to search using information given by the service providers based on a client needs and preferences;
- it offers bundle services composed out of services provided by different service providers to bring complete AAL solutions;
- it has integrated ordering and paying services for AAL services' acquisition.

Once the order had been finalized, sensors arrived and their installation in the kitchen was done by *SensorTera Inc.* Jack also downloaded the corresponding software. During software installation Jack's father got registered as a main client and Jack as the caregiver. Peter also had to specify in the form of consent the access policy to his data that are collected by the sensors and parts of software services. First Peter chose from a list of available nutrition experts Anthony Hoest to be his nutrition coach. Then he digitally signed the consent where he allowed data needed for appropriate running of Nutrition Service to be only accessed by Anthony. During the installation process it was clearly stated that the services in the package would only be using secure communication channels for data communication and would only be storing personal data in the encrypted form. As a finishing touch, Jack and his father personalized the services to take into account some of Peter's food preferences and peculiarities, e.g. he is allergic to honey. Services of *uStore Inc.* are trusted and popular, since

- services support secure communications of client data;
- privacy and security policies are in place and can be personalized;
- clients can request and obtain physical or remote assistance with installation and maintenance;
- services can be personalized.

Peter started to use the nutrition services. He receives his daily menu and shopping list every day. Anthony is adjusting his menu using questionnaires that are sent to and filled by Peter and monitoring Peter's cooking activities and behaviour with a help of designated services. Jack feels confident that his father has regular and healthy eating pattern. Moreover over time Jack also noticed that his father's blood pressure and cholesterol level started to improve.

One day Peter got a cold and stayed in bed. Sensors in the kitchen detected limited cooking activities, and a monitoring service triggered an alarm for both Anthony and Jack. Knowing Peter's health status, Anthony contacted Peter to find out the course for this alarm and, advised him take more liquid. He also adjusted Peter's menu accordingly. Jack also contacted his father to check what was going on. This event made Jack consider getting an extra service for his father. He searched *uStore Inc.* portal for emergency notification service. He found a number of services. He read the reviews of the users and in the end he liked two of them, as both had high user ratings and good reviews. So engaged into the discussions of these services in the forum and in the end got convinced to select one that was offering fall detection together with GPS positioning. Jack ordered the service. While ordering the service, he also noticed that there was an additional interface available for the Nutrition Service that gave information on calories intake, so he requested it as well. The interface was easy to add to the existing nutrition service, with no need to restart the system. For the emergency notification service, Peter also signed the consent only allowing his caregivers, Jack in this case, to access his location information as well as fall detection alarm warnings. Moreover, he allows these data to be commutated to emergency departments of Vestfold hospitals in case of a critical situation. Now Jack can better monitor the status of his father. The users of *uStore Inc.* portal enjoy functionality of the offered services that

- support user interaction by e.g. allowing to enter information and get correspondingly adjusted service;
- support auditing trails that allows e.g. to support client monitoring service;
- are easily pluggable;
- provide possibility to communicate for the users of the portal and get service feedback of other users.

Peter began to explore his services and posted on the Forum of *uStore Inc.* portal his experience with nutrition service giving an enthusiastic example how it was being adjusted if you got sick. He also put a post commenting that he got a new emergency notification service. Anthony was also monitoring the forum of uStore Inc. portal to learn about client experience and opinion about nutrition service. He was happy to see positive reaction of Peter. Both users and service providers appreciate the fact the uStore Inc. portal supports

- possibility to provide service feedback for users;
- possibility to receive service feedback for service providers.

Ben Stiller works for national welfare organization in social service department. His department was experiencing problems due to the fact that the number of people who require social or health care had increased dramatically, while there was a lack of funding and personnel to deal with it. He learned about AAL solutions and started to monitor the developments of AAL technologies at the uStore Inc. portal. Seeing how this *uStore Inc* portal helps to promote well-being of individuals in need, he recommended the local government to mandate the usage of universAAL platform for development and provision of AAL services. His proposal got accepted. Ben is actively using *uStore Inc*. portal to share the experience of deploying AAL services, to collect and analyse user experience of supported service, so that he can advise the best practices to the local government. He also provides an active feedback to the service and platform providers with respect to the possibilities for the quality improvement from government and user perspective, and even puts requests for new services, which the community would benefit from. The *uStore Inc*. portal is constantly in development improving existing and providing new AAL services, since it

- collects feedback and requests for its services;
- supports communication with AAL Service Provider community.

