
 <p>SEVENTH FRAMEWORK PROGRAMME: PRIORITY 7.1B LARGE SCALE INTEGRATING PROJECT (IP)</p>	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	
<p>Document Type: “Deliverable:” Item Appearing in “List of Deliverables in DoW with delivery date shown in bold “Supplementary Report” As “Deliverable”, but delivery date <i>not</i> shown in bold. These documents are formally internal to the consortium, but can be delivered on request.</p>		Project Deliverable, with independent sub-parts. <i>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.</i>
		Project Deliverable (single document, no sub-parts).
	X	Sub-part of a Project Deliverable.

Document Identification			
Deliverable ID:	D2.2-A	Part title:	Part I – Report on the development work
Release number/date:	1 / 05.04.2011		
Checked and released by:	Sergio Guillén/ITACA		

Key Information from "Description of Work" (from the Contract)	
Deliverable Description	<i>Software that is installed on top of the universAAL execution environment, adding generic AAL service functionality. AAL platform services are software that is installed on top of the universAAL execution environment and Generic Platform services, providing business level AAL service functionality. Ontology artefacts are software and documents that enable developers to use a shared view on the AAL concepts and information models</i>
Dissemination Level	PU=Public
Deliverable Type	P = Prototype
Original due date (month number/date)	Month 11 / 31.Dec.2010 (changed in the first project review)

Authorship & Reviewer Information	
Editor (person/ partner):	Oliver Höftberger / TUV & Zaher Owda /USIEG & Saied Tazari / Fh-IGD
Partners contributing	AIT, CERTH, CNR-ISTI, ENT, Fh-IGD, FZI, IBM, ITACA-UPV, TSB, UPM, TUV / USIEG
Reviewed by (person/ partner)	Salvatore Flavio Pileggi/ ITACA

Release History

Release number	Date issued	Milestone *	eRoom version	Release description /changes made
1	13.01.2011	PCOS proposed	1	Initial version containing the structure an outline of content
1	22.02.2011	Intermediate approved		Only based on software repository and the wiki pages
1	28.03.2011	External proposed	2	All input from subfolders integrated
1	30.03.2011	External reviewed	3	
1	31.03.2011	External revised	4-7	All feedback incorporated
1	04.04.2011	External approved	8	
1	05.04.2011	Released	9	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 an 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 VALENCIA (ITACA, Technical manager) Contact person: Laura Belenguer 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: Casper Dahl Marcussen Email: cma@medcom.dk
PROSYST SOFTWARE GmbH (PROSYST) Contact person: Kai Hackbarth Email: k.hackbarth@prosynt.com	TECHNISCHE UNIVERSITAET 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

Release History.....	2
universAAL Consortium	3
Table of Contents	4
Executive summary	5
1 About this Document.....	6
1.1 Relationship to other sub-parts of this deliverable	6
1.2 Relationship to other versions of this Part.....	6
2 Development goals	7
2.1 Scope of the developments	7
2.2 Related requirements	8
3 Status Report	9
3.1 CM – Context Management	9
3.1.1 Context Bus	9
3.1.2 Jena Conversion.....	10
3.1.3 Context and Profiling Ontology Artefacts.....	11
3.1.4 Context History Entrepôt.....	12
3.1.5 Situation Reasoner	13
3.1.6 AAL Space Manager	14
3.1.7 Platform-information Providers and Rules.....	15
3.2 SIEG – Service Infrastructure Expert Group.....	16
3.2.1 The Service Bus.....	16
3.2.2 AAL Space Orchestrator	17
3.3 UIM – User Interaction Management.....	18
3.3.1 I/O Buses	19
3.3.2 Dialog Manager	20
3.3.3 Different I/O Handlers.....	22
3.4 RIEG – Remote Interoperability Expert Group.....	23

Executive summary

This document is Part I of D2.2-A, which reports about the work done in Task 2.2 “universAAL generic platform services, AAL platform services, and ontology artefacts”. It emphasizes the development goals with regard to the requirements of the current iteration phase. Then, an overview of the steps is provided that were taken towards the alpha releases of the first set of software artefacts. Finally, it reports about the status of the work done based on an extensive list of artefacts for each expert group.

Task 2.2 has taken advantage of the consolidation process in universAAL as it adopts a selected set of existing software components from the input projects which match the best the universAAL reference architecture.

