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

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 Supplementary Report, 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.*
	X	Project Supplementary Report (single document, no sub-parts).
		Sub-part of a Supplementary Report.

Document Identification			
Deliverable ID:	**D1.2-A**	Deliverable title:	AAL Reference Architecture requirements
Release number/date:		V1.0 24.06.2010	
Checked and released by:		Sergio Guillén / ITACA	

Key Information from "Description of Work" (from the Grant Agreement)	
Deliverable Description	*Specification of consolidated requirements coming from original projects and identification of new requirements not taken into account before. The deliverable will include: A consolidated version of Technical Requirements for AAL systems, a list of functional and non-functional requirements to be taken into account in the reference architecture and a glossary of Technical concepts.*
Dissemination Level	PU=Public
Deliverable Type	R = Report
Original due date (month number/date)	Month 3 / 30.Apr.2010

Authorship& Reviewer Information	
Editor (person/ partner):	Saied Tazari / Fh-IGD
Partners contributing	CERTH, CNR, ENT, IGD, FZI, IBM, ITACA, ProSyst, TUW, UPM
Reviewed by (person/ partner)	Anders Kofod-Petersen / SINTEF

Release History

Release number	Date issued	Milestone*	eRoom version	Release description /changes made
A	13.Mar.10	Planned	15	The plan added as a separate sheet to the sprint management workbook of WP1 (https://project.sintef.no/eRoomReq/Files/ikt/ICT-20097-ICTAgeing/0_35809) → revision number 15 belongs to this workbook
A	22.Mar.10	PCOS proposed	3	The table of content as approved by the working group with a short description for each section
A	6.Apr.10	PCOS approved	5	The table of content is approved.
A	9.Apr.10	Intermediate proposed		skipped
A	14.Apr.10	Intermediate approved		skipped
A	28.Apr.10		6	Simple integration of all input available so far
A	2.May.10		7	Edited from the beginning of section 2 to the end of 3.2 and added the appendix
A	31.May.10		8	Edited up to the end of section 4.2.2 / beginning of section 4.3 and updated the appendix with the results of work by the req. Task Force
A	31.May.10	External proposed	9	Edited the remaining parts and checked the spellings
A	14.Jun.10	External reviewed	10	Feedback, comments, and corrections by Anders
A	14.Jun.10	External revised	10	Most of the above accepted and few comments on comments added
A	21.Jun.10	External approved	11	Latest agreements with Anders applied
A	24.Jun.10	Released	12	Technical Manager 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

universAAL Consortium

universAAL (Contract No. 247950) is a Large Scale Integrating Project *(IP)* within the 7th Framework Programme, Priority 7.1.b (ICT & Ageing). The consortium members are:

STIFTELSEN SINTEF (SINTEF, Project Coordinator) Contact persons: Joe Gorman Email: joe.gorman@sintef.no	**UNIVERSIDAD POLITECNICA DE MADRID (ITACA, Technical manager)** Contact person: Laura Bellenguer Querol Email: laubeque@upvnet.upv.es
AUSTRIAN INSTITUTE OF TECHNOLOGY (AIT) Contact person: Sten Hanke Email: sten.hanke@ait.ac.at	**CONSIGLIO NAZIONALE DELLE RICERCHE (CNR-ISTI)** Contact person: Francesco Furfari Email: francesco.furfari@isti.cnr.it
CENTRE FOR RESEARCH AND TECHNOLOGY GREECE (CERTH) Contact person: Nicos Maglaveras Email: nicmag@med.auth.gr	**FRAUNHOFER-GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V (Fh-IGD)** Contact person: Saied Tazari Email: saied.tazari@igd.fraunhofer.de
ERICSSON NIKOLA TESLA (ENT) Contact person: Ivan Benc Email: ivan.benc@ericsson.com	**IBM ISRAEL – SCIENCE AND TECHNOLOGY LTD. (IBM)** Contact person: Yardena Peres Email: peres@il.ibm.com
FORSCHUNGSZENTRUM INFORMATIK AN DER UNIVERSITAET KARLSRUHE (FZI) Contact person: Andreas Schmidt Email: Andreas.Schmidt@fzi.de	**PHILIPS ELECTRONICS NEDERLAND B.V. (PHILIPS)** Contact person: Milan Petkovic Email: milan.petkovic@philips.com
IMPLEMENTAL SYSTEMS SL (IMPLEMENTAL) Contact person: Jordi Valles Email: jordi.valles@implementalsystems.com	**REGION SYDDANMARK (RSD)** Contact person: Christina E. Wanscher Email: cew@medcom.dk
PROSYST SOFTWARE GmbH (PROSYST) Contact person: Kai Hackbarth Email: k.hackbarth@prosyst.com	**TECHNICSHE UNIVERSITATET WIEN (TUW)** Contact person: Roman Obermeisser Email: romano@vmars.tuwien.ac.at
TSB SOLUCIONES TECNOLOGICAS (TSB) Contact person: Juan-Pablo Lázaro-Ramos Email: jplazaro@tsbtecnologias.es	**VDE VERBAND DER ELEKTROTECHNIK ELEKTRONIK INFORMATIONTECHNIK EV (DKE)** Contact person: Henriette Boos Email: henriette.boos@vde.com
UNIVERSIDAD POLITECNICA DE MADRID (UPM) Contact person: cvera@lst.tfo.upm.es Email: cvera@lst.tfo.upm.es	

Table of Contents

Table of Figures

List of Tables

1 Executive summary

D1.2 reports about the requirements analysis process in universAAL and summarizes the related results. The current document, D1.2-A due at month three, is the first report from four planned versions. It deals with the initialization phase of the requirements analysis process. In this phase, a starting set of requirements are worked out based on the consolidation of requirements already identified by the input projects GENESYS, MPOWER, OASIS, PERSONA, and SOPRANO. D1.2-B due at month 12 will be the main version defining the actual universAAL *reference requirements*. Section 2 provides more details on the purpose of D1.2, its relationships to the other universAAL deliverables as well as the positioning of D1.2-A in relation to the other versions of D1.2.

Consolidation is the main motif in the universAAL project in order to benefit from the plenty of high quality engineering work already done in the field of Ambient Assisted Living. It should help to reach consensus in the research community and a wider acceptance in the general public based on the provision of a set of converging conclusions.

To assure ourselves of the quality of the input set of requirements, we first checked the requirements analysis process in each of the input projects and the used engineering tools and techniques. The results of this survey are summarized in Section 3.

Section 4 is the main part of this report that describes the engineering works in universAAL towards the provision of a first consolidated version of the requirements in AAL. It introduces the ARCADE-based methodology for collecting the input requirements in a harmonized way, mapping them to each other, and providing a first prioritization of the resulted set of requirements. The output of this process is included in form of a table in the appendix of this report.

The report concludes with some advice for the next phases of work, especially towards D1.2-B with the AAL reference requirements.

2 About this document

2.1 Role of the deliverable

Deliverable 1.2 defines and presents the universAAL Reference Requirements and the process of requirements collection and consolidation.

Requirements act as the central point of reference in software development like a contract between the involved parties. In essence, they capture the functional and non-functional expectations from the target system based on a concrete set of agreed use cases, on one side, and the general concerns of users and stakeholders that deal with the system, on the other side. Hence, requirements glue the scenarios and use cases defined by users to the target system to be implemented by the developers. As a consequence, at the very end of software development, the success of a software system or application is measured according to its compliance to the requirements.

As the central point of reference, a collection of requirements should be prioritized, traceable, verifiable, and specified with sufficient formality. The latter ensures that developers can use them in system design and implementation. In general, requirements should be formal enough that at least one software test case can be derived from each requirement. Each requirement associated with one or more test cases is automatically verifiable; however, even without test cases, requirements must be specified without any ambiguity so that one can easily verify if the delivered software fulfils its requirements or not. Traceability, in turn, helps to always be able to check the origin of each requirement (often a certain use case) and the course of its changes at any point in time. This is important because requirements are volatile and changes must be managed continuously as different versions of the specifications evolve. Last but not least, prioritization of requirements helps to link the "wish"-based definition of requirements to the plan-based development of versions of software. Prioritization is usually the result of a consensus between the urgency of the requirements from the viewpoint of the customer, on one side, and technical feasibility and project management concerns, on the other side.

The most important point about the requirements analysis process in universAAL is that it started with the consolidation of the results of work already done in its input projects instead of waiting for the set of use cases in order to extract the first set of requirements from them. Hence, the mapping between the use cases and the requirements is not given per se but must be derived when the two sets – that result from the related parallel threads – are ready for consistency check. The further steps in the universAAL requirements analysis will follow the conventional process of gap and SWOT analysis as well as change management.

The universAAL consortium has agreed on using the ARCADE architecture development framework[1] to guide development of the universAAL reference architecture in WP1. Developing the architecture according to the ARCADE approach enhances architectural consistency, enables integration of other systems and ensures that concerns and requirements of stakeholders are properly acknowledged. The ARCADE Requirements Viewpoint specifies two models for identifying and specifying requirements: the *Requirement Model* and the *Target System Interface Model*. Consequently, D1.2 specifies these two models in its different versions.

2.2 Relationship to other universAAL deliverables

Due to the fact that requirements are the central point of reference both in the architectural design and in the development of software systems, this deliverable will obviously have several relationships with

[1] See D1.1-A for a quick overview of ARCADE [7].

many other universAAL deliverables. A good starting point for enumerating these relationships is the figure used in Section 3 of D1.1-A, where also a quick overview of ARCADE is provided. According to the ARCADE framework, the requirements viewpoint provides a bridge between the context viewpoint (subject of work in D1.1) and the component and distribution viewpoints described in D1.3 (see Figure 1).

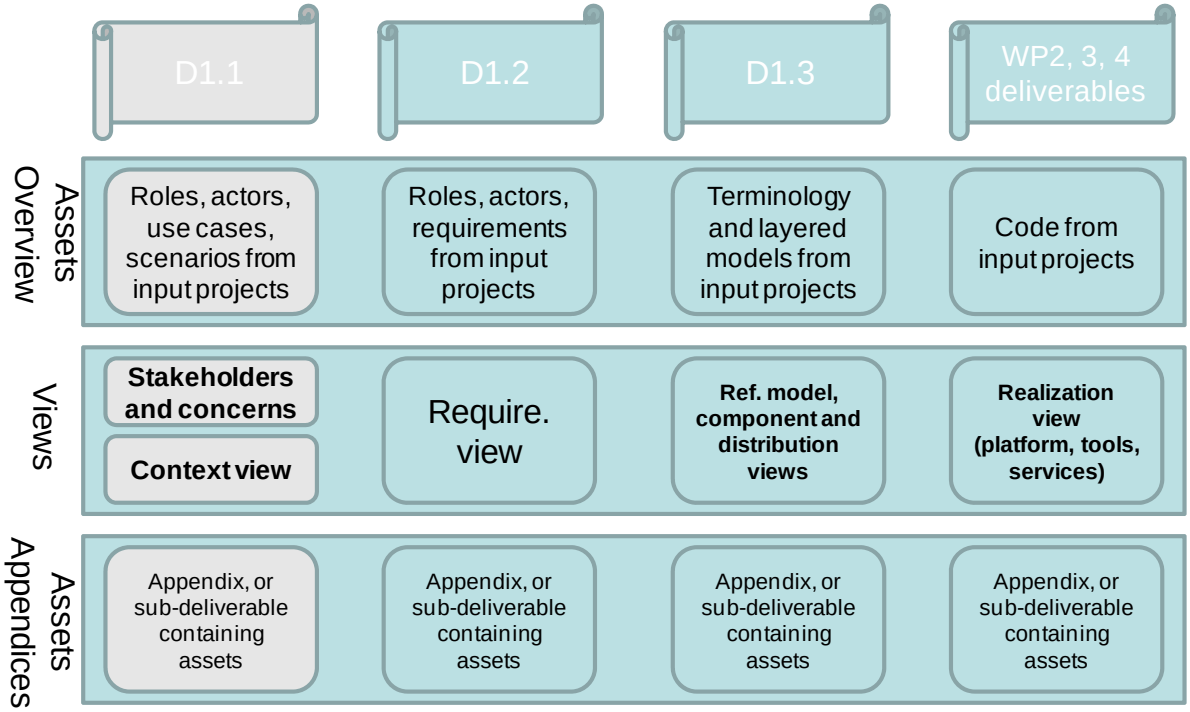

Figure 1: Organization of WP1 deliverables according to ARCADE

In particular, this deliverable is related to the following universAAL deliverables and work packages:

D1.1 – Reference Use Cases of AAL: In general, use cases and requirements are strongly connected since requirements are usually extracted from use cases in the course of analyzing the realization of the use cases. Specific to the universAAL approach, where use cases and requirements consolidation has been done in parallel, the traceability of the requirements in compliance with the use cases from D1.1 must be assured in retrospect. To achieve this, D1.1's list of reference use cases is assumed to take the lead in the process of alignment. In the case of new use cases (e.g., those related to the Developer Depot and uStore), the conventional process of extracting requirements from use cases will apply.

D1.3 – universAAL Reference Architecture: The reference architecture has to fulfil the relevant requirements from D1.2 (this deliverable), but also here, the consolidation of assets inherited from universAAL's input projects is being done in parallel. Similar to the above case, subsequent alignment and consistency check is essential in order to guarantee the coherence of the WP1 results.

D2.1 – universAAL Execution Environment / D2.2 – universAAL Platform Services: WP2 deals with the reference implementation of the universAAL reference architecture. In order to derive a concrete implementation from an abstract architecture, several design decisions must be taken. The set of reference requirements will guide these decisions. Also, all the requirements that deal with the details of the platform realization, and hence were not abstract enough to be considered at the level of the reference architecture itself, must be fulfilled by these two deliverables of WP2. Hence, it is D1.2 that plays the role of input for them. Nevertheless, a detailed consideration of runtime behaviour can lead to the identification of new requirements that need to be incorporated into D1.2 in the course of change management.

WP3 – Tools and Tutorials: WP3 provides tools for developers and service providers that choose to build on the universAAL platform. These tools enable easy development and personalized deployment of services. Past AAL-projects have not captured these aspects in detail. Therefore, work in WP3 will result in new use cases that need to be processed for extracting new requirements to be incorporated into D1.2. Such requirements will guide the development of the uStore, the Developer Depot, and all other WP3 tools.

WP4 – Innovative service concept implementation and industry cases: Most of the functional requirements from D1.2 will be related to the AAL services and applications to be realized in WP4. As D1.2 provides the reference requirements, it will strongly influence the choice of the innovative services and applications to be developed in WP4. Further, evaluation of AAL domain and market ecology will also result in new knowledge about the AAL domain that needs to be reflected in the reference requirements.

WP5 – Verification and Validation: The success of software systems is measured according to their compliance with their requirements. Hence, D1.2 will serve as the reference for evaluating all the software developed in the work packages 2, 3 and 4. However, as WP5 will verify and validate the universAAL developments in cooperation with the relevant users, it is also expected that several change and feature requests emerge so that also here new requirements may arise that have to be incorporated into D1.2.

2.3 Relationship to other versions of this deliverable

This deliverable will have four different versions to be delivered at months 3, 12, 21, and 30 that are labelled as D1.2-A to D1.2-D, respectively.

D1.2-A, due in M03, presents a consolidated and prioritized list of requirements as collected from the input projects. In that sense, it initializes the process of requirements analysis in universAAL. With regard to the ARCADE Requirements Viewpoint, D1.2-A provides the Requirement Model in its preliminary stages. The public version of this model together with the ARCADE-based Target System Interface Model will be presented in D1.2-B at month 12. In order to produce the public version of the requirement model,

- more work will be done on the harmonization and prioritization of the consolidated requirements in alignment with the parallel work on D1.1-A, D1.3-A, and D1.3-B, and
- new requirements will be identified and added to the model, based on a SWOT and gap analysis as well as in alignment with the parallel work on D1.1-B as it is expected to identify new use cases, especially related to the WP3 tools.

D1.2-B is accordingly planned as the main release of the universAAL reference requirements. Hence, the versions D1.2-C and D1.2-D will be maintenance versions in accordance with change management[2]. Additionally, we plan to consider alternative presentations of the developed models: D1.2-C, due in M21, should provide the two models to the general developer community in an appropriate format, and D1.2-D is supposed to provide a web-based marketing view on requirements at M30 by making use of more visual and dramatic elements.

[2] Recall that "requirements are volatile and changes must be managed continuously as different versions of the specifications evolve" [Section 2.1]; hence, change management is considered unavoidable in requirement engineering.