In this whole scenario, RA plays an important role as in each of the stakeholder (e.g. end user, health service provider, community welfare personnel) has used the underlying RAS provided. Using the instantiation process in RA, specific concrete system architecture can be built to provide those RAS to meet the requirements. Note that the ecosystem is flexible and can adapt to new actors and their needs and requests. And this can be achieved by the steps described in chapter 507.

7 Instantiation and consolidation process from RA to CA

The Reference Architecture provides the mechanism to capture system constraints to deal with legacy components, resource, time and technical capabilities. The purpose of this chapter is to provide a guideline for a methodology to instantiate a Concrete Architecture (CA) from the Reference Architecture (RA) we have presented. Thus the instantiation process from Reference Architecture to Concrete Architecture includes implementing specific architecture by decomposing into implementation specific components and by identifying actual relationships from the high level relationships provided by Reference Architecture. The instantiation process from RA to CA includes the identification of significant components with components' responsibilities, interactions and control and data flow. This instantiation process also identifies the non-functional architectural concerns like security, reliability and other crosscutting concerns. From the reference architecture high level services from different stakeholders and high level interactions among them, the instantiation process begins with identification of significant services. As a whole the instantiation process from Reference Architecture to Concrete Architecture includes the following possible steps.

1. Identification of specific stakeholders for the target Concrete Architecture
2. Identification of specific interconnection and interaction among the stakeholders for the target Concrete Architecture
3. Apperception of reference use cases and stakeholders' interest
4. Designation of reference architecture services using the mapping with RUCs and high level requirements
5. Decomposition of services for Concrete Architecture
6. Atomization of component model for Concrete Architecture
7. Concrete deployment scheme for Concrete Architecture

The instantiations that are needed to transform an abstract Reference Architecture into actual systems begin with instantiating a system architecture based on the Reference Architecture. This system architecture is used to design and engineer the system, resulting in description of how the system can actually be ordered, assembled and tested. The design and engineering effort not only provides constraints on architectures but also opens opportunities.

The consolidation process plays an important role besides the instantiation. Using the experiences from the existing AAL solutions provides a high relevance for the RA. Specific solutions from subsisting concrete AAL architectures are analysed and this will lead the further refinement of the RA with possible new requirements, stakeholders and their expectations and refined building blocks.

The instantiation process includes identification of significant components and connectors with component responsibilities, interactions and control and data flow. The communicating architectural details are constructed by using the reference architecture services from technical and non-technical stakeholders' point of view. The use case, the requirements and the mapping between the RUCs, requirements and Reference Architecture Services (in our case, these mappings are presented in Appendix A and Appendix B) provide a process for the identification of system properties for system evolution, system load and portability. In Concrete Architecture, the Reference Architecture is attained by replacing the mechanisms envisaged in the Reference Architecture with actual standards and specifications. For example, a Concrete Architecture may specify that the run-time environment deployed on the hosting nodes will be the Web Services Application Framework, and that a number of

specific communicating Web Services will implement the 'runtime orchestration of services' functional component.

The phases in the instantiation process

Phase 1: As reference architecture provides the business context with stakeholders and their expectations, the first step of the instantiation process starts with identification of specific stakeholders for the target system. From the AAL value network provided by universAAL reference architecture, the relationships and the interactions of the stakeholders are identified for the target Concrete Architecture for AAL.

Phase 2: This leads to the next step with identification of use cases related to those stakeholders. As the Reference Architecture provides the mapping between RUCs and RAS (reference architecture services) and also the high level requirements and RAS, the next step of the instantiation process discovers the RAS provided by Reference Architecture.

When the target stakeholders are identified, Reference Architecture provides their service capabilities. Using these service capabilities, an architect can design a concrete architecture by developing more concrete service decomposition for Concrete Architecture. This associates atomization of services with the identification of low level requirements, configuration properties and technical features of the target system.

Phase 3: The next step of the instantiation process includes looking at the component model of Reference Architecture. Using this component model, the Concrete Architecture is built by implementing and decomposition of the specific components for the target system. This leads to the need for identification of the services exchanged within the components identified in the previous step. By using the service collaboration patterns, the instantiation of Concrete Architecture includes the specific implementation of the services and the interaction among different components.