In the standard Section 1, the relationship between this document and the other sub-parts of D2.2 is presented. While Part I presents an overview about the current status and work done in Task 2.2, the other two parts include actual results of recent activity in the task in form of wiki pages and software. Thereafter, this document also provides the appropriate links to the respective wiki pages and SVN repositories. To access those repositories, you might need to create your own account on <http://depot.universaal.org/>.

1 About this Document

This document is the first sub-part of deliverable D2.2. It presents the actual state of the development of generic platform services, AAL platform services, and ontology artefacts. The document focuses on the development goals of the actual iteration of the development process, which includes the scope and the requirements of the current iteration. Another important aspect is the approach (see D2.3-A and its changes described in D2.1-A) used to obtain design decisions, and the selection and adaptation of existing software components from input projects. Finally, the list of software artefacts is presented for each expert group in task 2.2 with info about the status and development plan for each of them.

1.1 Relationship to other sub-parts of this deliverable

Deliverable D2.2 is divided into three sub-parts labelled Part I to Part III. While this part presents an overview of the strategy and work done in task 2.2, the other two parts include actual results of recent activity in the task. The second part consists of the wiki pages which are an important tool to document individual design decisions, the structure, hierarchy and interaction of software components listed in this document. In part III source code of the software components developed so far are contained.

1.2 Relationship to other versions of this Part

This document is part of the first version of deliverable D2.2. It contains a report about the scope of the first iteration of this task and the work done so far. The first iteration concentrates on a basic set of software components upon which other services (mainly from other work packages) are constructed. These software components are currently in an alpha release state.

In the next versions of this part will follow descriptions of changes to the currently existing software components, which elevate those components to the beta release state. Additionally, a second set of software components will be introduced, that extend the functionality of the basic components. This second set of components will be available in the alpha release state.

In later versions further improvement of the basic set of software components as well as the second set of software components will be documented.

2 Development goals

The software development in task 2.2 concentrates on the creation of a reference implementation of platform services and ontology artefacts that will be used by other work packages and application developers within the universAAL community. Thereby architectural design decisions from WP1 - Consolidate a standard reference architecture for open AAL platforms - are considered as basic input for emerging services.

In addition to the specifications from WP1, several software components have already been created in some of the input projects. It is a goal of universAAL to reuse as much existing code as possible in order to save the efforts for new developments. Hence, at first these software components need to be analysed individually and if necessary be modified so that they are compliant with the specifications in WP1 and D2.3.

2.1 Scope of the developments

As explained in Part I of D2.1, the scope of the developments within WP2 has been mapped to the scope of work within seven expert groups that have been formed based on major structural elements of the universAAL reference architecture described in D1.3-B. The expert groups in the scope of Task 2.2 are:

- **Context Management Expert Group (CM):** The CMEG concentrates on the provision of context information that can be used to adapt applications to the actual situation of the user. Context information can, for example, be obtained from sensors that measure real world conditions, but it can also be personalization information that describes preferences of the user. Please refer to <http://depot.universaal.org/wiki/cm:Overview> for more details on the scope of this group.
- **Service Infrastructure Expert Group (SIEG):** It is the purpose of this expert group to define and implement the service infrastructure for the universAAL project. This comprises, e.g., the service model, brokering between service providers and service requesters, querying of available services, handling of composite services, and provision of common services. Please refer to <http://depot.universaal.org/wiki/sieg:Overview> for more details on the scope of this group.
- **User Interaction Management Expert Group (UIM):** This group provides a common I/O infrastructure for multimodal user interaction of independently developed applications. With this I/O infrastructure, it is possible to switch between different mechanisms for user interaction (e.g, display or sound system for output, keyboard or microphones for input) in a context-aware and personalized way without the need that the application is aware of such situational switches. Hence, the system adapts the mode of user interaction depending on the current situation of the user and/or his/her preferences. For more details on the scope of this group, please refer to <http://depot.universaal.org/wiki/uim:Overview>.
- **Remote Interoperability Expert Group (RIEG):** Interactions between an AAL space and the outside world are treated by this group. This includes the communication of humans in an AAL space with other humans or systems outside this AAL space, as well as interactions of the AAL system with humans or systems at other places. For more details on the scope of this group, please refer to <http://depot.universaal.org/wiki/rieg:Overview>.