2.4 Structure of this document

As stated above, D1.2-A presents a consolidated and prioritized list of requirements as collected from the input projects. The actual result of work on the consolidation of the requirements, however, can be found only in Appendix A. The remaining of the document is structured with the goals of

1. showing the legitimacy of the universAAL decision to base its requirements analysis process on the consolidation of requirements already specified in the input projects, and
2. reporting how the consolidation work has been done.

Accordingly, the next section will show that there is no need to go through the whole process of requirements analysis from scratch. To this end, Section 3 consists of specific subsections per input project[3], each summarizing the requirements analysis process in the corresponding project.

Consequently, Section 4 reports about the work done by explaining the methodology used and summarizing the different steps in this process.

Finally, Section 5 closes this report by providing an overview of the next steps towards D1.2-B.

[3] From among the input projects with representatives in universAAL, only the Amigo requirements could not be included in this process in due time. During the gap analysis towards the version B of this deliverable, this shortcoming will be rectified.

3 Requirements Analysis Process in the Input Projects

3.1 GENESYS Requirements Analysis

The FP7-STREP GENESYS provides a cross-domain architectural framework for embedded systems development. It is a candidate for the ARTEMIS[4] European Reference Architecture for embedded systems. The development of GENESYS was driven by the challenges of ARTEMIS (i.e., compose-ability, networking, security, robustness, diagnosis, integrated resource management and evolve-ability).

In order to make the GENESYS architecture suitable for multiple application domains, requirements for GENESYS have been captured from five different application domains, namely *Avionics*, *Automotive*, *Industrial Control*, *Consumer Electronics*, and *Mobile Industry*.

The main inputs to the requirements of the GENESYS architecture are constraints and requirements, which have been defined previously in the ARTEMIS European cross-domain embedded system architecture [1]. A group of experts composed of members of each application domain used the detailed ARTEMIS constraints and requirements as a basis to derive a set of requirements for the GENESYS project. The focus of the GENESYS project was on domain independent requirements (e.g., technological constraints and challenges). In addition, domain experts contributed requirements for their respective applications and standards. Finally, all requirements were categorized and prioritized by the expert group.

The requirements have been classified into two *application-oriented* categories and four *technology-oriented* categories of requirements. *Application-oriented requirements* either originate from industrial applications (e.g., automotive, avionics and industrial control systems) or from consumer applications, with focus on product property related and product creation related requirements of mobile devices and ambient intelligence applications.

Technology-oriented requirements are common to most of the application domains and have been divided to the categories *Methodology and Tools*, *Networking along with Security and Resource Management*, *Robustness and Diagnosis*, and *Compose-ability*.

The template for requirements specification in GENESYS captured the following mandatory attributes:

- **ID:** unique identifier according to the format: <WP number>_<sequence number>
- **Name:** synonym for the ID, which is also unique.
- **Responsibility:** denotes the work packages and partners that will be responsible for ensuring the satisfaction of the requirement in the GENESYS architecture.
- **Description:** specifies the requirement. The description is expressed in a way, that the requirement becomes verifiable.
- **Rationale:** is an explanation why the requirement is necessary to achieve a specific goal of the GENESYS project.
- **Architectural Significance:** specifies whether the fulfilment of the requirement through the platform is necessary to achieve a desired goal. The following Architectural Significance options were distinguished:
  - Must be guaranteed by the platform architecture by construction

[4] See http://www.artemis.eu/.

- Must be implemented by the platform architecture
- Must be enabled by the platform architecture
- Not relevant to the platform architecture

- **Application Significance:** can either be *high*, *medium*, *low*, and *optional*. This attribute denotes the importance of a requirement for systems/products in the domain, but not for the platform. Still, the reference architecture template and the cross-domain architectural style must not be in conflict with the requirement, i.e., it must be possible to implement the requirement on top of an instantiation of the reference architecture template.
- **Integration Level:** denotes the layer in a system-of-systems at which the requirements is expressed: chip, device, and system (open or closed).

Additionally, the following optional attributes could be specified:

- **Relations:** describe how a requirement is related to other requirements (e.g., a requirement may require, imply or be in conflict with another requirement).
- **Domain:** lists the domains for which the requirement is important or cross-domain in case of a universal requirement.
- **Additional Classification:** is used to classify requirements (e.g., as physical or logical).
- **Additional Information:** can be supplied in order to help the reader to understand the requirement and its context.

The definition of requirements used in the GENESYS project began with the definition of requirements for the ARTEMIS European Reference Architecture for embedded systems in 2006, and ended in 2008 with the final agreement on the requirements for GENESYS.

During the analysis of requirements a total of 298 requirements within six different groups have been elaborated as follows:

Category	***Number of Requirements***
Industrial applications	141
Consumer applications	15
Methodology and Tools	41
Networking, Security and Resource Management	56
Robustness and Diagnosis	15
Compose-ability	30

3.2 MPOWER Requirements Analysis

The FP6-STREP MPOWER aimed at defining and implementing an open platform as a suite of independent building blocks to simplify and speed up the task of developing and deploying services for persons with cognitive disabilities and elderly. The platform-related goals of the project were to support *(1)* the integration of smart house and sensor technology, *(2)* interoperability between profession and institution specific systems, *(3)* secure and safe information management, including both social and medical information, and *(4)* mobile users which often change context and tools.

Requirements analysis in MPOWER has been based on group interviews and questionnaires prepared for the target groups in the partner countries consisting of older people in a senior's home in the Netherlands, relatives/family carers of people with dementia in Austria and Norway, and experts on older people and dementia in Austria, Poland and Norway.

The partners chose to use a recruitment procedure that was suitable for reaching the actual target group. The methods for data collection were different in each country, due to both different target groups and different resources available. For example, the experts were asked to complete a questionnaire and send it back, while family carers were invited to a group interview. The analysis was done separately for questionnaires and interviews. After the results were identified, a validation of user needs led to development of problem scenarios and activity scenarios. The activity scenarios presented ideas about possible solutions to choose, and thus created a basis for writing use cases and features. The process of writing use cases and features was an iterative process where researchers within health and social domain collaborated with computer and information technology scientists.

The partners had experiences in using different methods for data collection. For instance, the Dutch partner in MPOWER had long experience in using the OWOG-method "Over Wonen van Ouderen Gesproken" ("Speaking about living for elderly"). In Norway and Austria, the KJ-method[5] was used as a tool for data collection. Since the users were not aware of the potentials of the AAL technologies, it was important to use an iterative approach for both identifying user needs and analyzing the possible related solutions.

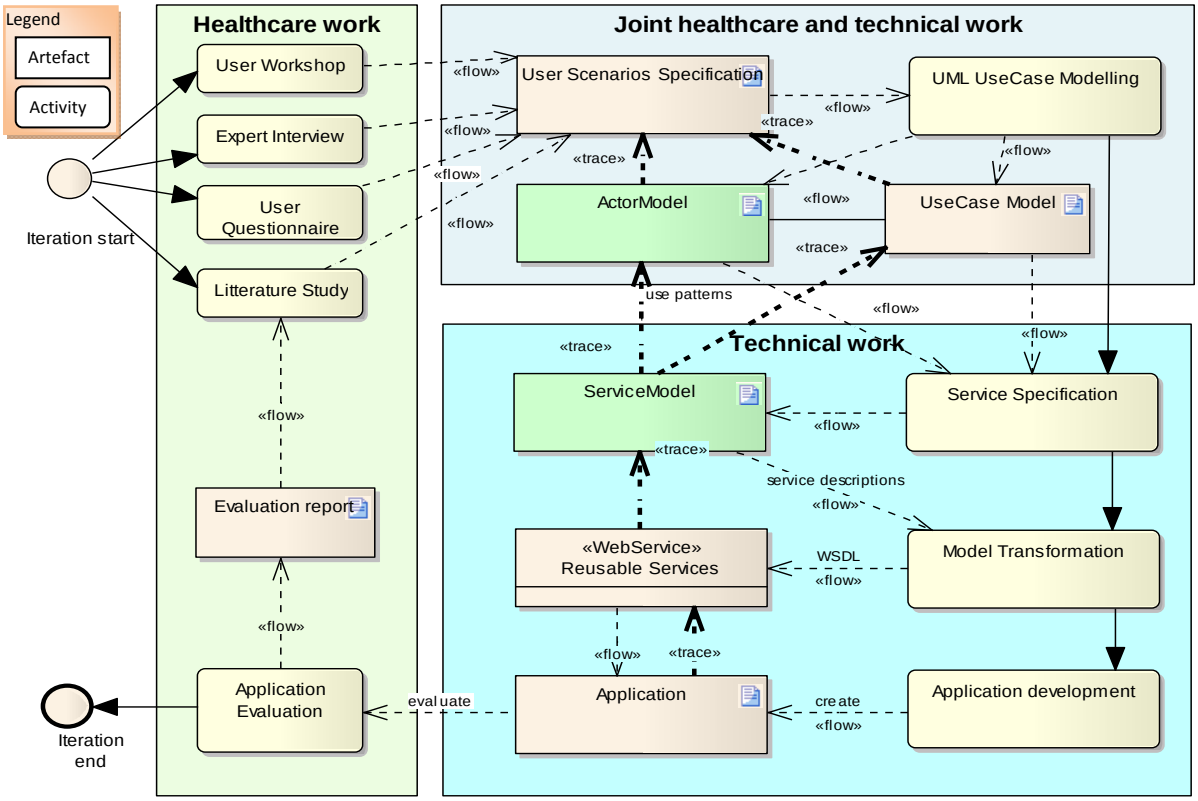

Figure 2: The iterative model-driven development process used in MPOWER to identify actors and services

A set of illustrations should now help to summarize the MPOWER requirements analysis process, internally often called "feature extraction": Figure 2 shows the overall iterative approach while Figure 3 illustrates the main activities along with the expected results, and Figure 4 the feature extraction loop. Finally, the major categories of requirements bridging between user needs and the development work is summarized in Figure 5: The inner circle summarizes the requirements for products and

[5] KJ, or affinity diagram, is a creative group technique used to organize and create relationships between apparently unrelated ideas. The name "KJ" comes from the initials of the Japanese anthropologist Jiro Kawakita.

systems for end users while the utter circle shows non-functional features and the expectations of implementing technology.

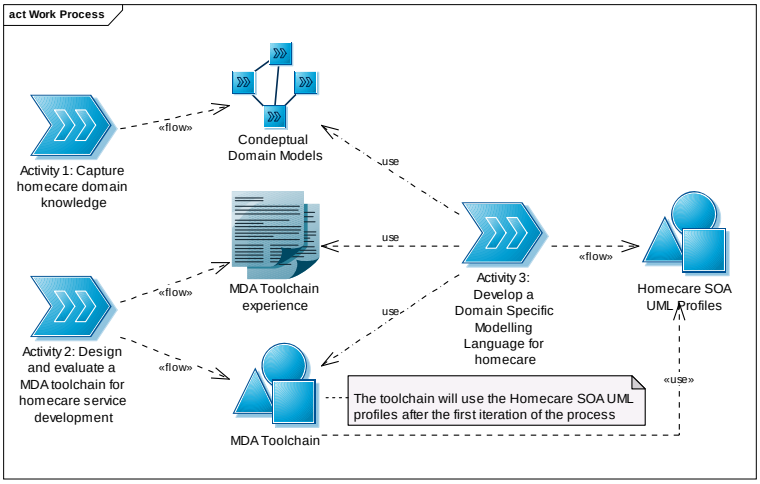

Figure 3: The three main activities in MPOWER and their work products

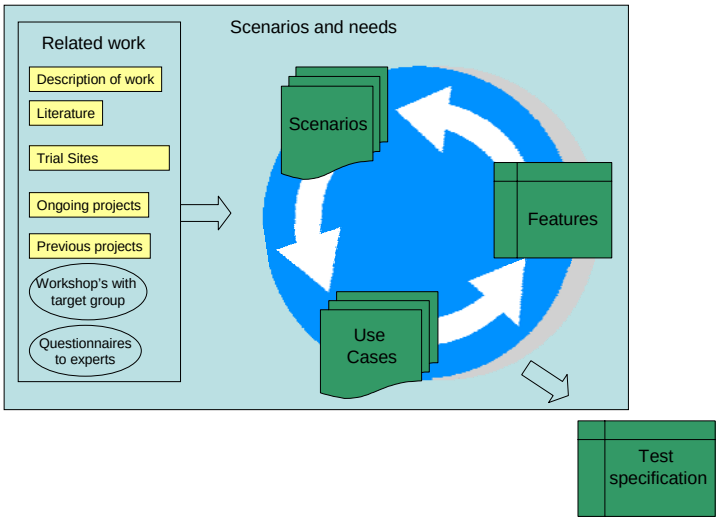

Figure 4: The process of extracting features and specifying test cases in MPOWER

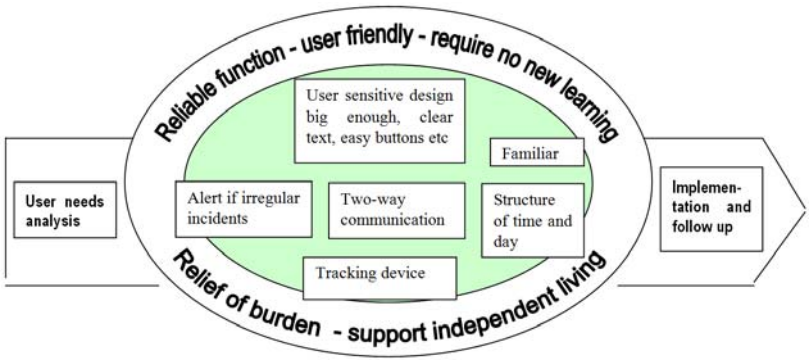

Figure 5: Categories of requirements bridging between user needs and the MPOWER development work

The main tool used for the definition of use cases and features was the Sparx UML tool called Enterprise Architect. Textually, the requirements were collected in a table that was derived from ARCADE architecture framework approach. Several categories of requirements were important; namely: viewpoint related, concern related, assets related and reference architecture requirements. Almost all of the MPOWER requirements fall into the concerns related category where application

functionality and user experience and interaction concerns make the biggest part. Requirements that describe functionalities of certain applications are mostly related to different alarms, notifications and calendar evens while requirements that describe user experience and interaction mostly refer to customized User Interfaces and different ways of interacting with system. Every requirement was enumerated according to the hierarchical numbering pattern in order to better distinguish requirements category and a concise description of each requirement is given together with the customer on whom it is related to. Since the focus of MPOWER was to support elderly and cognitive disabled in a smart-house environment most of the requirements are related to Assisted Person as a customer.

An interesting aspect of the MPOWER work on requirements specification was the attempt to map features to existing HL7[6] domains and topics in order to better re-use existing assets. Another fundamental work was the investigations on existing related work from three different sources, namely literature (relevant reports, articles and books), relevant research projects, and relevant existing systems and tools. For each of the categories a database was set-up. The literature search, for instance, was performed within medical databases, technical databases and on Google scholar. Search on the Internet was performed to identify projects and newer articles published. The abstracts for all identified articles were read and evaluated with respect to their relevance to the MPOWER project. To create easy access to the content of the articles, a database was created and the articles were categorized according to relevance and a summary was written for each.

The prioritization of the functional requirements was done based on the following criteria:

- It should reflect a challenge for family carers, for professional care givers, for service providers and for society
- It should improve everyday living for older people and people with cognitive impairments and dementia
- It should possibly lead to cost effectiveness regarding safety and security (safety value)
- It should cover a future need for society and provide improvements of care services

The final set of requirements in MPOWER consists of over 177 entries organized in over 20 categories introduced by the ARCADE Requirements Model. These categories overlap with the ones used in universAAL (cf. Table 3) therefore we avoid their repetition here.

3.3 OASIS Requirements Analysis

The FP7-IP OASIS aims at the implementation of an Ontology-driven, Open Reference Architecture and System in order to enable and facilitate interoperability, seamless connectivity and sharing of content between different services and ontologies in AAL.

In OASIS a thorough process for requirement analysis has been conducted with a set of users and user representatives. Different types of questionnaires were devised for collecting user needs and requirements based on the different sub-project topics and areas, e.g. the Questionnaire on Independent Living Applications [2] and the Questionnaire on Autonomous Mobility and Smart Workplaces applications [4].

For each one of the aforementioned questionnaire types, two surveys conducted; one for elderly and one for caregivers in five European countries (Bulgaria, Germany, Italy, Romania and Spain) and related to the approach of older adults towards the ageing. In selecting the locations where the interviews were conducted we gave the preference to those where we plan to organize the pilot sites of

[6] The global authority on standards for interoperability of health information technology with members in over 55 countries – see http://www.hl7.org.

the OASIS Project during the verification phase with the involvement – in most of the cases – of the same persons.