Phase 4: When it comes to the deployment of the services, the Reference Architecture provides the high level process of deployment. So the target Concrete Architecture instantiation accommodates those high level deployment scenarios into concrete ones by replacing the mechanism with actual services for deployment. Last but not the least, the sample ecosystem in Reference Architecture provides example scenarios for the use of Reference Architecture in different scopes with different perspective. Designing the Concrete Architecture from Reference Architecture is facilitated by this sample ecosystem which includes the mapping of the Reference Architecture to some specific ecosystem. In the following chapter, a sample ecosystem is described from two different stakeholders' perspective and hence the use of RA is depicted specifically.

8 Architecture Compliance

This section explains how other architectures intended to model AAL platforms can be compared to the Reference Architecture and in what measure they can be compliant with it. It is important to stress the fact that the **RA can be compared to other platforms** and not to specific applications or services.
As universAAL has a very wide scope, it is to be expected that other architectures will not easily implement all the functionalities designed in universAAL. Nonetheless a formal compliance assessment is necessary to identify the level and easiness of integration of other platforms with universAAL.
Given the level of abstraction of the RA it is not possible to base the compliance on specific standards or technologies, but it has to be done on the general principles of the architecture and the functionalities. Two levels of compliance have been identified and described as follows.

Level 1: scope and stakeholders compliance

At this level the comparison among architectures is done on the very general purpose of the platform being analyzed. The compliance check has to answer the following questions:

- What is the generic scope of the platform? In what domain does it fall?
- Who is the platform intended for? Who are its stakeholders?
- What concerns does the platform address?

Regarding the scope of the platform, it has to be specified what main application fields it covers. There are different views that are related to different aspects of the AAL ecosystem: as reflected by the RA and also the RUCs, the platform under analysis can fall into one or more of these categories: **Runtime Platform**, **Community Platform**, and **Developer Platfor**m. Each category reflects different functionalities, as well as different technical and organizational issues.
A tool for checking compliance can be based on a scoring sheet like the one represented below:

Type of platform	**Level of compliance as regarding the scope (0: none, 5: completely matching)**	**Comments and characteristics that do not match**
Runtime Platform		
Community Platform		
Developer Platform		

Other scope can be defined for a certain platform which may not fall into these three categories; in this case they should be listed and described carefully.
Regarding the stakeholders it has to be specified what of the following categories are covered, as already described in section 3:

- Platform stakeholders
  - Runtime Platform Provider
  - Developer Platform Provider
  - Community Platform Provider
- Client stakeholders
  - End users
  - Developers
  - AAL service providers
- Authorities

Another possible view of the stakeholders' model is the one offered by the AALiance

- Primary stakeholders.
- Secondary stakeholders
- Tertiary stakeholders

When specifying stakeholders it is useful to address also what concerns are covered by the system under analysis. Within universAAL, concerns are described in Deliverable 1.1.
As with the scope, a checklist can be implemented on a table like the one below:

Stakeholder	Concerns addressed	Level of compliance as regarding the stakeholders (0: none, 5: completely matching)	Comments and characteristics that do not match
Runtime Platform Provider			
Developer Platform Provider			
Community Platform Provider			
End users			
Developers			
AAL service providers			
Authorities			

Level 2: services compliance

The functionalities of the universAAL RA are described as Reference Architecture Services or RASes. The most detailed way of checking the compliance among the RA and another platform architecture is to go through all the RASes and identify the level of compliance. The questions to be answered at this level are:

- What categories of RASes does the platform under analysis support?
- What concrete RASes does the platform support? To what extent?

The tool for checking the compliance can be based on table as follows:

RAS category	RAS	Description	Compliance level as regarding the RAS (0: not supported at all, 5: completely supported)	Comments and characteristics that do not match
Platform-to-Platform services	1.1 Remote maintenance and configuration	Allow remote Configuration, management and maintenance of software installed in AAL spaces		
	1.2 Audit data management	Allow acquiring usage data about service usage in AAL spaces		
	…	…		
Platform-to-Client	1.4 User interaction	Facilitate user interaction with the		

services	management	runtime platform, e.g. by providing consisting look and feel and adaptability		
…	…	…		
Peer services	1.11 Communication	Allow generic communication among different types and instances of runtime platform		
…	…	…		
Platform-to-Platform services	2.1 Publish development tools	Upload, publish and advertise development tools, training courses, related documentation, etc.		
…	…	…		

As extra columns it would be interesting to also specify what specific technology, standard or interoperability mean is adopted for every RAS. The checklist can be also limited to RAS categories for a simpler analysis.