2.2 Related requirements

In this section a number of main requirements related to artefacts presented in this deliverable are introduced:

- In Context Management, the requirements which are directly fulfilled by these components can be summarized as follows: provide an asynchronous push mechanism for notifying subscribed components about any changes in context as soon as changes occur in which they are interested; provide a synchronous pull mechanism responsible for giving components the ability to query the shared knowledge about context at any given time; define a shared understanding of context data for exchanging them and a brokering mechanism in order to reduce interdependencies at the development time; preserve a unique temporal order between the context events so that all context consumers receive the events in the same order as they occurred; interaction between subsystems must be restricted to the exchange of messages at the linking interface (LIF) of components; a membership service shall exist within the architecture, which consistently provides subsystems with the health state of other subsystems; finally, facilitate the simultaneous deployment of both formal axiomatized ontologies with rich semantics and lightweight ontologies, as appropriate for defining the content of the messages between nodes. For the exhaustive list of the requirements per CM building blocks, please refer to the subpages under <http://depot.universaal.org/wiki/cm:Overview>.
- Service Infrastructure is responsible for facilitating the sharing of functionality between components of AAL systems with the following requirements: specify format of service profiles and service requests; provide a matchmaking mechanism to find appropriate service utilities derived from the service profiles that match a given service request; provide facilities to query existing services without calling them as well as to register as a listener to the availability of services for receiving notifications when services become available / unavailable; provide a scripting language for service composition along with the appropriate interpreter (workflow engine / service orchestrator); the above service-based interoperability solution must support semantic interoperability in order to provide for a real ecosystem of independently developed components. For the exhaustive list of the related requirements, please refer to the <http://depot.universaal.org/wiki/sieg:Overview>.
- UI Management identifies solutions for explicit interaction between the inhabitants of AAL spaces and the respective AAL space. It must ensure that applications can be developed independently from the concrete I/O infrastructure in specific AAL spaces and independently from the concrete interaction capabilities of the human users. Nevertheless, the UI management must be able to provide for context-aware and personalized interaction using the most appropriate modality and device depending on user situation. For the exhaustive list of the related requirements, please refer to the <http://depot.universaal.org/wiki/uim:Overview>.
- Remote Interoperability must facilitate the provision of remote assistance by service providers to the inhabitants of AAL spaces. Therefore, it must enable seamless access by the inhabitants of AAL spaces (the customers of the service providers) to the sites of the providers, and vice versa, enable the remote access of the service personnel to the agreed data and functionality available in the AAL space. Also pure software-based interaction for exchanging data and utilizing services must be supported. All these interactions must be done in a secured way based on agreed permissions. For the exhaustive list of the related requirements, please refer to the <http://depot.universaal.org/wiki/rieg:Overview>.

3 Status Report

In the following subsections, each of the expert groups in the scope of Task 2.2 reports about its status with regard to the procedure described in Section 3 of Part I of D2.1-A¹. Hence, we strongly recommend to first read that section to get familiar with the WP2 development strategy and approach.

Note: Since Task 2.3 is performing certain laboratory tests by installing and configuring all the artefacts from all expert groups while also measuring performance and executing artefact-specific tests, all expert groups in the following subsections share the following two steps in their agendas that we avoid to repeat by introducing them here:

- M16: Complete the documentation of all artefacts based on feedback from Task 2.3 and update the development agenda accordingly
- M18: Apply urgent bug fixing and implement the most urgent/mandatory tasks for stabilizing the releases of M18

3.1 CM – Context Management

This group has finished the procedure defined in Section 3 of Part I of D2.1-A successfully. The group has decided to adopt the software artefacts developed within the PERSONA project after comparing all the candidates in the light of the group design decisions. As a matter of fact, this will also facilitate the integration work with the middleware, because universAAL has decided to reuse the PERSONA middleware, as well. Please refer to <http://depot.universaal.org/wiki/cm:Overview> for the details of the design decisions and the results of the evaluation of the “distance” of the input projects to the design decisions of the group.

Apart from harmonizing all of the imported software artefacts with the guidelines of D2.3-A, this group has also performed some preliminary development work: (1) all artefacts were adapted to the changes posed by the middleware expert group, and (2) the context ontology has been modularized further in order to improve ontology sharing and deployment conditions.

The following are the software artefacts that implement the responsibilities of the context management.