The interviews involved elderly living in sheltered homes, elderly living at home (alone, in pair or with relatives), and formal and informal caregivers. The involved people cover a wide spectrum of primary and secondary users by including older adults living both in sheltered homes and at their homes (alone, in pair or with relatives) and carers with various responsibilities and roles. A good mix of age, gender and education level was achieved.

After analyzing the results from the questionnaires, the following domains were identified [5]: *Designing for the elderly* with adaptability to the characteristics of the older adults as the key requirements in order to improve technology acceptance for this sector of the population, *Personalization* in interaction of disabled and elderly people with ICTs, provision of content and information search, *ICT in the ageing process* with Reassurance & Reminding, Stimulation & Enabling, Belonging & Participation, and Safety & Protection as the four clusters of needs, *Consumer behaviour* using a "gerontographic model" with the segmentation of the elderly population into four groups (see Figure 6).

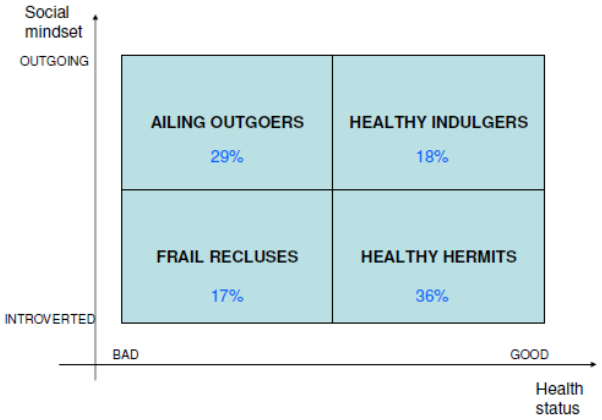

Figure 6: A model of segmentation of the older population in OASIS

The outcome of the devised methodology was the extraction 60 use cases and application scenarios for Independent Living Applications clustered in 6 categories: Nutritional Advisor, Activity Coach, Brain and Skills trainer, Social Communities platform, Health monitoring and Environmental Control. In addition, three cross-domain use cases have been defined that relate the areas of Independent Living with Autonomous Mobility and Smart Workplaces.

To facilitate the compilation of information and analysis of all the identified products, services and research projects a special database has been developed (see Figure 7). The set of the requirements were then extracted from this gathered data by analyzing the statistics and interrelationships and comparing the needs with the achievements of the AAL research so far. As a result, 141 requirements in 18 categories were identified [6].

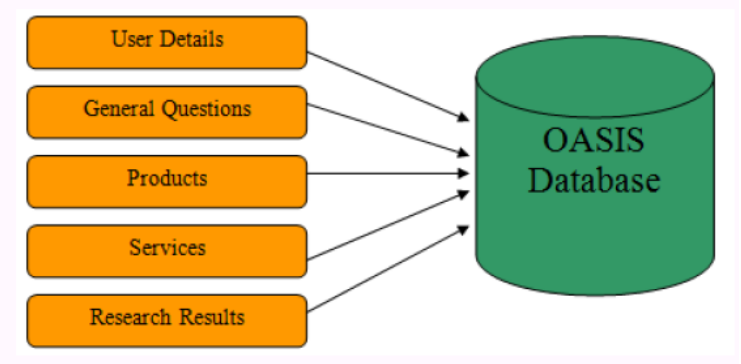

Figure 7: The OASIS information gathering database.

3.4 PERSONA Requirements Analysis

The FP6-IP PERSONA aims at providing an open and scalable technological platform that facilitates the development and deployment of a broad range of AAL services as well as developing and prototyping of example AAL services on top of the envisioned platform.

The requirements analysis process in PERSONA started in two parallel threads, one with regard to the platform and the other with regard to user requirements.

In the platform-related thread, the collection of requirements was based on investigation of scientific literature and existing solutions and studies, such as the strategic research agenda of ARTEMIS[7]. These investigations were done in the context of a series of state-of-the-art studies in the fields of architecture and middleware, context-awareness, service infrastructure, multimodality and multimedia integration, and privacy-awareness and trust. An interesting aspect here was that PERSONA, similar to universAAL, had an explicit focus on the creation of an AAL platform and hence tried to reuse existing achievements in the field as much as possible. For this purpose, the platform-related requirements were simultaneously looked at as a set of criteria for evaluating existing solutions. To this end, the consortium agreed on a small set of essential requirements in each of the above fields pretty quickly, based on own experiences from previous projects and the frequency of referring to those requirements/features in the reference material.

A more complex method was applied in the user-related thread. Here, the requirements analysis started with the creation of an overall user experience model based on the specification of users' needs in the application domains of social inclusion, assistance in daily activities, safety and security, and mobility, while considering the socio-dynamic factors that influence and shape the acceptance of this kind of services. For determining the socio-dynamic factors, an analysis of the current state of AAL services and technologies was performed focusing on the differences between European countries in general as well as in the pilot sites. The analysis included the following dimensions: socio-economic factors, socio-cultural factors, services and technologies, legal and regulatory environment and ethical aspects. The methods used were literature review for the European level, gathering relevant information from Austria, Belgium, Denmark, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Netherlands, Portugal, Spain, Sweden, and UK; and experts' questionnaires and interviews for the pilot sites, so acquiring more details from Denmark, Italy and Spain. These needs were organized according to the four domains and were linked to the PERSONA spaces (physical environments where the elderly is supported, namely body, home, neighbourhood and village/town).

The result of the work performed in this first phase was a set of high-level user requirements that were used to create 19 PERSONA scenarios, based on storylines where an elderly had solved a specific need by means of PERSONA solutions. These scenarios were used to devise the services from a business perspective and the functionalities from a user experience perspective, while using them to derive additional technical requirements. The *role of the scenarios* was to provide a bird's eye view on the project development with a user-centred design strategy.

A first prioritization of scenarios was done based on their added value and usability to the end user, against existing solutions and specific to the pilot sites, and based on its technical feasibility and system capabilities. The first one was done by elderly experts from the pilot sites while the second one was done by technical experts from the consortium.

At a second stage, mock-ups were created for the 8 high ranked scenarios to gather end-user feedback. These scenarios were evaluated at the selected pilot sites using individual interviews with elderly persons as the method to collect qualitative insights together with questionnaires to assess, in a

[7] See https://www.artemisia-association.org/sra.

standard and formal way, subjective judgments, attitudes, opinions or feelings about the scenarios presented during the interview.

The results of this assessment were used to refine the scenarios, improving the use cases derived from them as well as the service specification from the business and technical perspective. A summary of this partially iterative process is shown in Figure 8.

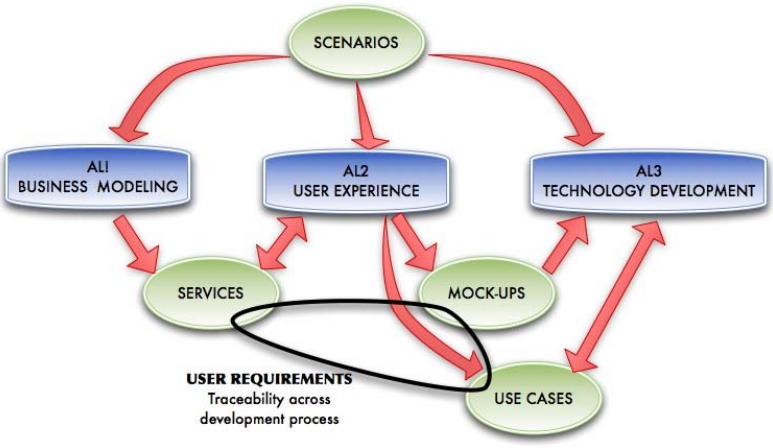

Figure 8: PERSONA user requirements analysis process

Figure 8 also shows that user requirements in PERSONA were extracted from a combined analysis of the mock-ups, use cases, and the services. It is worth mentioning that users explicitly expressed some requirements during the mock-ups evaluation process or by the domain experts during workshops. This approach had linked two main problems:

- Requirements were scattered across different information sources, and
- There was a multiplicity of ways of expressing the requirements depending on their source.

The VOLERE[8] methodology was selected due to its capability of solving those problems as it defines a common language for the expression of all requirements, combined in a single document, and it provides the ability to keep track of all requirements and its sources during the development phase.

VOLERE provides a standard classification of requirements (see the provided statistics at the end of this section for the list of used categories). Accordingly, functional and non-functional requirements were specified for each one of the available and supported PERSONA services and an adaptation on the original VOLERE template was done to accommodate their traceability across Scenarios, Services and Use Cases. In the case of the non-functional requirements, its sources were completed with the study of related European projects (Aladin, MPOWER, and West Lothian Council) and the external expert interviews and workshops, where PERSONA approach was discussed from different perspectives, such as the ethical and legal implications.

The user requirements collected in this way were then undergone a prioritization process based on their assessment from three different perspectives, using the following criteria:

- PPG1 – Business: Market size, Unique selling proposition, Cost/benefit
- PPG2 – Coverage of user needs: Independency of elderly, Facilitating process of care provision, Quality of care
- PPG3 – Service development: Degree of innovation, Effort of implementation, Complexity of implementation

[8] See http://www.volere.co.uk/.

One of the most important points when defining the user requirements was the correct definition of the *fit criterion.* That is, the measurement of the requirement that can be monitored by the system or an evaluator in order to test until which degree the solution matches the requirement. Three types of *fit criteria* have been defined that are also included in the parameters defined in the pilot sites evaluation methodology:

- Capability checked during implementation
- Performance tested in the field
- Benefit evaluated in the field from a long-term perspective

The following two tables were used for describing the functional and non-functional requirements:

Table 1: Functional Requirements Template in PERSONA

Description	Short description of the current system requirement.
Status	Status of this requirement report (e.g.: "Proposed"), Version and Phase numbers
Revision	Information about revision history *(e.g.: Created on 16/6/2008. Last modified on 4/7/2008).*
Requirement type	One of the VOLERE proposed requirement types (e.g.: «15. Security»)
Rationale	The rationale behind this requirement.
Priority	Level of priority about the fulfilment of this requirement.
Difficulty	Level of difficulty about the fulfilment of this requirement.
Supporting materials	Additional supporting elements.
Notes	Additional explanatory notes.

Table 2: Non-Functional Requirements Template in PERSONA

Requirement #	A sequential number, starting from 1 for each service.
Requirement Type	Use the full list of requirements types as described in VOLERE Methodology.
Use Cases List	The IDs of the Use Cases (UC) (e.g. UC3.1.2) where this requirement is needed.
Scenarios List	The IDs of the Scenarios (e.g. Sc3.3) where this requirement is needed.
Services List	The IDs of the Services (e.g. Sv03) where this requirement is needed.
Description	A one-sentence statement of the intention of the requirement.
Rationale	A justification of the requirement.
Originator	The source of this requirement (e.g. IR2.2.1 part I, User comment – please identify the mock-up evaluation session).
Fit Criterion	A measurement of the requirement such that it is possible to test if the solution matches the original requirement.
Customer Satisfaction / Dissatisfaction	Degree of the stakeholder happiness if this requirement is successfully implemented (scale from 1 = uninterested to 5 = extremely pleased) / Degree of the stakeholder unhappiness if this requirement is not part of the final product (scale from 1 = hardly matters to 5 = extremely displeased).
Priority	A rating of the need of this requirement in the Persona platform. The priority is a result of different contributing factors, arriving from different contexts (industrial context, business context, etc.).
Conflicts	Some other requirements that cannot be implemented if this one is. Write one requirement ID per cell (the requirement ID in this case is composed by the requirement number + the service ID, e.g. Rq03Sv07).
Supporting Materials	These are documents that may illustrate this requirement. For example other reports, scientific papers or case studies.

Created	Date of creation followed by the author's name.
Updated	Date of update followed by the author's name.

The collaborative work during the VOLERE-based analysis of user requirements was organized using Google docs and Google groups. After the requirements were validated, they were collected in an excel file. Non-functional requirements were collected using the Enterprise Architect CASE Tool that provides the documentation of the service requirements.

The gathered and validated user requirements were processed by 19 contributors and assigned to 19 single services, and an additional group of requirements, mainly Cultural and Political Requirements (Requirement type 16) and Legal Requirements (Requirement type 17). After a consolidation phase, conducted by 4 contributors, a total of 202 requirements within the 10 requirements types remain:

Category	***Number of Requirements***
09. Functional requirements	66
10. Look and feel	14
11. Usability	38
12. Performance	24
13. Operational	20
14. Maintainability and Support	6
15. Security	10
16. Cultural and Political	3
17. Legal	17
18. Business	4

3.5 SOPRANO Requirements Analysis

The FP6-IP SOPRANO is developing supportive environments for older people based on the concept of “ambient assisted living”, using pervasive ICTs to enable older Europeans to live independently in their own homes. A feature of the Experience and Application Research (E&AR) methodology used in the project is the active and strong involvement of older users throughout the entire R&D process. Participative methods in the area of research and development enable to thoroughly focus on the users when defining the user requirements, iteratively generating design solutions and evaluating those design solutions in real life settings.

Analysis of user needs is, beside market analysis and implementation of a technical platform, one of the major SOPRANO goals. Defining a scientific sound SOPRANO evaluation technique, conducting different phases of user involvement and implementation of a still ongoing (March 2010) large-scale evaluation together have accounted for at least one third of the SOPRANO’s budget.

The SOPRANO requirements analysis was conducted in an iterative step-by-step manner with strong user involvement as well as technical feedback in most iterations (see Figure 9).

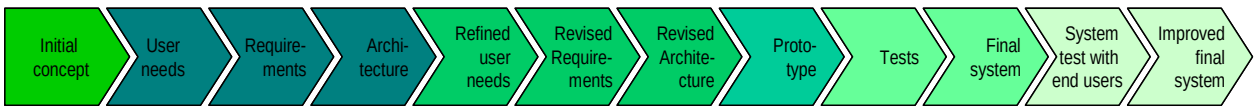

Figure 9: SOPRANO specification process

As the Description of Work in SOPRANO was based on well-founded research, some of the initial conceptual work with regard to the requirement analysis and system design was influenced by it. Based on the Description of Work and further literature analysis, about 80 core user needs were collected in a short XML-like format. The user needs focused on situations of concern and situations of assistance highly focused on older people living in their own homes. This collection of user needs served as input to the requirements analysis, which included the first interaction with potential end-user. Within focus groups in four EU countries discussions were held with elderly people, formal and informal carers. Within these discussions relevant needs and situations were identified. The user needs served as initial guideline in those discussions.

Based on these discussions a set of key challenges to independence and initial ideas for possible technological solutions where derived. Ideas for improved services were developed in response to each challenge to independence, taking into account not only features of technology seen as desirable by focus group participants but also their fears or rejection of other features of technology and of action by service providers seen as intrusive or unnecessary. These key challenges (over 22) acted as the basis for further scenario development. Example key challenges are: Feeling safe and secure, Staying in contact, Being socially active, and Receiving support in case of forgetfulness.

In a next step, 11 scenarios (in SOPRANO also called use cases) were developed that effectively helped to capture the functional requirements via describing interactions between users and the system. Each of these scenarios covered one or more key challenge.

In a final step of the first iteration, a list of ca 100 requirements capturing the essence of these scenarios were extracted. These requirements and scenarios then served as basis to further develop the SOPRANO architecture and the different SOPRANO components. Use cases and requirements were then further explored in subsequent iterations with heavy end user involvement.

In the first cycle of the so-called multilevel prototyping, multimedia mock-ups and theatre presentations were prepared in order to help users get an impression of the technologies prior to prototyping. This phase of user interaction aimed at making decisions on conceptual design, i.e. what kind of functions are to be developed within the SOPRANO system and what the interaction sequences and modalities should look like. Feedback from users was used to refine use cases and revise/detail user requirements. Based on that, the system architecture was revised and SOPRANO component prototypes were further developed.

The second cycle of multilevel prototyping focused on testing usability of the different SOPRANO components and resulted in concrete design refinements of certain aspects of the SOPRANO components. Based on this, the final system was developed that will again be tested in the third cycle of multilevel prototyping with potential end users in a further stage of user interaction. Results from this interaction phase will be used to further improve the system as a whole at the very end of the project (October 2010).

The SOPRANO project has focused mainly on requirements of end-users, namely elderly people, formal and informal requirements. Therefore most of the requirements are captured in the category of functional requirements. These requirements describe functionalities of certain application and services from a user perspective and have been categorized into the groups Reminders, Alerts and messages, User interfaces, Locators, Home management and security, System administration, AP safety and living habits, and General functional requirements.

To capture the refinement according to input from the first and second user cycles, the requirements were collected in a tabular format, e.g. with one column for the original requirement as derived from the first version of the UC, and an additional column for refinements that have to be applied to the original requirement to be compliant with user input from the first and second user cycles.