9 Validation of Reference Architecture

The purpose of this chapter is to provide a validation of the Reference Architecture using the mapping among Reference Use Cases (details in D1.1) that evince stakeholders' expectations and Reference Architecture in terms of Reference Architecture Services (RAS) because the reference use cases from D1.1 are used to generate a complete set of requirements and this reference architecture is based on the concerns of a specifically identified set of stakeholders. Using this mapping of RUC and RAS, the reference architecture can identify the mechanisms, patterns and approaches that are in common across implementations of the reference architecture. The requirements (details in D1.2) represented by these use cases provide a set of measurable constraints on the architecture by which conformance can be determined. Not all RAS will be responsible for all the requirements in all of these RUCs. However, any system that may want to leverage this reference architecture should have some overlap with at least a subset of these RUCs and High level requirements (RC).
From another perspective (which uses the top down approach), the RA is validated against the concepts and concept maps from Reference Model (RM) (Section 9.3). This includes a mapping of RM concepts and RA services that build the behavioral aspects of RA. As there is many to many relationship among Building blocks (bottom up approach) and RAS (top down approach), it also suffices that this mapping of RM to RAS implies implicit mapping of RM to Building blocks with a many to many relationship too.

9.1 Verification of Reference use cases addressed by the Reference Architecture

In order to make stakeholder expectations more tangible and understandable to multiple parties, and make them easily mapped onto a technical architecture, the developed set of Reference Use Cases (RUCs) demonstrate, in an easy-to-understand way, what specific expectation of a stakeholder means and how it can be supported by AAL technologies. Three categories of RUCs have been defined in D1.1:

- Category 1: A platform for enabling innovative and commercial grade AAL services. These are use cases that show how a platform can support higher QoS for AAL services at runtime.
- Category 2: A platform for creating a market place for AAL services. These are use cases that show how a platform can promote an online community for AAL stakeholders, and remove barriers for uptake of such services through promoting collaboration and business.
- Category 3: A platform for supporting developers to innovate in the AAL market. These are use cases that show how the platform can help building a knowledgeable and capable community of developers who can produce high-quality AAL services in collaboration with users and service providers.

There is a mapping of the three categories onto the three platform stakeholders as we believe platform stakeholders wish to specialize in their own area of expertise. Category 1 maps to *Runtime Platform Provider*, category 2 to *Community Platform Provider*, and category 3 to *Developer Platform Provider*. However, the RA also considers collaboration *among* the three platform stakeholders. Appendix A shows an overview of which RUCs are supported in the RA.

As it is shown in Appendix A, all RUCs are supported by one or more RAS, indicating the high degree of alignment between RUCs and the resulted Reference Architecture.

9.2 Verification of Reference Requirements

Due to consolidation of existing concepts in several parallel threads for producing early results, it might seem that there will be no controlling entity behind the scene that would force the fitness of the results for their intended purpose. Nevertheless, the project does not trust the results blindly and has agreed on a general approach for consistency check and verification of "fitness for purpose". For this purpose, the consolidated set of requirements from D1.2 is taken as the driving force. Besides, consolidation is not a fully mechanical way of dealing with information and knowledge; on the contrary, the analytical judgement of the engineers is always playing an important role in this process.

In the case of the component view, this would mean that the requirements are used to *verify* the system decomposition by mapping them to the identified components based on assumptions about the set of requirements that each component is supposed to generally be able to fulfil. The first case should lead to a double check to see if any new component must be added or the expectations from building blocks should be completed, and the second case should trigger the process of checking if the component can be eliminated for the sake of simplicity.