3.1.1 Context Bus

Black-box description	The Context Bus is built on the middleware providing publish-subscribe mechanism for peers of the AAL Space where they can forward Context Events, built with ontological descriptions. Artefacts deliver these events by extending the abstract class ContextPublisher and using its event publishing capabilities; they can register for listening to events based on patterns by extending the abstract class ContextSubscriber and using the provided subscription mechanism.
Wiki link	http://depot.universaal.org/wiki/cm:Context_Bus
SVN link	http://depot.universaal.org/svn/cm/trunk/bus.context/
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/cm/mw.bus.context
Current status/	The bus is in charge of the semantic matchmaking between published events and the

¹ An update of the integration strategy originally specified in D2.3-A and modified later as described in D2.1-A.

functionalities	<p>registered patterns from the listeners. There can be as many instances of the same bus in an AAL space as there are AAL-aware nodes running the universAAL middleware. When a context provider publishes an event to one these bus instances, the event will be broadcast to all instances of the middleware, then each instance of the context bus performs the matchmaking locally and delivers the appropriate event to the matching local subscribers.</p> <p>The artefact contains:</p> <ul style="list-style-type: none"> • The basic model, mainly consisting of events, patterns, publishers, and subscribers, that is used by artefacts interacting with the Context Bus. • Implementation of the actual engine behind the matchmaking of the bus.
<p>Development agenda</p> <p>ITACA: 2PMs</p> <p>Fh-IGD: 0.5 PM</p>	<ul style="list-style-type: none"> • M21: Enhance matchmaking to incorporate permissions • M24: provide support for activating and deactivating context publishers depending on available subscriptions and Situation Reasoner rules (considering “always-on” cases) → enhance the ContextProvider ontology to force context publishers to also provide info about the patterns that apply to their context events

3.1.2 Jena Conversion

Black-box description	<p>This artefact has been imported but may not be required in the end because the performance of the Jena database for RDF data was not satisfactory. However, it serves as a placeholder for design decision about the new RDF database and the associated data conversion (in both directions) among the middleware implementation of RDF+OWL and the target model, no matter which other existing implementation is taken. If the expert group decides to implement its own database model from scratch, the effort needed for the implementation of the new database model (while having the middleware implementation of RDF+OWL in mind) substitutes the development effort for a new conversion.</p> <p>The module for converting the middleware implementation of RDF+OWL to Jena implementation and vice versa. Both are Java implementations but not compatible. The Jena implementation was needed because of using the Jena database for persistent storage of context events. The middleware had avoided using Jena implementation because it is too complete and general-purpose and hence too big with many features not needed at all. The middleware implementation was very compact, targeted towards specific needs, and also runs under Java 1.3 without the need for external libraries.</p>
Wiki link	http://depot.universaal.org/wiki/cm:Jena_Conversion
SVN link	http://depot.universaal.org/svn/cm/trunk/ctxt.jena.converter/
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/cm/ctxt.jena.converter
Current status/ functionalities	<ul style="list-style-type: none"> • The artefact contains two packages, one with the interface of the parser and the other with the activator and the implementation.
Development	<ul style="list-style-type: none"> • M18: Finalize design decisions on the RDF database with SPARQL engine

agenda	(contribution by ENT, Fh-IGD, and UPM)
ITACA: 1.5 PMs	<ul style="list-style-type: none"> M21: perform the resulted development task (either an own DB schema with own implementation for working with the DB, or a new conversion software)