A far shorter list of non-functional requirements (about 14) mostly captures security concerns and emphasizes SOPRANO's focus on the home of the assisted person. This list has been extended during the process of the project mostly driven by input from technicians but also user input.

In general, due to the strong user involvement an overly formal approach to requirements and use case specification (e.g. UML-diagrams or enterprise architecture) has been avoided. Exchange and specification formats for UC and requirements were mostly based on MS Word and Excel documents.

4 universAAL Requirements Analysis

4.1 The Method and General Planning

As explained in Section 2, the two principles of the universAAL engineering work on system analysis and design are the consolidation of the results of the input projects and the usage of the ARCADE framework. Obviously, each of these two principles influences both the method and the planning of the tasks significantly.

For requirements, consolidation is understood as the **collection** of all requirements from the input projects, **mapping** them to each other, and finally **merging** them into one **harmonized** and **prioritized** list of requirements. However, universAAL has to also complement this process with a gap analysis, especially because of the innovative aspects of the project work, such as the creation of a developer depot and the uStore. Combining this with the deadlines from the Description of Work, we decided to close the consolidation phase at the first deadline at month three and perform the gap analysis based on that before the second deadline at month 12.

The outputs defined by ARCADE have also been divided accordingly. ARCADE defines two models as the main results of the requirements analysis phase, namely the Requirement Model (as a textual description of requirements on a target system in a table) and the Target System Interface Model (as a supplementary specification to the requirement model focusing on the interfaces of the target system with its environment). As most of the input projects provide just a tabular view to the requirements, similar to the ARCADE requirement model, we decided to perform the consolidation based on the input tables of requirements until month three and add the target system interface model based on the results of both the consolidation phase and the gap analysis phase until month 12.

In summary, the tasks related to the first deadline (subject of this version of the deliverable) were identified as collecting, mapping, merging, harmonizing, and prioritizing the requirements inherited from the input projects in form of a table serving as the ARCADE requirement model. We performed these tasks in two internal phases, namely the collection phase and the finalization phase. The following sections provide more insights into the work done in these two phases accordingly.

4.2 Collection of Requirements from Input Projects

The projects referenced as the universAAL input projects consist of Amigo, GENESYS, MPOWER, OASIS, PERSONA, and SOPRANO. Since representatives of these projects are also participating in the universAAL project, the universAAL consortium can naturally handle those requirements more efficiently and without any IPR conflicts. Yet, in the course of the gap analysis, other projects may be taken into consideration, as well.

However, after gathering the statistics from the input projects, it was immediately clear that even having this focus on the above set of projects, the identified tasks of collecting, mapping, merging, harmonizing, and prioritizing the requirements cannot be done mechanically in a simple sequential way. Finally, the conclusion was to bring important aspects of the mapping and harmonization tasks into the collection phase in the sense that all the experts from the input projects that are contributing to the universAAL set of requirements have to agree on a target form of representing the requirements so that each of them does a mapping and harmonization of “its own” set of requirements in terms of the target form of representation (cf. Figure 10).

The basic form was known to be a table, as this is the way ARCADE defines the requirement model. Hence, it was immediately clear that the agreed and common way of representing requirements in universAAL had to be related to the structure of this table. The first consequence was to agree on the set of the columns in this table, which we called the requirement collection template. In the

identification of the columns in this template, the further needs for prioritization of requirements as well as formality and traceability of requirements were taken into consideration. Hence, by describing the input requirements based on this set of columns, they were “automatically” getting harmonized at a certain level. The specific features of this template are described in details in Subsection 4.2.2.

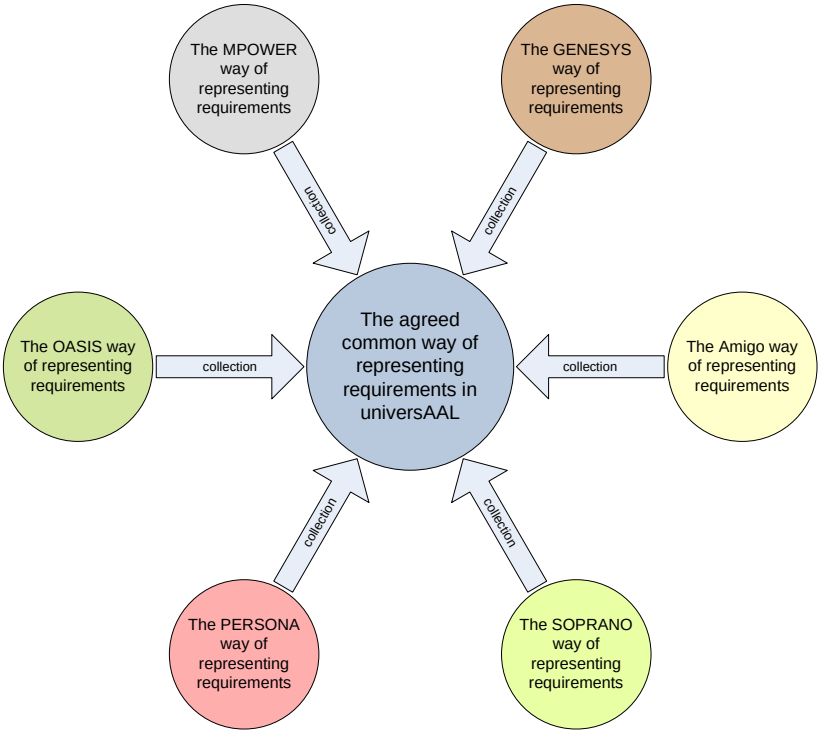

Figure 10: Collecting requirements from input projects with mapping and harmonization

Additionally, the usage of the ARCADE framework provided us readily with means to further structure requirements coming from input projects, namely with a hierarchical list of categories. This list, with its hierarchical numbering scheme that reflects the hierarchy between the categories, was the ideal means for grouping requirements into sets of related requirements. As suggested by ARCADE, each group of requirements falling into a certain category reuses the numbering scheme of that category simply by adding a new hierarchy level. These numbers can then be used as the unique identifiers for the requirements. By mapping the input requirements to concrete categories we could simplify the merging step significantly. This list of categories is described in detail in Subsection 4.2.1.

Obviously, the collection task until here was split between small groups of partners, with each group collecting and extracting requirements of one input projects into the common format. This resulted in the collection of approx. 750 requirements.

The collection phase was then closed by merging the six tables into one single table. This was achieved by splitting up work between several partners along categories. The main goal was to provide a reduced collection of universAAL requirements that, nevertheless, captures all initial requirements. Additionally, resulting requirements should achieve a homogeneous look at style and, as new universAAL requirement, a certain independence of concrete features and components used in the input projects. To achieve this, a series of best practices were used in the consolidation process:

1) Copy requirements of one category of one input project to the universAAL collection sheet. The following points refer to modifications of these copied requirements.
2) Identify requirements that are without much sense, delete them and mark the corresponding requirement in the input project sheet red.

3) Identify requirements that are obviously in the wrong category, delete them and mark the corresponding requirement in the input project sheet yellow (later on, this can be easily merge into the right category).
4) Add the original requirement ID to the column called “reference” (see Section 4.2.2) as reference to where this requirements stems from.
5) Reformulate all requirements
    a. so that they do not reference system components,
    b. so that generic references contain some concrete examples (e.g. “devices, e.g. stove, heater”),
    c. so that overly specific requirements are more generic without losing information,
    d. so that the elderly user is always named assisted person (AP),
    e. so that references to the project names in terms like “Amigo user” or “GENESYS system” are omitted and the requirements are kept as general as possible without losing information.
6) Add requirements from the same category of other input projects by
    a. checking whether it is already captured in existing requirements,
    b. checking whether it can be fit by reformulating an existing requirement, or
    c. adding it as a new universAAL requirement.
7) In case of 6a or 6b add the requirement ID to the reference column (see Section 4.2.2). This indicates the corresponding universAAL requirement captures several requirements.
8) Modify the other columns according to the above suggestions as well as by preserving rating (see Section 4.2.2) with highest importance and changing the universAAL status (see Section 4.2.2) to “proposed”.

With these best practices in mind, it was possible to merge the initial list of around 750 requirements into an initial set of about 400 universAAL requirements. In Section 4.3, we describe how this list was reduced further. In subsequent versions of the deliverable, this list can be further minimized, e.g. by consolidating across different categories.

4.2.1 Hierarchical classification of requirements

As explained above, categories introduced by the ARCADE requirements viewpoint have been used to structure requirements and to ease the consolidation process. Capturing requirements according to the ARCADE categories will facilitate the further integration and exploitation of requirements in the other viewpoints and models of ARCADE.

The following table presents the categories that have been used and short definition of those categories.

Table 3:Adopted version of ARCADE categories of requirements

ID	Category name	Description
R1	**Viewpoint-related requirements**	Requirements specific to a certain ARCADE viewpoint.
R1.1	Component view	Requirements related to the decomposition of a system into its components.
R1.2	Distribution view	Requirements related to the physical and/or logical distribution of software and hardware components.
R1.3	Realization view	Requirements describing constraints that need to be considered during implementation and deployment.
R2	**Concern-related**	Expectations of the stakeholders of a system from the

	requirements	system.
R2.1	Application functionality concerns	Concerns related to the functionality expected Most of the user requirements belong to this category.
R2.2	Quality-related concerns	Concerns related to the quality mostly known as non-functional requirements
R2.2.1	*Communication*	Requirements specific to communication between components.
R2.2.2	*Concurrency*	Specific requirements related to concurrency of software modules and services.
R2.2.3	*Configurability*	Requirements related to the flexibility in the configuration of a system as a whole or internal configuration of its parts.
R2.2.4	*Distribution*	Distribution concerns that have broader impact than can be captured in Distribution view.
R2.2.5	*Fault handling*	Requirements defining how to deal with system or software faults, especially during runtime.
R2.2.6	*Flexibility*	Requirements that demand certain flexibility of the system as a whole or specific functional parts.
R2.2.7	*Heterogeneity*	Requirements related to dealing with possible heterogeneity of components participating in a system.
R2.2.8	*Interoperability*	Requirements indicating the need for interoperability between independent components.
R2.2.9	*Maintainability*	Specific concerns that need to be taken into account when considering Maintenance of the system.
R2.2.10	*Naming*	Requirements that describe specific guidelines for naming system parts and software.
R2.2.11	*Performance*	Requirements on system performance.
R2.2.12	*Privacy-protection / -awareness*	Requirements related to the protection of privacy, especially when private data is going to be shared outside the boundaries of a trusted domain.
R2.2.13	*Quality of Service*	Related to “Quality-related concerns” but with specific focus on service quality.
R2.2.14	*Reliability*	Adherence to specifications: the ability of a system or component to perform its required functions under stated conditions for a specified period of time.
R2.2.15	*Replication*	Things to be considered when replicating data.
R2.2.16	*Safety*	Requirements concerning safety of the user.
R2.2.17	*Security*	Specific requirements on security in a system, e.g. access rights to data and/or services.
R2.2.18	*Scalability*	Requirements regarding the capacity of the system to enlarge, e.g. increase the number of nodes, the amount of data, or number of parallel accesses to a resource.
R2.2.19	*Synchronization*	Specific requirements that indicate special synchronization concerns between components of one system or other systems.
R2.2.20	*User experience & interaction*	Important concerns related to user interaction with the system.
R3	**Assets-related requirements**	Should capture requirements regarding things that exist and are expected to be used in the design and development of a system
R3.1	Standards-related requirements	Requirements on compliance to standards as the most common asset.
R4	**Reference-architecture requirements**	Should capture requirements in strong relation to the architectures outside view.
R4.1	Environment	Things that indicate constraints indirectly relevant...
R4.1.1	*Socio-political requirements*	..., such as the socio-political requirements...
R4.1.2	*Environment system*	... and requirements embraced by external systems.
R4.1.2.1	Binding legacy	Usage of components already existing in the

	components	environment
R4.1.2.2	Binding ultra-thin nodes	Usage of nodes with limited processing power, memory capacity, networking capabilities, etc.
R4.2	Target system	Specific requirements, when the universAAL system is seen as an information integration system, i.e. a target system that integrates information coming from input systems.
R4.2.1	*Target system configurations*	This includes specific configurations...
R4.2.2	*Target system interfaces*	..., Interfaces, and
R4.2.3	*Functionality & data in target system components*	... Functionalities.

4.2.2 Requirements specification template

The column-structure of the collection table was the main utility for harmonizing the description of the requirements imported from the input projects. Table 5 describes the purpose behind these columns.

Table 4: Excerpt from the template provided for the collection phase

ID	**Concise Description**	**Customer**	Customer (dis)satisfaction if		**Measure to check solutions against req.**	**Rationale**	**References**	**Status**
			req. met	**req. not met**				
R1	**Viewpoint-related requirements**							
R1.1	Component view							
...	...							

Table 5: Description of the columns of the requirements table

Column name in the template		**Description**
ID		The unique identifier by continuing the hierarchical scheme of numbering the categories
Concise Description		The essence of each requirement: in most case, the complete requirement description has been added here.
Customer		The stakeholder who benefits if this requirement is considered. To enhance similar look and feel a list of 5 possible customers is provided to choose from: Developer, Assisted Person (AP), Sys-Admin, Relatives, and External service providers. Of course, some requirements have more than one customer.
	satisfaction if req. met	The two "Customer (dis)satisfaction" columns rates the

	dissatisfaction if req. Not met	degree of (dis)satisfaction of the related customer if the req. at hand is (not) met, by specifying a number from 1 (hardly affected) to 5 (extremely affected) -> in both cases, the larger the number, the more important the req.
Measure to check solutions against req.		A notion of how we can measure if the related universAAL result fulfils the req. at hand. This enables testing the system for compliance to the requirement later on. In the case of the universAAL requirements this must be filled later on, since most input projects did not define those measures.
Rationale		can be added to further clarify the requirement
References		A reference to the document where the requirement was taken form to enhance traceability.
Status		One of "proposed", "approved", "implemented", "verified", or "deleted".

As seen above, some of these features enhance traceability and formality of requirements. Most importantly, they capture information that is useful in rating the requirement according to its importance for the customer or the status of its implementation. This information is used in the later stage of prioritization, described in the next section.

4.3 Finalizing the First Stage

The first consolidated list of requirements after the collection phase still suffered from: (1) missing data in the different columns because of the differences between the templates used in the input projects and the template used in universAAL, and (2) the different levels of abstraction and quality in the formulation of the requirements. These issues had made the task of "merging similar requirements into one unique requirement" almost impossible. Especially the first deficit prevented the work group from producing a prioritized list of the requirements.

Therefore, a first attempt aimed at complementing the data according to a division of work between different partners. At the end, however, none of the partners was satisfied with the results because of the low level of interaction between the partners each representing a special field of expertise. During the prioritization process, for instance, the work group realized that at least three people from different partners should be involved in the estimation of the priority *for each requirement*. Moreover, the team identified the need for more criteria and a better tool for scoring.

As a result, the work group decided to form a short-term task force with the mandate to meet physically for several days and fill all the gaps based on intensive discussion of the different viewpoints. To make sure that work packages can later benefit from the results directly in their work, one distinct partner was made responsible per each of the work packages 1 to 4.

One of the important initial works by the task force was to agree on the methodology for prioritizing the requirements. In the beginning, different criteria from diverse viewpoints, such as project management, methodological evidence, technical innovation, realization, and impact, were gathered. In a second step, these groups of criteria were prioritized, according to which the impact-related criteria and those related to the methodological evidence have gotten higher scores. Then, the group tried to reach a common understanding of each of the selected criteria. All these steps were taken based on collective discussions in meetings. In order to be pragmatic when assigning priorities to

requirements, the task force decided to use the Planning Poker method of scrum meetings[9] as a tool: each individual estimated a total priority for each requirement as sum of his / her estimations at the level of each concrete criterion[10]. The two extreme estimations had a very short period of time to reason about their judgements and then a new round of voting was done until there were no more than two different values, where the final decision could be made based on simple majorities.

The following statistics summarize the results of the work by this task force:

Total number of requirements		294
Number of requirements relevant for	WP1	169
	WP2	170
	WP3	58
	WP4	194

The statistics in the different categories are:

R1.1 Component view	*9*
R1.3 Realization & deployment view	18
R2.1 Application functionality concerns	125
R2.2 Quality-related concerns	107
R3.1 Standards-related requirements	4
R4.1 Environment	9
R4.2 Target system	22

The complete list of the requirements has been attached to this report as an appendix. In order to adapt the width of the table to the physical limitations of the appendix, only the columns essential for understanding the requirements have been included.

[9] See, for instance, http://www.scrumbasics.com/planning-poker/. Planing poker is a pragmatic method used when some people have to rate a long list of items very quickly and find a consensus. It is a way of bringing some method into a subjective rating process.