Layer	Expert group	Building Block	Subcomponent / Feature	Req	Req	Req	Req	Req	Req
Middleware	Middleware	Container		R2.2.05.07	R2.2.05.08	R2.2.05.04	R2.2.05.01	R2.2.06.1	R2.2.06.
		Communication		R2.2.01.01	R2.2.01.02	R2.2.01.03	R2.2.01.04	R2.2.01.05	R2.2.01.
		Discovery & Peering		R2.2.01.12	R2.2.01.14	R2.2.01.15	R2.2.01.04	R2.2.03.3	R2.2.05.
			Specific Protocol	R2.2.01.07	R2.2.01.16				
			Bridging	R2.2.01.14	R2.2.01.15	R2.2.01.04			
Generic Platform Services	Gateways and communication with external world	AAL Space Gateway		R1.3.12	R2.1.1.16	R2.1.2.21	R2.1.3.02	R2.1.3.08	R2.1.8.1
	service infrastructure	Service Management		R1.1.1	R1.3.01	R2.1.6.7	R2.1.7.01	R2.2.19.5	R2.2.01.

Figure 24: The table used for mapping building blocks to requirements (cropped both vertically and horizontally)

The mapping was performed in two iterations, with the first one being an initial view and the second being an additional verification. A tabular format proved to be useful for this task. As expected, all building blocks could be connected to some requirements. Some requirements, however, didn't match any distinct building block for different reasons, e.g.:

- Some requirements can be evaluated in the context of the other models in the component view, which will be the *System Information Model*, the *System Collaboration Model*, and the *Component and Interface Specification Model*. For example, the requirement *R1.1.3* about "Minimalism: At the LIF[8] only objects and functions should be visible, that are required for the interaction with other subsystems" is related to the *Component and Interface Specification Model*. These requirements will be taken into consideration when the corresponding model is specified.
- Some requirements are valid for all components, e.g., *R1.3.13 (Off-line operation: For running the system internet connection should not be required all the time)*.
- Some requirements are only verifiable in the context of a specific application, e.g. *R2.1.7.06 (No interference with life sensors: (Wearable) Sensors must not interfere in the common behaviour of the user)*.

[8] Linking Interface. A Term borrowed from the GENESIS project, defined the following way: "A job provides its real-time services, and accesses the real-time services of other jobs by the exchange of messages across its Linking Interface (LIF). These messages have to be fully specified in a LIF specification which consists of an operational specification and a LIF service model specification."

It is worth to mention that this verification has provided us also with some important insights:

- Figure 8: The consolidated decomposition model of the Runtime support platform at the level of abstract building blocks
- , enumerates sometimes "features", such as *publishing*, *subscribing*, or *brokering*, for the building blocks that were added to reflect important expectations. Although these features are necessary for a component's functioning, some of them could not be mapped to any requirements. This can be an indication that these "features" are related to the internals of those components and hence might not find any reflection at the requirement level, unless we relate them to the more generic requirements, such as separation of concerns.
- On the other hand, for a few requirements, where it was possible to assume that they could be satisfied by a pluggable component, it was not decidable if this pluggable component should be classified as a "pickable" manager or an AAL application because application-level interoperability is supposed to be supported anyhow (cf. the case of "indirect intra-layer communication through the middleware"). Quite a few of such cases have been encountered, like *support for VOIP*, *Fall handler* and *Health Manager Assistant*. This caused discussions about the necessity of differentiating "pickable" managers from applications. For example, according to one opinion, every component that does not belong to the core set of managers and is not mandatory can be classified as an application. However, finally it was acknowledged that under the premises of the collaboration model (see next section); such a differentiation has only a logical value anyhow and will not have any practical consequences for the architecture of the system.
- Finally, it was found out that some of the requirements that can be mapped to all components, e.g. *R2.2.09.08 Standardized Test-Interfaces*, fit perfectly the Container building block, thus relieving "all components" from fulfilling the requirements because the Container is supposed to force them due to the expectation to provide control over the execution model of the other components in the architecture.

In the case of the collaboration view, this would mean that the requirements are used to *verify* the service collaboration patterns by mapping them to the identified RAS. The results show that all the high level requirements that are identified in D1.2 are supported by at least one Reference Architecture Service (RAS). As high level requirements are met, it also implies that corresponding Technical Requirements (TR) are also supported by RAS. As for example, the high level requirement in '*Reliability and Safety*' category (RC9_C1) is supported by RAS#1.9 which is the *Runtime orchestration of services* provided to the Community Support Tool. The complete analysis can be found in Appendix B.

As conclusion, it can be summarized that the Reference Architecture covers appropriately the consolidated list of requirements, ensuring the consistency between these two project results.