3.1.3 Context and Profiling Ontology Artefacts

Black-box description	Pluggable ontology artefacts to support context-awareness and personalization, corresponding to the Common Model Extension building block in the universAAL reference architecture as introduced in D1.3-B.
Wiki link	Currently, distributed in three pages due to the original bundling in PERSONA: http://depot.universaal.org/wiki/cm:CASF_Ontology http://depot.universaal.org/wiki/cm:Profiling_Ontology http://depot.universaal.org/wiki/cm:CASF_Ontology_Extension
SVN link	Ontologies can be shared between all expert groups; for this reason, they have their own SVN repository: http://depot.universaal.org/svn/ontologies/trunk/
Nexus link	Collection of ontology artefacts available in http://depot.universaal.org/maven-repo/org/universAAL/ontology
Current status/ functionalities	<p>PERSONA had bundled the whole ontologies in only three bundles: (1) The Context Awareness Supporting Framework (CASF) Ontology contained an eclectic set of ontology domains. Domains ranged from the most basic "physical thing" and "location" concepts to other concepts needed for the integration of devices, such as "temperature". (2) The profiling ontology contained all data related to user model. It was based on the CASF ontology and defined multiple user types and accordingly profiles, each with specific additional information such as identification, preferences, impairments and illnesses. (3) An extension to the CASF ontology with simultaneous reference to the profiling ontology that was bundled separately in order to avoid cyclic references between the CASF and profiling ontologies.</p> <p>When importing to universAAL, we started to split the original artefacts to several smaller ontology modules to be able to work on their improvement more effectively while supporting a better management of ontological dependencies (e.g., generally resolving the possible problems with cycles) and facilitating the efficient deployment of ontology modules. The splitting has not finished yet; the following modules have been created so far: audio_video, blind, device.extra, furniture, lighting, location.extra, measurement, medication, physical.world, powersocket, profiling, weather, and window.</p> <ul style="list-style-type: none"> There are two location ontologies: one more exhaustive model in physical.world with support for image recognition, positioning of objects and space configurations and one compacter model in location.extra. There is an ontology supporting multimedia streaming communications in audio_video. There are some concepts related to medication but there is no medicine or

	<p>health ontology.</p> <ul style="list-style-type: none"> • The weather and measurement ontologies are very incomplete. • Some device ontologies lack the ‘Service’ class needed on the service bus.
Development agenda	<ul style="list-style-type: none"> • M15: decide about which location ontology to use in universAAL (design decision by the whole CM EG) • M16: complete the modularization task for all parts of the PERSONA artefacts casf.ont, casf.ont.ext, and prof.ont (ITACA, Fh-IGD) • M17: stabilize the user model (design decision by the whole CM EG) • M18: adapt the existing ontologies to the changes planned by the middleware group for mw.data.representation, improve them (e.g., add the ‘Service’ classes), and complete their documentation (Fh-IGD: 0.75 PM, ITACA 0.75 PM) • M27: continuously add new ontologies as need arises (all partners from WP2, WP3, and WP4; efforts classified as side effect of the actual development work that arose the need)

3.1.4 Context History Entrepôt

Black-box description	<p>A singleton component that takes care of logging every context event that is published on the context bus. It gathers the events by subscribing to the context bus with a Context Subscriber that defines a "pass-all" filter so that every context event received from the bus is forwarded to the component. These events are stored in a relational database as reified RDF statements that can be held in an external (to CHE) database server.</p> <p>An external RDF database allows all components in the network that have an appropriate DB account to access it directly without passing through the CHE. However, this kind of access must be reserved only for platform components, e.g., the Situation Reasoner, apart from CHE itself. All the pluggable components will have to use CHE services on the service bus to access the historical data. There are ongoing discussions about the necessity of allowing this distributed access to DB to other components. In any case always some access to the knowledge on the database would be provided.</p> <p>There is also a mobile version of the CHE that only gives the functionality of storing context events. It is assumed that no component in the mobile device will have to access these stored events. They are only saved to be transferred to the central CHE at home. Therefore it provides no services in the Service Bus, at least in the current version. It just connects to the Context Bus and stores every event in serialized XML form inside a file that will be moved to the main computer when the user gets home.</p>
Wiki link	<p>http://depot.universaal.org/wiki/cm:Context_History_Entrepot_Artefact</p> <p>Mobile: http://depot.universaal.org/wiki/cm:Context_History_Entrepot_Mobile_Artefact</p>
SVN link	<p>http://depot.universaal.org/svn/cm/trunk/ctxt.che/</p> <p>Mobile: http://depot.universaal.org/svn/cm/trunk/ctxt.che.mobile/</p>