[10] At this stage, the previous work on customer (dis)satisfaction (see Table 5) was very helpful in the category of impact-related criteria.

5 Conclusion and Further Work

The major goal in the first phase of the requirements analysis in universAAL was to benefit from the scientific and engineering work already done in the input projects and consolidate the set of inherited requirements as a shortcut to a set of requirements that have already found a broad common consent in the existing communities.

The remarkable experience we made, however, was that there is no log of software engineering techniques on efficient consolidation processes in the scientific and technical literature. For this reason, we had to follow our general expertise and intuition in the identification of the tasks and "try and error" in doing them. In retrospect, we estimate the process illustrated by Figure 10 as well as the approach taken by the task force reported in Section 4.3 as very reasonable and insightful that can be useful in documenting experiences with consolidation processes in software engineering.

The path toward the universAAL set of reference requirements is already discussed in Section 2.3. The immediate tasks that has to be done will be the further refinement of the requirements on a per work package basis with the necessary topical focus and performing the gap analysis. Furthermore, appropriate forms of sharing the reference requirement with the AAL R&D community must be found that fits to the general idea within universAAL with regard to development support consisting of the developer depot, developer tools, the reference architecture and the related guidelines.

References

[1] ARTEMIS. *ARTEMIS Final Report on Reference Designs and Architectures – Constraints and Requirements.* ARTEMIS Strategic Research Agenda, 2006. http://www.artemis-sra.eu.

[2] S. Bonfiglio. *Questionnaire on Independent Living Applications (ILA)*. OASIS Technical Report, 2008

[3] R. Bræk and J. Floch. *ICT Convergence: Modeling Issues, in System Analysis and Modeling.* 4th International SDL and MSC Workshop, SAM 2004, Ottawa, Canada, June 1-4, 2004, Revised Selected Papers, D. Amyot and A.W. Williams, Editors. 2005, Springer-Verlag GmbH. p. 237-256.

[4] K.V. Isacker, M. Stamatov, and M. Goranova-Valkova. *Questionnaire on Autonomous Mobility and Smart Workplaces applications (AMSW)*. OASIS Technical Report, 2008

[5] OASIS Deliverable D1.2.1. *Use Cases and application scenarios for independent living applications*. Technical Report, 2008

[6] OASIS IR2.1.2. *Independent Living Applications: Preliminary set of user requirements*. Technical report, 2008

[7] E. Stav, S. Walderhaug, and U. Johansen. *ARCADE – An Open Architectural Description Framework*. Developed by SINTEF ICT, Trondheim, Norway, 2010

Appendix A. Detailed List of Requirements as of M3

ID	Name	Description	Rationale	Rating
R1.1	**Component view**			
R1.1.1	Compose-ability	Prior properties and services of the system or a subsystem must not be invalidated by the integration of new subsystems.	Compose-ability facilitates the construction of a system out of pre-validated subsystems. Without compose-ability, properties and services need to be validated again after the integration of subsystems.	5
R1.1.2	Linking Interface Specification	Interaction between subsystems is restricted to the exchange of messages, at the linking interface (LIF) of a component. For those messages an operational specification (i.e., specification of syntactical and temporal properties) and a meta-level specification (i.e., the meaning of the information) have to exist.	Cognitive complexity is reduced, as a system designer can abstract over the inner structure of a subsystem. Only the interface specification of a subsystem is relevant.	5
R1.1.3	Minimalism	At the LIF only objects and functions should be visible, that are required for the interaction with other subsystems.	Internal objects and functions of a subsystem that are visible at the LIF increase the cognitive complexity.	5
R1.1.4	LIF library	Formal descriptions of each subsystem's interface behaviour (i.e., in the functional and time domain) should be collected in a library, where tools can use the formal meta-data to support the integration of subsystems.	Within a library and with tool support it is easier to find available components and subsystems, which raises the degree of component and subsystem reuse.	3
R1.1.5	Simulation-in-the-loop	Simulated environments (e.g., other subsystems, sensors, actuators) should make it possible to test a subsystem before the integration to the system.	Developers can test a subsystem without having access to the (cost-intensive) intended environment, which reduces development risk.	4
R1.1.6	Core management services	A set of management services (i.e. user management, access management, role management and token management) should be included to manage different aspects of the access control solution.	Access to the system should be configured to be adapted to different users and usage profiles	4.5
R1.1.7	Interface to core management service	Interface to core management services (i.e. user management, access management, role management and token management) should be defined for the developer	Developers can integrate access control in their components at their convenience	4

R1.1.8	Public Key infrastructure	A **public key infrastructure** (PKI) is needed to be able to rely on public key cryptography and to issue digital certificates	Public key cryptography is essential to be able to provide end-to-end message security in Web Services-based environments. The use of digital certificates can simplify the access control for elderly and demented (as they do not need to remember usernames and passwords).	2
R1.1.9	Context management	Existence of deliberation mechanisms in order to make decision about which services best suit user needs and profiles	One of central points in AAL (and in AmI in general) is the reaction to user needs and preferences in changing contexts	5
R1.3	**Realization and deployment view**			
R1.3.01	Cross-domain Reusability	IP (intellectual property) modules (e.g., data encryption module) shall be usable for different application domains (e.g., for mobile phones).	Development cost of IP modules can be amortized across different application domains (cg. economy of scale).	4
R1.3.02	Early Prototyping	The architecture shall support prototypes early in the design process for analysis of acceptance of new products.	Design decisions can be validated early in the design process and fundamental errors can be corrected before they become expensive in later design phases.	4
R1.3.03	Independent Certification of the Platform	It should be possible to certify the platform, subsystems and applications independently.	A once certified platform can be reused for different applications. Subsystems can be reused more cost-efficiently when they are certified individually. Additionally, different levels of criticality are possible (i.e., each subsystem can be certified to an individual criticality level).	4.5
R1.3.04	Design for Testability	Testability shall be supported by the architecture (e.g., design, testing, system-integration testing, manufacturing testing and assembly testing).	When test requirements are already considered by the design decisions, the effort of testing can be reduced.	5
R1.3.05	Opportunistic Product Development	Development of products based on available technology rather than on the user's requirements shall be supported (e.g., bottom-up development).	Many products are developed when new technology is available not because of new user requirements (e.g., mobile phones with cameras or MP3 players).	5
R1.3.06	Test Reuse	Reusing of tests for subsystems shall be supported by the architecture.	Test development time and cost can be reduced if a test can be used for multiple subsystems.	4
R1.3.07	Verification Reuse	Verification of subsystems should be supported by reused verification patterns and verification environments.	It is easier to reuse IP modules when verification patterns can be reused.	2

R1.3.08	Formal Methods Qualification	Formal Methods Qualification should be supported by the architecture to apply formal methods.	With formal methods the correctness of specifications, interfaces or properties of subsystems can be formally proved and ambiguous statements can be avoided.	2
R1.3.09	Model-Checking Support	To formally verify properties of the architecture model-checking techniques shall be supported.	With model-checking techniques all possible execution paths of a model can be analyzed rather than only one execution path, like it is done by a normal simulation process.	2
R1.3.10	Theorem-Proofing Support	Theorem-proofing techniques shall be supported by the architecture to formally verify properties of the architecture.	In contrast to model-checking methods, with theorem-proofing techniques deductive correctness proofs of properties can be derived.	2
R1.3.11	Correctness-by-Construction	Probably correct design methods shall be supported by the architecture with which a specification is transformed step by step into a correct design.	Verification of a system or subsystems is facilitated if correct design methods have been used.	2
R1.3.12	On-line operation	The system should have the possibility to connect to the internet connection	Internet connection is needed for communication with external parties or remote management of the system	5
R1.3.13	Off-line operation	For running the system internet connection should not be required all the time.	The unavailability of internet connection should not make the system inoperative	5
R1.3.14	Security standards	Security standards should be used for the implementation and integration of information security aspects	Successful implementation of these standards is to a certain degree dependent on the application platform chosen for development, i.e. Sun Java System Application Server.	5
R1.3.15	Usage training	Initial introduction/demo of the system should be provided.	Systems like this need an initial training of the users	5
R1.3.16	Delivery time	It should not take long time between the need is identified till the equipment or product is installed.	Once the user has taken the decision to install a system like this, time of delivery is an important factor for user acceptance	4
R1.3.17	Cost accounting	Existence of mechanisms that simplify fair and traceable cost accounting policies.	Service providers can offer services (e.g., streaming of movies), which can be paid according to the amount of usage.	2.5
R1.3.18	Design-for-Verifiability	Design-for-Verifiability shall be enforced by the architecture.	In an architecture that considers requirements for verifications, effort for verification can be reduced substantially.	2.5
R2.1.1	**Daily routines, medications, exercise**	**Description**	**Rationale**	**Rating**

R2.1.1.01	Regular activity report	The data collected about activities of the assisted person must be summarised into a report on regular basis.	This reduces the amount of data that has to be sent.	2
R2.1.1.02	Not active enough	The system should detect when AP is not active enough and send a notification about this to responsible parties	The user shall be informed about whether his activities correspond with the recommended levels of physical activity.	2.5
R2.1.1.03	Exercise: recommendation, planning, review	The system should support the AP's engagement in exercises by recommending exercises and showing them to the AP, providing support for planning exercises, review of exercises by some professional. Exercises include physical and mental kinds	Increasing physical and mental exercise is important goal of assistive system	3
R2.1.1.04	Exercise: multi-modality	System should ensure that exercises are visually shown and can be heard	To enhance understanding of instructions	3
R2.1.1.05	Exercise: feedback	System should collect feedback from user after the exercises	As part of a remote or digital physiotherapist	2
R2.1.1.06	Exercise: automatic evaluation	System automatically evaluate correctness of exercise done by user	feature that could enhance impact of exercise	2
R2.1.1.07	Access to Recipes	System should provide access to recipes, e.g. via the internet	easy way to support healthy eating	2
R2.1.1.08	Healthy eating	the system should support healthy eating habits, e.g. by reminding to have regular meals, providing menu suggestions	Healthy eating habits as central part of assistance	4
R2.1.1.09	Medication reminder	Reminder to take medication should be provided.	central AAL topic	5
R2.1.1.10	Medical Health Record	Medical Health Record including medication history should be available on demand	for managing/exchanging health and patient information	4
R2.1.1.11	Medication plan	Individual medication plans should be possible. The system should support in creation of these plans and these plans should be accessible to services and the AP.	central AAL topic	5
R2.1.1.12	Medication alarm	The system should be able to inform a specified carer in case medication has not been taken in time by the AP.	standard escalation procedure	3.5
R2.1.1.13	Medication dispenser integration	If a pill dispenser is connected the system must ensure that no additional reminder is fired when the pill dispenser has already been opened	most commercial available products lack integration possibility	2.5

R2.1.1.14	sleeping problems	The system should collect information about sleeping patterns and informs the AP in case of irregular sleeping habits	Sleeping problems like Apnea are a major diseases among the elderly	3
R2.1.1.15	Intercom	the person should be able to know who is outside the door when the bell is ringing	was mentioned several times in user discussions as main security concern by people with mobility disabilities	3
R2.1.1.16	Shopping list	System should support creation of electronic shopping list, enable ecommerce shopping and provide relatives with access to the shopping list	Support for shopping list can be part of assisting in good eating habits	3.5
R2.1.2	**Assistance outside/ planning of and assistance while on trip/holiday**	**Descr./Rationale**	**Descr./Rationale**	**Rating**
R2.1.2.01	Presence simulation	System should simulate presence of a human being in the home when AP is on holiday	System should simulate presence of a human being in the home when AP is on holiday	3.5
R2.1.2.02	Trip planning	The system should be able to plan a trip	The system should be able to plan a trip	2
R2.1.2.03	Trip planning while on trip	The system supports trip planning while on-trip	The system supports trip planning while on-trip	1
R2.1.2.04	Trip's destination information	The system shows destination information	The system shows destination information	1
R2.1.2.05	Route info	The system gets additional route information	The system gets additional route information	1
R2.1.2.06	Trip info	The system shows short and long trip's info	The system shows short and long trip's info	1
R2.1.2.07	Touristic info	The system shows touristic info of a POI/event/trip in a city	The system shows touristic info of a POI/event/trip in a city	1
R2.1.2.08	Trip evaluation	The system supports getting/giving evaluation on a POI/event/trip	The system supports getting/giving evaluation on a POI/event/trip	1
R2.1.2.09	Time tables	The system is able to get the timetables of the public transportation	The system is able to get the timetables of the public transportation	2.5
R2.1.2.10	Real-time traffic info	The system gets real-time traffic information	The system gets real-time traffic information	2.5
R2.1.2.11	Multimodal trip planning	The system is able to organise multimodal trip planning	The system is able to organise multimodal trip planning	1
R2.1.2.12	Route guide	The system supports elderly friendly route guidance	The system supports elderly friendly route guidance	2.5
R2.1.2.13	Parking info	The system shows parking availability	The system shows parking availability	2.5
R2.1.2.14	Bookmark trip data	The system can bookmark POIs (point of interest) and routes	The system can bookmark POIs (point of interest) and routes	1
R2.1.2.15	Dynamic map updates	The system updates the maps dynamically	The system updates the maps dynamically	1

R2.1.2.16	Trip re-planning on-the-fly	The system notifies and provides the user with the ability to re-plan in case of a problem in the pre-planned route	The system notifies and provides the user with the ability to re-plan in case of a problem in the pre-planned route	1
R2.1.2.17	DRT	The system provides on demand DRT (demand responsive transport) services	The system provides on demand DRT (demand responsive transport) services	2.5
R2.1.2.18	Special DRT	The system provides on-demand special car rental services	The system provides on-demand special car rental services	2.5
R2.1.2.19	Critical situations outside	The system supports detection and assistance in (subset of) critical conditions while outside, e.g. automatic notification of ambulance when health problems are detected by wearable sensors, or help in situations where the AP has fallen.	The system supports detection and assistance in (subset of) critical conditions while outside, e.g. automatic notification of ambulance when health problems are detected by wearable sensors, or help in situations where the AP has fallen.	4
R2.1.2.20	Road accidents outside	The system supports emergency call for road accidents	The system supports emergency call for road accidents	1
R2.1.2.21	Company connection	The system facilitates the connection of the user with his/her company	The system facilitates the connection of the user with his/her company	1
R2.1.2.22	Workplace	The system guides the user in the everyday working conditions	The system guides the user in the everyday working conditions	1
R2.1.3	**Critical situations and emergencies**	**Description**	**Rationale**	**Rating**
R2.1.3.01	warnings and reminders when leaving	When the AP leaves the home, the system should warn the AP about running devices and forgotten objects, open windows, devices that put home-safety at risk, i.e wearable sensors, key, turned on and forgotten oven/stove etc.	Ensures security and safety when leaving the home. A major problem in the domain consider Dementia and minor forgetfulness	5
R2.1.3.02	Indoor security	The system should be able to detect critical situations like smoke sensor is triggered and inform relevant parties such as fire department should be notified.	Security is a major concern in the AAL domain considering problems like increased forgetfulness etc.	5
R2.1.3.03	Critical health status	The system should be able to detect critical changes of the AP's health status and living habits like breath distortion, abnormal blood pressure, sleeping problems, irregular meals etc. and send a notification to a responsible person	Each service should collect information from order services in order to built a global system	5

R2.1.3.04	Health status notifications	Notification shall be send to health and care professionals only if data indicate health problems, otherwise an automatic answer may be created that the parameter are okay.	Health professionals must not be overloaded with information and umber of interactions shall be minimized, however elderly might need feedback and motivation in the meantime.	3
R2.1.3.05	Cancel button	A cancelation button must be provided to cancel alarms in case of false positives	For hearing impaired and/or situations where the system cannot communicate with the user a cancel button is necessary	4
R2.1.3.06	Emergency button	Emergency button system should be integrated to universAAL	To start the emergency process if the system doesn't detect the risk situation	5
R2.1.3.07	Raising alarm	The system should be able to raise an alarm call to a responsible party/external application, if the required infrastructure is available.	Sending an alarm is the natural response to critical situations	2
R2.1.3.08	Data transmission in case of emergencies	The system should be able to transmit AP-related and emergency-related information to a responsible party/external applications in case of emergencies	Data provides useful information for people dealing with this emergency	4
R2.1.3.09	Fall	If fall of the user is detected the system should send an alarm to a responsible person.	Central problem in AAL domain	5
R2.1.3.10	Feedback to user	In case of a fall detection the system should inform the AP that the fall is properly reported and the help is on its way.	Feedback to user is important in critical situations	5
R2.1.3.11	Pervasive fall detection	System should ensure that a fall can be detected in every room.	pervasive detection capabilities are important	5
R2.1.3.12	Prevent fall at night	System should provide facilities to **prevent** a fall during the night (e.g. by lighting the way through the home and avoiding sudden light changes)	prevention of specific problem in cases of dementia	5
R2.1.3.13	Fall outside	System should detect fall outside, at every location where the AP currently is	System should detect fall outside, at every location where the AP currently is	4
R2.1.4	**Calendar, appointments, contacts, user profiles and shopping lists**	**Description**	**Rationale**	**Rating**
R2.1.4.01	Calendar	The system should be able to maintain a calendar for different types of appointments and information, e.g. appointments/comments/memos/activities.	Can act as corner stone for many AAL-related services	5
R2.1.4.02	Calendar API	the calendar service should be accessible by a clear defined and shared interface to developers	If considered as platform services this is necessity	5