9.3 Verification of the abstraction process: mapping of RM to RAS

In terms of instantiation (i.e. the top-down approach), to validate the Reference architecture the mapping from the RM concepts to the RA services can be considered. The direct qualitative property that follows from this method is coverage, a method to measure the relationships between RM concepts and RA services. Indirectly this provides a qualitative property for deductibility and consistency. The primary relationships that are considered are the "instantiates"-relationship and "is abstraction of"-relationship. For the analysis, the RM concept map terms and descriptions are used. For the RA services the names and descriptive key lines are used. The previous results in explicit matches. Where reasonable also synonyms and strongly related words and concepts are counted as matches, thereby providing more implicit matches. Figure 25 provides a summary of the analysis. Detailed results are provided in Appendix C.

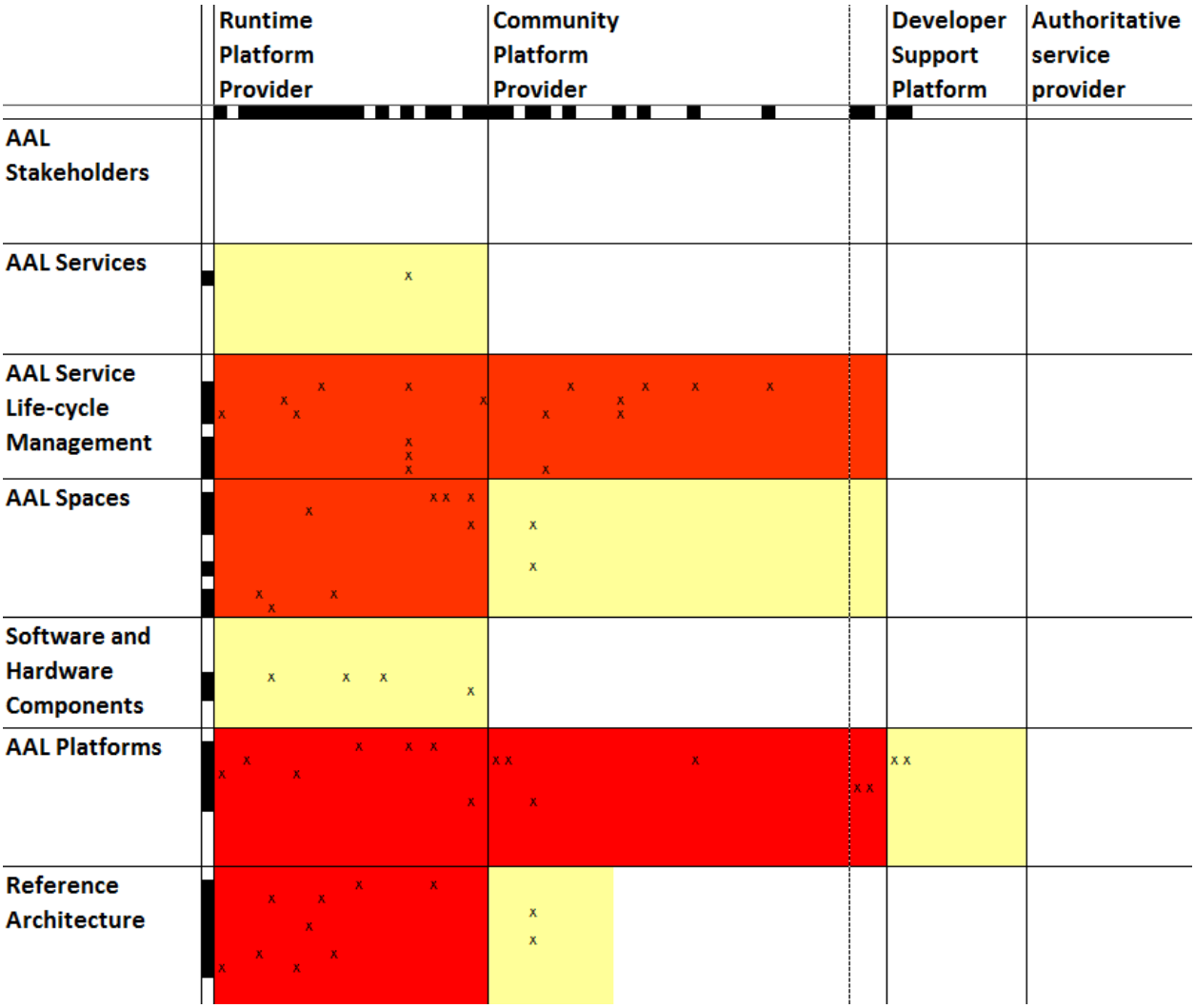

Figure 25: Summary of mapping RM concepts to RA services

The main result of the mapping analysis is that there is a good relationship between the RM concepts and RA services, which positively validates the Reference architecture. This is supported by the many "X" marks in the summary table and details table. One may notice that the mapping has a number of RM concepts and RA services categories where mapping relationships are concentrated, which are indicated through a red color in the figure. This conclusion however also comes with a few critical remarks: a significant number of RM concepts as well as RA services do not hold a "is abstraction of"-relationship or "instantiates"-relationship respectively with the other, which can be recognized in the figure and table through a white instead of black block behind respectively below their names. Closer inspection of this leads to the expectation that this is mainly a matter of rigor and not of architectural inconsistencies.

10 Conclusion/Further Work

The RA is the result of two main convergent processes: on one hand a top-down process from the RM in order to introduce technical contents with a reduced gap of abstraction and, on the other hand, a bottom-up process from the CA that has the main goal to provide a generalized view at the universAAL architecture. This last version of the deliverable has completed the abstraction process from the Concrete Architecture (D1.3 part IV), ensuring that appropriate level of abstraction has been provided in each part. Also, a consistency check with both RM and CA has been performed together with verification of compliance with RUCs and Requirements demonstrating that the resulting Reference Architecture is consistent and coherent with all those other project results.