<p>Nexus link</p>	<p>http://depot.universaal.org/maven-repo/org/universAAL/cm/ctxt.che Mobile: http://depot.universaal.org/maven-repo/org/universAAL/cm/ctxt.che.mobile</p>
<p>Current status/ functionalities</p>	<ul style="list-style-type: none"> • Automatic storage of context history: The component stores every context event forwarded through the context bus. • Retrieval of context events from/to/between timestamps: It provides methods (services in Service Callee) to retrieve the list of all events form, to or between given timestamps (given in the Service Request). • SPARQL query over service bus functionality: A service (in Service Callee) is provided that takes as input a SPARQL query, formatted as a String, which will be executed in the CHE database. The resulting events are returned to the client, so the query must be properly formatted to ask for context events. • SPARQL query over database functionality: Since the history is stored in a Jena RDF relational database any component with access to this database can use Jena to ask directly for whatever data it wants, using SPARQL. • Automatic removal of context history: Removal of context history is performed periodically to keep the database size under control and prevent it from growing unlimitedly. • Short-term context history storage: The CHE provides interfaces (Service ontology) to allow other components implement their own context history storage for a short period and act as a short-term CHE. • Mobile version synchronization: The CHE is in charge of synchronizing events stored in the mobile version. To do this first the mobile device will have bulked the context history file it uses into the running folder of PERSONA. Then the CHE parses this file and stores each parsed event it contains into the database. <p>Mobile:</p> <ul style="list-style-type: none"> • Automatic storage of context history: The component stores every context event forwarded through the context bus. • Automatic removal of context history: Removal of context history is performed periodically to keep the stored data size under control and prevent it from growing unlimitedly. <p>NOTE: The synchronization capability of the mobile version is not implemented yet. It should consist only of a feature to copy the stored data file in the main home computer when the mobile device gets at home.</p>
<p>Development agenda ITACA: 1.5 PMs</p>	<ul style="list-style-type: none"> • M21: Adapt to the storage system selected / implemented under 3.1.2 • M24: Implement and test the synchronization of the mobile version with the stationary version. Enhance the stationary synchronization with mobile version once that is implemented.

3.1.5 Situation Reasoner

<p>Black-box description</p>	<p>A general-purpose reasoner that uses the database of CHE (and the possibility of ontological reasoning of its storage engine) and builds up new contextual information using the power of the RDF query language SPARQL. It stores “situation queries” persistently and indexes them based on context events that must trigger its evaluation – as they are not meant as a one-time query that are answered</p>
------------------------------	--

	<p>and then forgotten, but they must generate related situational events whenever appropriate, depending on changes in the context.</p> <p>In this conjunction, the CM expert group believes that there is need for a graphically interactive tool to facilitate the definition and editing of rules based on the knowledge about available providers and / or loaded ontologies. This development task is however assumed to be done in Task 3.5.</p>
Wiki link	http://depot.universaal.org/wiki/cm:Situation_Reasoner
SVN link	http://depot.universaal.org/svn/cm/trunk/ctxt.gen.reasoner/
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/cm/ctxt.sr
Current status/ functionalities	<ul style="list-style-type: none"> • Scripted MySQL plug-in that is installed in the CHE database. It allows to monitor and react to the storage of new events that must be taken into account for the situation detection • It was supposed to provide two services on the service bus, one for accepting new situation queries and the other for dropping them, but these services are not realized. • Introduction of new rules currently is only possible by writing SQL insert statements and running them on the console of MySQL. • The source of data that can be used in the declaration of a situation is limited to the shared context. • Only situations that can be derived by a single SPARQL query can be recognized.
Development agenda Fh-IGD: 1.5 PMs ITACA: 1 PMs	<ul style="list-style-type: none"> • M21: Adapt to the storage system selected / implemented under 3.1.2 • M21: provide services on the service bus

3.1.6 AAL Space Manager

Black-box description	<p>This component is supposed to be a synthesis of four different artefacts from two of the input projects:</p> <ul style="list-style-type: none"> • The space.conf artefact of PERSONA that provided services with regard to defined locations and physical things with their shapes and coordinates. It worked in tight connection to the physical world ontology with related inference power and could answer queries about the configuration and state of the current AAL space. • The prof.server artefact of PERSONA that provided specific services for handling requests related to the different user types and their profiles. • The prof.editor artefact of PERSONA as an admin application for viewing and editing profiling data. • The “little world” editor from SOPRANO that was used to describe the initial state of AAL spaces.
--------------------------	---

Wiki link	prof.server: http://depot.universaal.org/wiki/cm:Profiling_Server
SVN link	No software imported yet
Nexus link	ditto
Current status/ functionalities	Still under the discussion how to combine the four above-mentioned artefacts and where to put the boundary between Task 2.2 and Task 3.5
Development agenda Fh-IGD: 1 PM FZI: 3 PMs	<ul style="list-style-type: none"> • M18: finalize the design • M27: finish the implementation