R2.1.4.03	Calendar notifications	It should be possible to send notifications about activities and appointments scheduled in the calendar to users	standard functionality Of calendar systems	5
R2.1.4.04	Notify about no confirmation	System should send a message to a previously defined contact if reminder in AP's individual plan system is not responded to or the AP is not following the agenda, e.g. by using user location	special issue in AAL domain to ensure response to critical situations	4
R2.1.4.05	Calendar recommendations	The calendar should be able to propose a list of possible social, mental and physical activities/exercises to the assisted person based on doctors recommendations, user profiles and current scheduled activities.	This enables the user to select the activity of interest.	4
R2.1.4.06	Conflicts between appointments	The system should recognize and assist in resolving of conflicts between calendar appointments also when scheduling appointments for, for example, exercise.	standard functionality Of calendar systems	3
R2.1.4.07	Route to meeting side	The system should provide an estimation of how long it would take to arrive at the appointment on time and should suggest a route to get to the appointment location.	Was asked for several times in user discussions	2
R2.1.4.08	Appointments with other users	System should provide a way for scheduling appointment with other users, e.g. general practitioner, that are able to do so.	standard functionality Of calendar systems	3.5
R2.1.4.09	Contacts over TV	System should provide access to list of contacts over TV screen, but we can also consider other devices	TV is main communication device for the elderly	3
R2.1.4.10	Contact management	The system should provide a contact management service	Can act as corner stone for many AAL-related services	5
R2.1.4.11	Contact notification	System should be able to notify/send a message to people from contact list.	standard functionality of contact management systems	5
R2.1.4.12	Availability of users	The availability of contacts must be definable in the user profile and taken into consideration when sending out messages.	standard functionality of calendar systems	3
R2.1.5	**Situation and sensor detection and situation awareness**	**Description**	**Rationale**	**Rating**

R2.1.5.01	Device sensor	The system should be able to detect status of relevant devices/sensor in the home, e.g. doors, windows, stove etc.,	basis for more complex recognition	5
R2.1.5.02	Device sensor API	Sensor, situation, activity, behaviour and device information should be accessible by developers by a shared and clearly defined interface	As part of the platform services this is a necessity	5
R2.1.5.03	Emergency detection	the system should be able to detect critical situations and emergency based on sensor information and vital signs	to indentify critical health states	5
R2.1.5.04	Sleeping detection	The system should be able to identify whether the AP is asleep.	basis for some use cases	3.5
R2.1.5.05	Leaving bed detection	System should detect when someone is leaving the bed at night	basis for more complex recognition	3.5
R2.1.5.06	Vital signs	The system is able to monitor vital signs	to indentify critical health states	5
R2.1.5.07	Activity and behaviour recognition	System should incorporate and support activity and behaviour recognition based on sensor information	important for identifying situations of concern	4
R2.1.5.08	Emotional state detection	The system should measure the emotional state of the AP, e.g. stress, mood	Important for reacting to depression and other issues	3
R2.1.5.09	Vital signs outside	The system provides health monitoring services outside of the home	to indentify critical health states	3
R2.1.5.10	Bed and chair usage	The system should be able to detect if the AP is lying on the bed or sitting on the chair or sofa.	basis for more complex recognition	2
R2.1.5.11	User location inside	System should be able to store and provide locations coordinates (GPS) and/or symbolic location of the users inside the home	basis for more complex recognition	5
R2.1.5.12	User location outside	System should be able to store and provide locations coordinates (GPS) and/or symbolic location of the users outside the home	basis for more complex recognition	4
R2.1.5.13	Location of objects	The system should be able to locate objects in the house.	basis for more complex recognition	3.5
R2.1.5.14	history location information	Available history of resident location should be stored and provided to applications	basis for more complex recognition	4.5
R2.1.5.15	Identify users	The system should be able to distinguish the AP among the other people present in the house.	needed for personalization	4.5
R2.1.5.16	Leaving detection	The system should be able to detect whether the AP is leaving or entering the home from sensor information	basis for more complex recognition	3.5

R2.1.5.17	Movement detection	The system should be able to detect information about human movement, e.g. speed, location of movement, anywhere in the house	basis for more complex recognition	3
R2.1.6	**User interface**	**Description**	**Rationale**	**Rating**
R2.1.6.1	Personalization of communication	facilitate natural communication in a personalized and context-aware way	the meaning of natural communication depends on user capabilities and preferences and may change depending on the situation the user is in	5
R2.1.6.2	Dialog privacy and language	The system should support multi-language and privacy of messages	to guarantee the consistency between the content and its presentation (e.g. using the same language as the content for the menu entries or using more private channels when people are present that shouldn't see the content; Along with dialog descriptions, applications must provide the UI framework with content-related info relevant for the presentation, such as the language and privacy level of the content	4
R2.1.6.3	Control of devices	User should have the ability to control connected devices, e.g. stove/door/heater through system user interfaces.	Import for the user to stay in control and also to use old devices in new situations	4
R2.1.6.4	Internet access	System should provide easy access to the Internet and access to websites and portals for information exchange, e.g. on healthy eating/living, photo sharing and gaming, according to individual preferences of AP.	Current browsers are not elderly friendly	3
R2.1.6.5	Interface for administrators	The system should provide differentiated access and administration functionality to the different user roles	To create a system that relies minimally on professional system administrators by providing easier interfaces for professional carers, relatives or assisted person	5
R2.1.6.6	Configuration support for relatives	The system should enable AP/relatives to establish the configuration settings	relatives with little or no technical knowledge should be able to change configuration settings, this would enhance acceptance of the platform	3
R2.1.6.7	Download services easy	The system should enable AP/relatives to manage/install/download the services	relatives with little or no technical knowledge should be able to change configuration settings, this would enhance acceptance of the platform	5
R2.1.6.8	Tele consultation	The system shall allow for tele-consulting with medical practitioners, e.g. physiotherapist shall be able to remotely watch exercises	To allow for consultation both in emergency situations and normal ones, without having to travel to doctors office	2.5
R2.1.7	**User experience**	**Description**	**Rationale**	**Rating**

R2.1.7.01	Activate difficulty levels	The system shall allow the users to activate different functionalities and difficulty levels of services.	Patient's physical and intellectual capabilities shall be stimulated from an educational perspective. For some patients some functionality might not be stimulating because they are assuming that the user has not enough knowledge, not being the case.	4
R2.1.7.02	TV on in standby	The system should be able to switch on the TV even when the TV is not in standby mode	Specific problem when TV is main communication device	1.5
R2.1.7.03	Pop-ups on TV	The system should display pop-up messages (warning, error, informative, motivating) on the TV	The system should be able to emit text messages on TV so that the running TV program is not interrupted	2.5
R2.1.7.04	Self-evaluation	The system should support self-evaluation and questionnaires management	self-evaluation is an easy way to get feedback for studies and developers	3
R2.1.7.05	Interfering notifications	The notifications and information to the user shouldn't interfere or bother the user,	The notifications and information to the user shouldn't interfere or bother the user, e.g. when several notifications are send to the user at the same time the system should try to aggregate them	4
R2.1.7.06	No interference with life: sensors	(Wearable) Sensors must not interfere in the common behaviour of the user	User must feel comfortable when using the system	3.5
R2.1.7.07	Daily question	The system should have an service, i.e. Avatar, that asks the AP every day how he feels and if he needs any suggestions for spending the day.	To gather information and introduce avatar more often	1
R2.1.7.08	Minimum login	Minimum login of user in the system	for users that access via non standard pc	3.5
R2.1.7.09	Look and feel	users should experience an integrated/common look and feel that makes easy to interact	Analysis of the realization of the PERSONA use case in its internal report 3.1.2; the SET concept introduced by www.emobility.eu.org	5
R2.1.7.10	Look and feel of messages	The different type of messages, e.g. warnings, notifications, alarms shall have different colours or categorized by category icons	It shall be easy for the sender to choose the right messages (reminder, question, alert, comment, etc) and for the receiver to recognise the message	4.5
R2.1.7.11	Look and feel of voice message	Introduce voice messages by some common jingle or sound	The system should ensure that in case of voice message the message should not appear out of nowhere	4.5
R2.1.7.12	Detect missing wearable sensors	System shall detect that a wearable device is not being worn	The system must known when a wearable device is not worn in order to issue the proper reminder	1
R2.1.8	**Communication between system and user**	**Description**	**Rationale**	**Rating**
R2.1.8.01	Transmission notification	User shall have the possibility to get a notification that receiver has actually got a specific message.	People can sometimes forget to click on the "send button"	2

R2.1.8.02	Transmission transparency	when a message is send on behalf by the system of the user the system should inform the user about this	Important to enhance acceptance of the system and to provide transparency of the system	3.5
R2.1.8.03	Intelligent notification hierarchy	System shall apply a decision/notification hierarchy from assisted person, informal carer, carer, general practitioner, to specialist especially in case of emergencies	Hierarchy allows adequate resource allocation to event or tasks to be performed and ensures user rights described in legal requirements	3
R2.1.8.04	Contact selection based on situation	If the systems sends out a message in an emergency the contact selected should depend on the current situation of the user and predefined set of emergency contacts.	In order to get the best response in each situation	3
R2.1.8.05	Communication in every room	System should enable communication with the system in every room	Must be ensured if we rely on the system in cases of emergency	5
R2.1.8.06	Recorded messages from carers	The system should be able to be extended to receive recorded messages from carers or other people and deliver them to the AP.	To enhance communication over the system and to adapt to more personal communication style	2
R2.1.8.07	Stored messages for externals	System shall provide standardized message to be used by health and care professional	Health professionals are under time pressure and standardised messages reduce time for repeating health and care related tasks or feedback. Standardised messages improve data mining, traceability, search capabilities	1
R2.1.8.08	Reminder about incoming messages	The recipient shall be notified that an incoming communication exist and the recipient shall be reminded to answer the incoming messages	The recipients shall not have to turn on the system to check if there is an incoming message. Also AP can forgot that message arrived.	3.5
R2.1.8.09	Reducing false-positives through feedback	The system must ask the user for feedback in case of risk or emergency situation to reduce false positives. If no response is given in some time span an notification is send to an external contact.	In order to avoid false positive	4
R2.1.8.10	Communication outside	System should be able to communicate with the AP even if the AP is outside of the home	Central corner stone for providing support outside of the home	5
R2.1.8.11	Past messages	The system should enable the review of past messages send to users	important to more complex recognition tasks and maintenance	3
R2.1.8.12	Transmission control	the user should have the possibility manually activate if information is send to some external user	Important to enhance acceptance of the system and to provide privacy	4
R2.1.8.13	Rejection of communication	Acceptance/rejection of the audio communication from the resident	Important to enhance acceptance of the system and to provide privacy	3.5
R2.1.9	**Multimodal interaction**	**Description**	**Rationale**	**Rating**

R2.1.9.01	Text-voice alternative	Text messages shall have a voice alternative	Visually impaired users could have difficulties receiving information from the screen of the PMD. The system shall provide alternative voice messages	4.5
R2.1.9.02	Multimodal interaction	The system must support multimodal interaction	The system must support several interaction ways for an action that means graphical, sound, voice recognition and text should be consider. The user can choose the better interaction mode in each moment. The system must be able to communicate with the user by different ways of channels (voice, graphical, text, etc)	5
R2.1.9.03	Voice inside, text message outside	System should identify whether someone is at home and reminds at home via pre-set modality and outside via a text message and acoustic signal	Was requested in user discussions	3.5
R2.1.9.04	Voip	The system should be able to initiate a phone call or VOIP to a specified person.	Phone as main communication device integrated with the platform	4
R2.1.9.05	Email	The system should be able to send e-mails to a specified person.	easy to achieve communication alternative	5
R2.1.9.06	SMS	The system should be able to send AP-related messages via SMS.	easy to achieve communication alternative	5
R2.1.9.07	Video conference	System should be able to establish videoconference, via VOIP	Was requested in user discussions	4
R2.1.9.08	Speech	The system should have the functionality of informing the AP via speech/audio and accept spoken commands	Especially for people that have problems with visual information	5
R2.1.9.09	Avatar	The system should use an Avatar displayed on TV to increase the acceptance and friendliness of the communication with the AP.		3
R2.1.9.10	Hands-free phone	system should provide hands-free telephone in every room with sufficient quality	Avatars enhance acceptance of the system	3.5

R2.1.9.11	Fusion/fission of modalities	provide for fusion of modalities on the input side and fission of modalities on the output side	entities responsible for interpreting user input coming from input channels monitored by them may need to combine input coming from different channels based on some synchronization mechanism; e.g., when a user says "turn off the lamp over there" while pointing with a finger to the relevant lamp, only a component that listens to events provided by both microphones and gesture recognition components can locate the lamp and provide a meaningful command to the system.	3
R2.1.9.12	In-dialog device/modality migration	support lossless migration of presentation to different presenters depending on the changes in the context; or change the layout-related instructions if the changes in context do not necessitate the change of the presenter but the previously selected presenter must only adapt the layout	the UI framework will be able to seamlessly support dynamic adaptation and, for example, implement "follow-me" scenarios (i.e., the user changes the location and immediately the I/O channels available at the new location are used for continuing the dialog that was not finished yet) or react in a privacy-aware way (e.g., transfer the dialog to more private channels when someone enters the same location as the interacting user and there is a conflict with the privacy-level of the presented content)	3
R2.2.01	**Communication**	**descriptions**	**rationale**	**rating**
R2.2.01.01	Control Loop	At network level control loops of applications shall be supported. Applications involving control loops, where sensing elements and actuators reside on different nodes, require predictable communication delay and jitter. Fixed timing of control loops (sensing, processing, acting),	The control loop is used for diagnosis of health status of components	2.5
R2.2.01.02	Delay/Disruption-tolerant Networking	Communication services that tolerate delays/disruption shall be provided by the architecture also by using back-up systems. Delays and disruption cannot be completely eliminated (e.g., wireless networks).	In case of temporary failure of the communication media the system should be able to use other communication media for communicating with other nodes of AAL spaces	3.5

R2.2.01.03	Event-triggered Message Transfer	The transfer of messages based on the occurrence of an event with guaranteed bandwidth and latency shall be supported by the architecture.	In many applications messages only need to be sent when an event occurs, which allows optimized resource utilization. To improve quality-of-service guarantees of latency and bandwidth are needed	5
R2.2.01.04	Communication Resource Guarantees	For the messages of a given subsystem guarantees of the lower bound on the communication bandwidth, upper bounds on the latency and jitter shall be determinable.	Resource utilization of subsystems that share the same communication infrastructure has to be bounded, especially in the worst-case scenario, in order to ensure the reliability of subsystems.	3
R2.2.01.05	Shared Communication Infrastructure	Sharing of the physical communication infrastructure among multiple subsystems shall be supported by the architecture.	Fewer hardware components, communication busses, cables and connectors are required to interconnect distinct subsystems, which is more cost-effective and increases the dependability of the system.	5
R2.2.01.06	Streaming Media	Streaming of data (sound during voice call, images, videoconference etc) shall be supported by the architecture. The architecture should at least provide a way to setup separate transmission channels.	Many applications require the streaming of data (e.g., surveillance system with cameras), which can be consumed while it is delivered.	4.5
R2.2.01.07	Follow Established Protocol Standards	Well established protocol standards shall be supported by the architecture (e.g., Ethernet, USB, CAN, Bluetooth,...)	Established protocol standards enable interoperability with existing systems.	5
R2.2.01.08	Integrity of Network Environment	The network shall recognize network environment and network changes, i.e. when the AP moves among Wi-Fi hotspots, the system should provide mechanisms for a seamless migration among the hotspots	At link establishment and during execution, authentication has to be assured and used for personalization. Modifications of network addresses or links have to be detected.	3.5
R2.2.01.09	Quality of Service in Communication	A Quality of Service for the communication of the systems should be available, the system should support: dynamic bandwidth allocation controlled actively by the application level or inferred upon the communication of the application, real-time message transfer(i.e. for critical services, such as emergency services), data-streaming	Different service of the systems may need a guarantee level of the communication (i.e. critical services)	2.5
R2.2.01.10	Communication might be initiated and cancel by everyone	All participants might need to initiate and cancel contact	The system shall be configured to make all participants able to initiate and stop the communication	4.5