The future work relies in wide spreading this result in the AAL Community and to start the process of standardization of the Reference Architecture in the appropriate bodies, to ensure its adoption by relevant stakeholders. This will be done in the scope of Task 1.5 and reported in D1.4.

References

[1] J. Amsden, "Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 1. Service identification," 07-Jan-2010. [Online]. Available: http://www.ibm.com/developerworks/rational/library/09/modelingwithsoaml-1/. [Accessed: 03-Feb-2011].

[2] H. Kreger and J. Estefan, "Navigating the SOA Open Standards Landscape Around Architecture," OASIS, OMG, and The Open Group, White paper, 2009.

[3] G. Van Den Broek, F. Cavallo, and C. Wehrmann, Eds., *Ambient Assisted Living Roadmap*. IOS Press, 2010.

[4] H. Chesbrough, "Open platform innovation: Creating value from internal and external innovation," *Intel Technology Journal*, vol. 7, no. 3, pp. 5–9, 2003.

[5] J. F. Moore, "Predators and Prey: A New Ecology of Competition," *Harvard Business Review*, no. May, 1993.

Appendix A. Mapping between Reference use cases and RA services

This appendix provides the mapping between RUCs and the RA services that provides support to realize them. The RAS definitions are provided in Section 5. For detailed description of each RUC please consult D1.1.[9]

Cat.	RUC ID	RUC description	RA services that provide support
1	RUC#1	Supporting rich human computer interaction	RAS#1.7, 1.24
1	RUC#2	Supporting intelligent context management and hardware abstraction	RAS#1.28,
1	RUC#3	Enabling system driven interaction	RAS#2.17
1	RUC#4	Supporting continuity of care in different AAL spaces	RAS#1.8
1	RUC#5	Supporting end user security and privacy	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47
1	RUC#6	Supporting installation, configuration and management of platform components	RAS#2.4, 1.24, 1.21, 1.2, 1.20
1	RUC#7	Supporting remote/local operation and provision of AAL services	RAS, 1.20, 1.4, 1.2
1	RUC#8	Supporting multi-user AAL services in each AAL space	RAS#1.4, 1.23, 1.8
1	RUC#9	Supporting interfacing with existing information systems	RAS#1.28
2	RUC#10	Supporting service providers in offering AAL services	RAS#2.13, RAS#2.18, RAS#2.19
2	RUC#11	Allowing users to easily find and acquire AAL services	RAS#2.4, RAS#2.5, RAS#2.7, RAS#2.9, RAS#2.11
2	RUC#12	Supporting exploitation of different business models	RAS#2.9, RAS#2.15, RAS#2.19, RAS#2.23, RAS#2.24

[9] It should be kept in mind that the list of RUCs presented here is the result of an extensive consolidation process where scenarios and requirements from a large group of EU projects (the so-called "input projects" in universAAL terminology) were analyzed and generalized into RUCs. Each individual input project has had its own research methodology for arriving at its scenarios and requirements. In this way the list of RUCs here is the result of extensive research done earlier, and not presented in this deliverable.