R2.2.01.11	Simultaneous support for axiomatized and lightweight ontologies	Simultaneous deployment of both formal axiomatized ontologies with rich semantics and light-weight ontologies, as appropriate for defining the content of the messages between nodes	Formal ontologies can be exploited in complex reasoning tasks about situation data, light-weight ontologies can be exploited in automatic interface generation and to facilitate shared understanding	3
R2.2.01.12	Seamless connectivity	support seamless connectivity in the distributed AmI environment underlying AAL systems, like, for example, nodes that find each other and connect in a dynamic way and can use the connections to communicate in an appropriate way	Seamless connectivity is given if nodes participating in an open distributed system find each other and connect in a dynamic way and can use the connections to communicate in an appropriate way. Seamless connectivity is especially appropriate for AmI systems because they provide a boundary for such automatism per se due to being bound to a certain locality.	5
R2.2.01.13	Goal based communication	Utilization of functionality (also known as service utilization) takes place in a goal-based way if requests simply express the meaning of what is requested to be achieved and hence both addressing concrete target components and using syntactical artefacts, such as interfaces, can be avoided. That is, goal-based interoperability can be achieved when in communication there is no dependency on technical details of the "how"s, but participants just focus on the "what"s.	support goal-based communication, like requesting a certain goal-state that has to be accomplish, e.g. request that all lights in a certain room have to be turned on.	4
R2.2.01.14	Uniform communication protocol	enforcing the usage of uniform communication protocols by all modules participating in an AAL system facilitates the modular design of the system in which many design principles can be shared both at the platform and at the application layer	AAL system is usually a distributed system which runs on different nodes. All the nodes should be able to communicate with each other	5
R2.2.01.15	Ubiquitous Connectivity	Ubiquitous (Secure) Connectivity: Access to communication services shall be provided by the architecture.	Access to architecture services or applications should not be restricted to dedicated places.	5
R2.2.01.16	Support medical electronic data interchange (EDI)	The system should provide support for the environment of medical electronic data interchange (EDI)	EDI is extremely heterogeneous and complex, and it is changing continuously.	2
R2.2.01.17	Multiple delivery mechanism	The system should be able to re-emit the high-priority messages after a timeout and drop the low-priority ones after a timeout.	The system should support different delivery mechanism depending the type of the message	2

R2.2.02	Concurrency	descriptions	rationale	rate
R2.2.03	**Configurability**			
R2.2.03.1	General Configuration Tool	Architectural tools shall support flexible configuration of basic software modules	For large systems manual configuration can be very time consuming.	5
R2.2.03.2	System-wide Sensor Calibration	System-wide Calibration: It should be possible to calibrate sensor nodes using a system-wide method.	For feasible fault diagnosis, accurate calibration of sensor nodes is vital.	4
R2.2.03.3	Dynamic Reconfiguration	Dynamic Reconfiguration: During execution the configuration of applications should be changeable (e.g., adding or removing software modules, reallocation of hardware resources, modality for messages to the AP) and we should support plug and play like behaviour	The assisted person does not have to reinitialize the system each time the configuration of applications (e.g., after download of new SW modules) changes.	5
R2.2.03.4	System-level Power Management	System-level Power Management: Power and energy management shall be possible at system-level, depending on actual subsystem requirements as well as available sources (e.g., battery capacity).	Power consumption can be reduced when different subsystems are switched to power saving modes. Furthermore, energy can be reserved for dedicated functions (e.g., only emergency call possible when the capacity of mobile phone battery is low).	3.5
R2.2.03.5	Easy configuration of hardware devices	Different hardware devices (sensors, actuators etc) need to be easily configured and have flexible configuration mechanisms	From AP's point of view AP wants to have the ability to configure the system, hardware as software modules.	3
R2.2.03.6	Users' profile and preferences persistence	All configurations, user preferences and profiles (e.g. main device for interaction with the system, main device for receiving messages, personalized avatar, nutritional profiles) can be saved for future reload, updated or deleted. Initial configuration can be defined.	The assisted person does not want every time to configure same things.	5
R2.2.03.7	Remote configuration	Remote configuration should be possible and enabled.	Don't have be at AP's home to make changes in configuration.	4.5
R2.2.03.8	Customizable system reaction behavior	Any system behaviour or data should be configurable by case managers (sequence of actions reacting to fall or forgotten medication, medication prescriptions, appointments, service for informing third parties, the time to wait for an AP reaction, menus/recipe, to whom to call when emergency happens, communication messages etc.)	Sys-admin want to have full control on AP's system.	5
R2.2.03.9	Customizable data sensing	System polls the sensor according to the general configuration, e.g. timing of polling etc.	To get the measures the number of times necessary to keep the system updated	3

R2.2.04	Distribution			
R2.2.04.1	Transparent Distribution	Transparent Distribution: Allocation of software modules to distinct hardware elements according to performance and dependability requirements shall be tool-based and transparent within the architecture.	Developers can abstract from technical details about the platform, while concentrating on the development of the application.	5
R2.2.04.2	Facilitate integration of ultra-low power nodes	facilitate the integration of different nodes, e.g. a general wrapper for low power devices, like sensors & actuators	The system may be able to interact with ultra-low power devices, such as sensors network either for data gathering or enabling domotic scenarios	4
R2.2.05.01	Unreliable Components	Unreliable Components: The architecture must be capable to tolerate the failure of individual devices and interconnects especially for safety critical components	It is impossible to build 100% reliable devices and interconnects. In case a system tolerates the failure of devices and interconnects, the cost of manufacturing, verification and testing can be reduced.	5
R2.2.05.02	Fault Hypothesis	Fault Hypothesis: Assumptions shall be identified that define the type and frequency of faults which the system has to be able to tolerate especially in safety critical scenarios	For a fault-tolerant system it is important that a fully specified fault hypothesis exists. If real-world faults are not covered by the fault hypothesis, even a fault-tolerant system may fail.	5
R2.2.05.03	Error-Containment	Error-Containment: The architecture must support the establishment of error containment regions, where errors can be detected with a defined error-containment coverage.	The definition of appropriate error containment regions and error detection mechanisms help to prevent the propagation of the consequences of an error within one subsystem to other subsystems.	5
R2.2.05.04	Minimum of two Fault-Containment Regions	Minimum of two Fault-Containment Regions: In case the occurrence of arbitrary (byzantine) failures within one fault containment region cannot be eliminated, an error containment region must be built of at least two fault containment regions.	When the error detection mechanism and the faulty service are located in the same fault containment region, the fault can affect both, the service as well as the error detection mechanism. Thus, error containment could not be guaranteed.	5
R2.2.05.05	Consistent membership Service	Consistent membership Service: A membership service shall exist within the architecture, that consistently provides subsystems with the health state of other subsystems.	For applications where multiple subsystems have to cooperate it is important that each of the subsystems is consistently informed about the current health state of the system. Otherwise consistent distributed decisions can be hard to find.	4.5

R2.2.05.06	Generic Fault-tolerance Layer	Generic Fault-tolerance Layer: Fault-tolerance mechanisms of the environment shall be transparently masked to the application by a common API.	In case the fault-tolerance mechanisms are provided by the architecture and transparent to the application, the complexity of applications can be reduced.	3.5
R2.2.05.07	Tolerance of Software Errors	Tolerance of Software Errors: Protection mechanisms within the architecture shall be able to handle software errors.	Often software is provided that is not free of errors. The AP does want to contact a technician or reboot the system each time an error occurs.	5
R2.2.05.08	Bounded Start-up/Restart Time	Bounded Start-up/Restart Time: A known, bounded and minimal start-up time of system components has to be assured by the architecture.	Predictable start-up/restart time is a key issue for fault tolerance and user satisfaction.	1
R2.2.05.09	Fault Classification	Fault Classification: Error-detection mechanisms provided by the architecture have to distinguish between transient and permanent faults, and as part of this to prevent false alarms.	Classification of faults is important for fault tolerance and maintenance, as in case of transient faults recovery mechanisms like component restart can be applied, while permanent faults require explicit repair action.	3
R2.2.05.10	Pre-emptive Resource Allocation	Pre-emptive Resource Allocation: It must be enforced by the architecture, that subsystems cannot dominate shared communication resources.	An architecture that relies on the cooperative behaviour of subsystems cannot tolerate malicious failures (e.g., babbling idiots) of subsystems.	3.5
R2.2.05.11	Worst Case Execution Time Analysis (WCET)	Worst Case Execution Time Analysis: The calculation of the worst case execution time (WCET) of software modules with feasible effort shall be supported by the architecture.	Deterministic behaviour in each layer of the system is a prerequisite to analyze the WCET of a system. Knowledge about the WCET is a key issue for robustness.	1
R2.2.05.12	Manual error reporting	The system should provide a mechanism to the users for reporting problems not identified by the system.	In case of a error not recognized by the system, the user should be able to provided information to the developer	5
R2.2.06	**Flexibility**			
R2.2.06.1	Technology Independence	Technology Independence: Model-driven design of system functionality in a platform-independent way shall be supported by the architecture. In a subsequent step this model is translated to a technology-specific implementation.	The design of an application is separated from the actual implementation. Hence, when the implementation technology is changed, the application design need not be updated.	5
R2.2.06.2	Implementation Independence	Implementation Independence: Only minimal effort shall be required to replace single subsystems with subsystems realized in another (newer) technology.	When new technology becomes available it is easier to update the system.	5

R2.2.06.3	Platform-independent Executable Code	Platform-independent Executable Code: It shall be possible to execute platform-independent code using virtual machines (e.g., Java Virtual Machine). More than one platform is supported.	For developers it is easier to develop applications that can be executed using different platforms.	4
R2.2.06.4	Plug and Play	Plug and Play: The user should have the possibility to add plug-and-play subsystems which are automatically detected and integrated.	The user can add subsystems by him-/herself without the need to have detailed technical knowledge.	5
R2.2.06.5	Support for Deterministic and Static Allocation	Support for Deterministic and Static Allocation: The architecture has to assure a deterministic and static allocation of time, space, and I/O, at the time of compilation, if required by an application.	Multiple applications (e.g., smoke detection with periodic "alive"-signal) require resources at predetermined points in time with guaranteed bandwidth and minimal jitter.	1.5
R2.2.06.6	Support functional redundancy and functionality overlap	functional redundancy and functionality overlap must be supported by the system	following the principles of open distributed systems: The components of an open distributed system may be redundant, competing with some existing components	5
R2.2.07	**Heterogeneity**	**Heterogeneity**		
R2.2.07.1	Heterogeneity	Heterogeneity: Integration of subsystems, realized with different technologies, has to be supported by the architecture.	Each customer should have the choice to select subsystems according to his/her preference, regardless of which technology is used to realize the subsystem.	5
R2.2.07.2	Mixed-Criticality Subsystems	Mixed-Criticality Subsystems: It shall be possible to use subsystems with different levels of criticality within the one system.	If mixed-criticality is supported by the architecture each subsystem can be certified to its specific criticality level. Otherwise, each subsystem has to be certified to the highest level of criticality in the system (e.g., the media system has to be certified with the same level as the fire alarm system).	4
R2.2.07.4	Generic ontology/model backbone	A generic ontology/model backbone is needed for the interconnection of heterogeneous domain ontologies/models	Existing ontologies in the real world are heterogeneous to a high degree and this fact needs now to be paid sufficient attention when targeting real world ontology development.	4
R2.2.08	**Interoperability**	**Interoperability**		
R2.2.08.1	Clear interface for external systems	A clearly defined interface is required for interfacing with external systems in terms of data formats, communication and ciphering protocols	The system must be able to be used by external system easily	3.5
R2.2.08.2	The system should not interfere with existing systems	The system should not interfere with the operation of existing systems like TV, telephone and especially other tele-care systems.	The system may be deployed on environment which other system like exist and both should be function correctly	4

R2.2.09	Maintainability			
R2.2.09.01	Diagnostic Service	Identification of faulty subsystems for maintenance should be supported by the architecture. The diagnostic service need a holistic view on the system, so that correlated failures and anomalies can be detected. It should be possible to receive the diagnostic service results remotely.	In large systems the identification of faulty subsystems during maintenance can be time-consuming. It is important to replace/repair the actual faulty component instead of a correct one. By recording of the system state after each failure, also failures of subsystems can be detected, that happen only in rare conditions.	4
R2.2.09.02	No Probe Effect	There must not be any interference from the diagnostic service on the subsystems that are diagnosed.	A diagnostic service that interferes with the diagnosed subsystems could introduce errors to that subsystems, and hence, also the diagnostic service has to be certified to the level of the diagnosed subsystems.	4
R2.2.09.03	No Modification	Existing software modules should not have to be modified in order to apply the detection mechanisms.	When the source code needs to be modified, new software faults can be introduced. Thus a recertification of software would be necessary. Additionally, developers may decide to only deliver precompiled code to save intellectual property.	3
R2.2.09.04	Systematic Diagnostic Methods	The detection of application-independent failures modes (e.g., communication errors) should be supported by providing systematic diagnostic methods.	No application-specific knowledge is required for the detection of failures.	3
R2.2.09.05	Application-specific Diagnostic Methods	Diagnostic services should be parameterise-able in order to detect application-specific failures.	It is not possible to detect all types of errors only with syntactic diagnostic methods. Diagnostic methods can then be extended with application-specific knowledge to detect these errors (e.g., the temperature in the house cannot jump from 15°C to 35°C within 10 seconds).	3
R2.2.09.06	Debugging	Whenever defined conditions are satisfied (e.g., a breakpoint within a program is reached), execution should be pause automatically, so internal state of the system/subsystem can be viewed and changed.	With sophisticated debugging features bugs can be found more easily.	4.5
R2.2.09.07	Individual Access	The architecture shall provide a testing mode where an individual subsystem can be provided with input messages in order to control it, and output messages can be observed, even in case the subsystem is already integrated.	The subsystem can be tested within its intended environment.	4

R2.2.09.08	Standardized Test-Interfaces	To test components, standardized interfaces shall be provided by the architecture.	With standardized test interfaces test reuse can be enforced, as the effort to adapt test environments can be reduced.	4.5
R2.2.09.09	State Enforcement	It shall be possible to reload the history state of a subsystem (also remotely).	The duration of testing can be significantly reduced when the state can be directly written to the subsystem instead of executing a test sequence in order to obtain the required state. Additionally, the state of the subsystem can be recovered after a transient fault of the subsystem.	3.5
R2.2.09.10	Dynamically Reconfigurable Built-in Self-test (BIST)	Dynamically reconfigurable BIST shall be supported by the architecture.	Intelligent and reconfigurable self-tests allow each subsystem to detect internal faults itself.	2.5
R2.2.09.11	Event Logging	The system should keep a log of important events for traceability purpose.	After problems the origin can be found easier when the events have been logged.	4
R2.2.09.12	Ontology Management Tools	The system should provided tools for maintenance of ontology	Ontology management tools should be developed within the project.	4
R2.2.09.13	Liability	For liability reasons subsystems that violated their specification must be identified by mechanisms that are provided by the architecture.	As subsystems will be produced by different suppliers, liability is a great issue.	4
R2.2.09.14	Update SW and HW	It shall be possible to update software modules and hardware components without the need to remove it from its location. It shall be also possible using external networks.	Updates can be done more frequently and cost-efficiently.	5
R2.2.10	**Naming**			
R2.2.10.1	Separate namespaces	The identification of subsystems shall be unique, but uniform.	The development of distributed subsystems is easier with separate and unique namespaces.	5
R2.2.11	**Performance**			
R2.2.11.1	Dynamically Scalable Performance	For reduction of power consumption and heat dissipation the architecture shall support hardware resources with scalable performance (e.g., by changing clock frequency).	As not all subsystems are operational at any time, performance of these subsystems can be scaled down to a required minimum.	2
R2.2.11.2	Guarantees for Computational Resources	Guarantees for the availability of computational resources should be supported by the architecture.	Only with guaranteed lower bounds for the availability of computational resources bounded execution times of functions can be achieved. A bounded execution time in turn is a key issue for robustness.	2

R2.2.11.3	Constraint-driven Processing	The system must be able to monitor time constrains when executing applications and provide alternatives when time-constrains cannot be achieved, e.g. routes planner is able to provide results of different quality at different times.	Scenarios like fall, require actions within a defined interval of time (e.g., fall detection). When applications are not able to deliver results in time, an alternative action must be triggered (e.g., an alarm is sent).	4.5
R2.2.11.4	Real-Time Applications	The architecture should support real-time applications (e.g. Activity monitoring).	All data from the sensor must be gathered with real-time or near real-time scheduling.	5
R2.2.12	**Privacy-protection / -awareness**			
R2.2.12.1	Confidentiality	It must not be possible for unauthorized persons to access information or system resources.	The assisted person does not want to have its private data be accessible by anyone.	5
R2.2.12.2	Integrity	It must not be possible for unauthorized persons to modify hardware or software (on platform and application level).	For several applications integrity is explicitly required (e.g., transaction of money).	5
R2.2.12.3	Copy-protection of Streaming Media	Streaming media shall be protected against illegal reproduction.	The provider of streaming media does not want the consumer to make a copy of the streaming content, only authorized persons may view the content (cf. Digital Rights Management - DRM).	2
R2.2.12.4	Automatic Confidentiality Check	Decisions on disclosing private data should be as automatic as possible and the user should only be asked if no clear decision can be made.	The user is not bothered with access requests.	5
R2.2.12.5	Logging of Access to Private Data	Users must be able to verify the results of their privacy settings by checking the resulted distribution of private data, missed services due to too restricted preferences, etc.	The AP can see which of his private data has been accessed by whom.	5
R2.2.13	**Quality of Service**			
R2.2.13.1	Low Power Consumption	Devices operating with batteries should have low power consumption.	Limited power consumption to avoid changing the batteries of the sensors too often.	1.5
R2.2.13.2	Adaptation of Video Resolution	System should allow adaptation of video resolution according to application needs.	Resources can be used more efficiently.	2
R2.2.13.3	Graceful Degradation of Service Quality	Maintenance of QoS and graceful degradation of service quality, regarding fluctuations in connection to the environment.	In case of degradation of the connection, services can still be provided, but with reduced quality of service.	3.5
R2.2.13.4	24x7 Mode	System should be able to operate in a 24x7 mode	The assisted person could want to access to the services at any time.	5
R2.2.14	**Reliability**			

R2.2.14.1	Different Levels of Reliability	The architecture shall provide different levels of reliability to all subsystems.	Distinct levels of reliability are required by different applications (e.g., Web surfing vs. money transactions).	3
R2.2.14.2	Handling of Changing Reliability	Fault tolerance mechanisms shall be capable of adapting to changed reliability of subsystems over life time.	During the life time of subsystems the reliability may degrade. This can be covered by appropriate fault tolerance mechanisms (e.g., in a multi-core system after one core becomes defect, other cores can take over the tasks of the defect one).	3
R2.2.14.3	Reliability	The product (devices, services, information…) should be reliable and remain continuously operational for a long period of time.	Users want to rely on the continuous operation of the system.	5
R2.2.15	Replication			
R2.2.15.1	Replication	Replicas and voting mechanisms (e.g., triple-modular redundancy) shall be provided for error detection and error masking.	It cannot be assumed that hardware components do not fail within some defined interval of time. Using replicated hardware components and comparing their outputs, these hardware failures can be detected.	2
R2.2.15.2	Replica Determinism	For replicated components, replica determinism has to be assured (i.e., replicated components are in the same state and produce the same output within a defined interval of time).	If replicated components are replica determinate, voting upon the outputs can be done bit-by-bit, which does not require specific knowledge about the application and voting can be applied systematically.	2
R2.2.16	**Safety**			
R2.2.16.1	Confirmation of Message	In case a user does not confirm a message, the system should be able to take appropriate predefined actions.	Maybe the AP is not able to confirm or has not noticed the message.	2
R2.2.16.2	Contact to Support Centre on Malfunction	System contacts the support centre automatically if it detects a malfunction.	To avoid the system doesn't work.	5
R2.2.17	**Security**			
R2.2.17.01	Data protection	System shall enable secure data exchange between any customer registered in the system	The customer does not want to have its private data being accessed by anyone when transmitted	5
R2.2.17.02	Non-reputability	Mechanisms for irrefutable proof of who initiated an action within a system, have to be provided by the architecture.	Non-reputability is the basis for accountability and liability.	4

R2.2.17.03	Data validation	System shall allow data validation at time of data reception to reduce threats represented by malicious content.	The system should be protected against any type of thread concerning security and data protection.	2
R2.2.17.04	Data Freshness Verification	Data freshness may be controlled to prevent changes of replay attacks.	The system should be protected against any type of thread concerning security and data protection.	2
R2.2.17.05	Authentication	The architecture has to be able do determine the real identity of systems or users.	In order to authorize the access to the system, authentication has to be assured.	5
R2.2.17.06	Authorization	Access to the system may only be granted to users and systems that have the appropriate permission.	A service can only be accessed by legitimate users or systems.	5
R2.2.17.07	Data integrity	System should support integrity protection of sensitive personal data while it is stored.	Private data is protected from being accessed by unauthorized persons or systems.	4
R2.2.17.08	Detection of Unauthorized Manipulation	System should be able to detect unauthorized manipulation of data that is being transmitted.	The system should be protected against any type of thread concerning security and data protection	4
R2.2.17.09	Logging of Security Incidents	System should be able to log security incidents, such as failed login attempts or unauthorised access attempts to services in order to discover and trace system abuse.	The system should be protected against any type of thread concerning security and data protection	5
R2.2.17.10	Availability	It has to be assured by the architecture that access of authorized persons or subsystems cannot be invalidated by unauthorized persons or subsystems (i.e., denial-of-service).	Denial-of-service may entail economic damages and significant reduction of user-value.	2.5
R2.2.17.11	Anonymous Entities	Differentiate between identifiable entities (e.g., users, modules) that can be authenticated (necessitates the management of their identities and credentials) and anonymous entities that cannot be authenticated while not excluding the second group blindly	Different data or services may be accessible by anyone without authentication.	4
R2.2.17.12	Node and Module Authorization	Ambient intelligent (AmI) environments formed as open distributed systems must be made trustable by providing for node & module authorization	Nodes and modules are identified individually, so a malicious module running on an authorized node cannot pretend to be another module.	3.5
R2.2.17.13	Partly Trusted Modules	Provide a control mechanism for module-to-module messaging, especially to handle the problem of partly trusted modules.	Modules which are trusted may not provide restricted information to modules which are not trusted.	3

R2.2.17.14	Trusted Zones	Trusted zones must be established for the system components that are processing and storing sensitive data.	A trusted site is typically a controlled area where servers and communication lines are protected from unauthorised external access. These areas should be physically access protected.	4
R2.2.17.15	Encrypted channels	All communication involving sensitive information outside the trusted zones must be integrity and confidentiality protected according to security requirements	Sensitive information is protected against security threats.	4
R2.2.18	**Scalability**			
R2.2.18.1	Parallel Processing	The architecture must not rely on increasing performance of single processing elements in order to achieve the required scalability.	It is easier to achieve scalability by using distributed parallel processing, as performance can be increased by introduction of additional processing elements rather than exchanging the existing processing element by a more efficient one.	3
R2.2.19	**Synchronization**			
R2.2.19.1	Global Time Service	A global time service has to be supported by the architecture that is consistent guarantees bounded precision, bounded accuracy, sufficient fine granularity and a sufficient wide horizon.	To temporally coordinate actions among distributed subsystems a global time service is required. Additionally, validity of real-time data coming from distributed subsystems can be determined.	5
R2.2.19.2	Real-time Data Fusion	The architecture has to support the timely and time-dependent combination of data-streams that originate from different subsystems.	Different applications require real-time data fusion (e.g., sensor fusion of control systems, lip synchronization in multimedia systems).	5
R2.2.19.3	Temporal Order	Each subsystem must perceive the occurrence of events (e.g., reception of messages) in the same order as other subsystems of the system do.	Temporal order is a prerequisite for replica determinism as it may have an influence on the internal state of subsystems.	4
R2.2.19.4	Automatic Data Synchronization	The system supports automatic data synchronization	The user can perform changes on one device and review them from another.	4.5
R2.2.19.5	On-the-fly Synchronization	The system supports on-the-fly synchronization	It facilitates migration of current application status while freely moving from each device to another.	3
R2.2.5	**Fault handling**	**descriptions**	**rationale**	**rate**
R2.2.6	**Flexibility**	**description**	**rationale**	**rate**
R3	**Assets-related requirements**			
R3.1	**Standards-related**			

	requirements			
R3.1.1	Standards compliance	uAAL architecture and components design and development are based on well established standards	It facilitates the openness of the platform and the interconnection of services and components developed by others.	5
R3.1.2	Multi Vendor Environment	In order to allow multiple vendors to develop subsystems, different standards have to be supported.	Vendors can follow their own well-established standards when developing subsystems.	5
R3.1.3	Imprecise Standards	Even in the case of differently interpreted standards, the system has to remain robust.	There is a high potential for different interpretations of standards.	5
R3.1.4	Conformance Checking of SW Modules	Tools shall be provided that check whether a SW module is conform to given standards.	Tool assisted conformance checks help developers to build SW modules which are 100% conform to given standards.	1
R4	**Reference-architecture requirements**			
R4.1	**Environment**			
R4.1.1	**Socio-political requirements**			
R4.1.1.1	Digital Rights Management	It shall be supported by the architecture that content (e.g., music, video, etc.) can be bound to a license and usage is restricted to the conditions given in that license.	Many digital media providers require DRM.	1
R4.1.1.2	Better assistance	User must feel the system provide a better assistance than not having it	Benefits of having a system like this should overcome the inconveniences in order to be accepted by users	5
R4.1.1.3	No personal care replacement	The system must not be designed as a full replacement of personal cares of relatives, neighbours or friends	Users don't want to stop receiving personal care, they wouldn't use this service if they think so	5
R4.1.1.4	Motivating factor	System shall act as a motivation factor for keeping the elderly active	The user has to feel that using this service is positive for his health. He shall obtain positive messages as he completes the activities. This way, he would get motivated to keep using the service	4
R4.1.1.5	User must feel in control	User should have the control of the system any time and receive info from the system status or action	User must feel assisted by the system,	4
R4.1.2	**Environment system**			
R4.1.2.1	**Binding legacy components**			

R4.1.2.1.1	Intellectual-Protected Software-Modules	It should be possible to integrate precompiled modules.	Suppliers that want to protect their intellectual property may only provide precompiled code.	5
R4.1.2.1.2	Defined Execution Environment	Software modules require a defined execution environment that offers standardized interfaces for software modules.	Standardized interfaces of the execution environment allow to develop software modules independently from each other by different developers.	5
R4.1.2.1.3	Legacy Integration	It has to be supported by the architecture to integrate legacy systems/subsystems/nodes.	A system does not have to be built from the scratch. Old systems can be migrated to a new architecture.	5
R4.1.2.1.4	Inverse Legacy Integration	It has to be possible to integrate new systems/subsystems based on a new architecture into an old environment.	New architectures can be used within old system environments.	1
R4.1.2.2	Binding ultra-thin nodes			
R4.2	Target system			
R4.2.1	**Target system configurations**			
R4.2.1.1	Modularity	Architecture should enforce modularity when defining its components	AAL systems must be developed modular enough to allow the substitution and improvement of all of its parts, no matter if at the platform or at the application level, just by minimal affections to the operation of the overall system	5
R4.2.1.2	Modules container		the modularity of AAL systems necessitates that they are based on a (distributed) "container" system that hosts the modules and manages their life cycle	5
R4.2.1.3		Keep the mandatory core configuration small and allow the system to be configured on a per user basis even at the level of components to install	the bigger the mandatory core of a system, the higher the rigidity of the system configuration and the more complex the system deployment; adaptability not only at the level of behaviour of components but also at the level of the selection of the system building blocks	5
R4.2.1.4	Compose-ability	It must be a matter of configuration to introduce new value that can be achieved through aggregation over existing values	reduce the development costs by reuse based on modular development principles	4
R4.2.2	**Target system interfaces**			

R4.2.2.1	Man-Machine Interface Decoupling	Specific man-machine interfaces (i.e., between the human operator and the machine) shall be decoupled from the general man-machine interface (i.e., between the man-machine subsystem and the rest of the system).	The special man-machine interface can be easily exchanged (e.g., using different languages, look-and-feels).	4
R4.2.3	**Functionality & data in target system components**			
R4.2.3.1	Shared context data	A shared understanding of context data, a unique language for the exchange of such data, and a brokering mechanism are needed in order to reduce interdependencies at the development time	facilitate the extensibility of the framework and the free plug-and-play of both context providers and context consumers	5
R4.2.3.2	Push mechanism for context management	Provide a push (asynchronous subscribe/publish) mechanism for notifying subscribed components about changes in the context as soon as changes interesting for them take place	prompt reactivity to certain changes in context that lead to certain situations of interest belongs to the basic principles of ambient intelligence as the enabling technology for AAL	5
R4.2.3.3	Pull mechanism for context management	provide a pull (synchronous query) mechanism so that components can query the shared knowledge about context at any given time based on a certain query language	context consumers may need to access the shared knowledge about context only in certain situations, maybe not only to retrieve the most up-to-date but also historical data, maybe not targeting just one info item but a combination of them	5
R4.2.3.4	Availability of context data	guarantee the availability of contextual data independently from the abilities of their original providers to keep a history or answer queries	fill the gap caused by context publishers that provide no query interface and / or cannot maintain the whole history of data provided by them, this way guaranteeing the essential support to reasoners that perform statistical analysis and need context stored over time	5
R4.2.3.5	Context reasoning	provide a general framework for reasoning about context (estimating the state of some context elements by combining different known information and applying certain methods of aggregation, statistical analysis, and / or logical deduction)	If for certain needed contextual info there is no provider that can report about it (e.g., no sensor to measure if someone is just relaxing or is already asleep), then, special context providers may be needed as to derive such info using aggregations, statistical analysis, and / or logical rules, to name the most common methods. Such context providers are normally called context reasoners or interpreters. Context reasoning is the major use case for playing both of the provider and the consumer roles simultaneously.	4

R4.2.3.6	Shared profile data in user context	handle shared profile data as part of user context in order to facilitate more powerful reasoning based on both user capabilities / preferences and context data published by context providers	shared profile data, as opposed to application-specific profile data, is an important part of the user context and hence can be used in the course of context reasoning	3
R4.2.3.7	Initialization mechanisms	provide mechanisms to reduce the overhead of initializing profiling data	experience shows that profiling data can get very complex very quickly; initializing this data manually is normally a cumbersome task that could lead to problems with use acceptance; different mechanisms can be applied, e.g., using default data based on stereotyping methods in combination with machine-learning methods or simple incremental collection / correction of profile data so that each portion of profile data is manually collected / corrected the first it becomes necessary	4
R4.2.3.8	Dialog parameterization	The UI framework must instruct a selected component from the presentation layer how to present a dialog by passing an appropriate set of adaptation parameters to it { e.g., voice gender and volume level in case of selecting the voice-based presentation)	the share of the presentation layer in adaptation (content-related to be done by applications, device & modality selection to be done by the UI framework, and "layout"-related to be done be the presentation components)	4
R4.2.3.9	Context awareness: easy access	The UI framework must provide applications with facilities to easily access relevant context and personalization parameters that may affect the process of preparing the content	applications may consider such parameters in the process of preparing the content to be presented to the user. E.g., adaptive applications may choose the content language or prioritize (e.g., through context-aware sorting) or emphasize different portions of info differently	4
R4.2.4	**Target System Devices**			
R4.2.4.1	PC platform for maintenance	PC platform used in maintenance should be supported	Needed for system maintenance	5
R4.2.4.2	PC platform for development	PC platform during development supported	Used during the development phase	5
R4.2.4.3	PC platform for deployment	PC platform to host the runtime environment should supported	Some functionalities may require a device with high technical capabilities	5
R4.2.4.4	TV for user interaction	TV as a communication platform should be supported	TV is the most familiar device for elderly and they won't need to learn to manipulate a new device	5

R4.2.4.5	Mobile devices for user interaction	Smart phones/PDA/tablet should be supported as a communication platform	Depending upon the smart-phone technical capabilities at the time of the project's realization, the functionalities, we will integrate those functionalities that are feasible to install.	5
R4.2.4.6	Mobile devices for deployment	Smart phones/PDA/tablet to host the runtime environment should be supported	Depending upon the smart-phone technical capabilities at the time of the project's realization, the functionalities, we will integrate those functionalities that are feasible to install.	3
R4.2.4.7	Embedded system for deployment	Embedded system (e.g. set top box) to host the runtime environment should be supported	Depending upon its technical capabilities at the time of the project's realization, we will integrate those functionalities that are feasible to install.	4
R4.2.4.8	Embedded system for user interaction	Embedded system (e.g. set top box) should be supported as a communication platform	Depending upon its technical capabilities at the time of the project's realization, we will integrate those functionalities that are feasible to install.	4