Cat.	RUC ID	RUC description	RA services that provide support
2	RUC#13	Capturing and utilizing user feedback	RAS#2.8, RAS#2.10, RAS#2.12, RAS#2.20, RAS#2.22, RAS#2.25, RAS#2.26
3	RUC#14	Supporting rapid development of AAL services	RAS#3.1, RAS#3.2
3	RUC#15	Model-based development of AAL services through integrated model transformation tools	RAS#3.2
3	RUC#16	Supporting online elicitation of requirements and collection of runtime feedback from users of AAL services	RAS#2.8, RAS#2.10, RAS#2.20, RAS#2.22, RAS#2.25, RAS#2.26
3	RUC#17	Supporting advanced search, reuse and sharing of service components and developer resources	RAS#3.1, RAS#3.2, RAS#3.4
3	RUC#18	Supporting utilization of personalization capabilities of AAL services	RAS#1.21, RAS#2.4

Appendix B. Mapping between high level requirements and RA Services

This appendix shows the relationship between Reference Use Cases, defined in D1.1, High level requirements, available in D1.2, and the supported RA services. The high level requirements are marked with an 'R' preceded by requirement category abbreviation mark 'RC'. As for example, 'RCx_Ry' denotes the high level requirement 'y' for requirement category 'x'. The detailed description of the high level requirements is available in D1.2. This table shows that all the high level requirements that are identified in D1.2 are supported by at least one Reference Architecture Service (RAS).

Requirements Category	**High level requirements**	**Reference Architecture Service (RAS) that provide support**
1 Context Information and Context Management	RC1_R1	RAS#1.7,1.24,1.28
	RC1_R2	RAS#1.7,1.24,1.28
2 Transparency, Privacy and Security	RC2_R1	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47
	RC2_R2	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47
	RC2_R3	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47
	RC2_R4	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47
3 Support for Designing Human-Computer Interaction	RC3_R1	RAS#1.7,1.24,1.9,2.17
4 Service Orientation	RC4_R1	RAS#1.28,1.9,2.17, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC4_R2	RAS#1.28,1.9,2.17, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC4_R3	RAS#1.28,1.9,2.17, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
5 Interoperability with Legacy and Remote Components and Systems	RC5_R1	RAS#1.28
	RC5_R2	RAS#1.28
	RC5_R3	RAS#1.28
6 Configuration, Personalisation and Maintainability	RC6_R1	RAS#1.9,2.17, 1.5, 1.8, 1.9, 1.38, 1.39, 1.47, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4, 1.20, 1.4, 1.2,1.23
	RC6_R2	RAS#1.9,2.17, 1.5, 1.8, 1.9, 1.38, 1.39, 1.47, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4, 1.20, 1.4, 1.2,1.23
	RC6_R3	RAS#1.9,2.17, 1.5, 1.8, 1.9, 1.38, 1.39, 1.47, 1.16, 1.2, 1.20, 1.21, 1.24,

		2.4, 1.20, 1.4, 1.2,1.23
	RC6_R4	RAS#1.9,2.17, 1.5, 1.8, 1.9, 1.38, 1.39, 1.47, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4, 1.20, 1.4, 1.2,1.23
7 Execution Environment and Component and Application Container for Heterogenic Devices and Operating Systems	RC7_R1	RAS#1.9,2.17,1.8, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC7_R2	RAS#1.9,2.17,1.8, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC7_R3	RAS#1.9,2.17,1.8, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC7_R4	RAS#1.9,2.17,1.8, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
	RC7_R5	RAS#1.9,2.17,1.8, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4
8 Shared Communication Infrastructure	RC8_R1	RAS#1.7,1.24
9 Reliability and Safety	RC9_R1	RAS#1.9,2.17, 1.16, 1.2, 1.20, 1.21, 1.24, 2.4

Appendix C. Mapping between RM concepts and RA services