3.1.7 Platform-information Providers and Rules

Black-box description	All of the software artefacts for runtime support produced in tasks 2.1 and 2.2 have certain knowledge about the operation of the system, especially the artefacts of the middleware. Here, the expert group will analyze how this knowledge can be shared as context info for increasing the reliability in system and adding diagnostic features.
Wiki link	There are some discussions related to this topic in the wiki pages of the middleware expert group, however, there is no specific wiki page dedicated to this topic yet. One might see also some relationships between this action point and the wiki page available under http://depot.universaal.org/wiki/cm:Common_Up-lifters_&_Providers_Building_Blocks
SVN link	Possible artefacts will be added to: http://depot.universaal.org/svn/cm/trunk/
Nexus link	Possible artefacts will be deployed to http://depot.universaal.org/maven-repo/org/universAAL/cm/
Current status/ functionalities	There is nothing existing.
Development agenda Fh-IGD: 1 PM USIEG: 5 PMs	<ul style="list-style-type: none"> • M18: Cooperate with the middleware group to analyze how the noticeable issues in the operation of the system found out by the middleware artefacts can be pushed up and caught here in order to produce and publish the appropriate context events. • M24: Provide the context publishing code in accordance with the design decisions in the previous step • M27: Specify diagnostic rules for the Situation Reasoner • M33: Provide appropriate handlers for critical situations

3.2 SIEG – Service Infrastructure Expert Group

This group has finished the procedure defined in Section 3 of Part I of D2.1-A successfully. The group has decided to adopt the service bus implementation from PERSONA but develop a new service orchestrator from scratch. The following two paragraphs provide an overview of the reasons for taking these decisions, but for more details, you can refer to <http://depot.universaal.org/wiki/sieg:Overview>.

The reason for adopting the PERSONA service bus was mainly because it provides a distributed service infrastructure that enables semantic interoperability to an extent that resolves the problem of a-priori agreement between Service Providers and Service Requesters about APIs so that they can formulate Service Profiles and Service Requests using an ontology of their own choice, possibly different ones, and develop their applications independently. The mapping between different ontologies used by Service Providers and Service Requesters can be created later, even by third parties. There were some similar approaches also in SOPRANO and OASIS, but due to usage of technologies with a longer tradition and bigger community (e.g., OWL-S as opposed to Diane) and also due to creating less programming overhead in comparison to other alternatives, the Service bus implementation of Persona was chosen. As a matter of fact, this will also facilitate the integration work with the middleware, because universAAL has decided to reuse the PERSONA middleware, as well.

The reason for developing the Service Orchestrator from scratch is that all of the input projects had used BPEL as the scripting language for service composition and all were reporting about difficulties in real usage. Finally, the group decided to develop a new scripting language that combines the service composition ontology provided by OWL-S with a simple embedding of work with the middleware buses (registering service profiles, sending service requests, publishing events, and subscribing for events) so that powerful scripts can be written as an easy and fast way for realizing AAL applications.

The following are the related software artefacts in the scope of SIEG.

3.2.1 The Service Bus

Black-box description	Extends the middleware by providing for service-based semantic interoperability. It mediates between independently developed provider and requester agents that might be arbitrarily distributed on different nodes within an AAL space. It hides the distribution and the possible heterogeneity of the execution environment of the provider and requester agents. It defines protocols for provider agents how to advertise their service utilities by semantically describing them in service profiles and how to cooperate in the processing of service requests. It also defines protocols for requester agents how to formulate their requests semantically in terms of goals that they want to reach. Finally, it resolves the mediation task by matchmaking between the received service requests and the advertised service profiles and by taking over all the necessary conversions for realizing the targeted end-to-end communication without the need that the two ends know each other. It should be sufficient that they share the same or at least a compatible understating of the related domain.
Wiki link	http://depot.universaal.org/wiki/sieg:Profiles_Requests http://depot.universaal.org/wiki/sieg:MatchMaking http://depot.universaal.org/wiki/sieg:Protocol http://depot.universaal.org/wiki/sieg:Querying
SVN link	http://depot.universaal.org/svn/sieg/trunk/bus.service
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/sieg/mw.bus.service/

<p>Current status/ functionalities</p>	<ul style="list-style-type: none"> • Provides an enhanced implementation of OWL-S in Java for creating service profiles and a mechanism for registering them by provider agents. The registered profiles are indexed appropriately for later use in matchmaking. • Provides a Java interface for creating service requests that are equivalent to an enhanced version of SPARQL particularly developed for service querying and provides a mechanism for issuing such requests and receiving the responses both in a synchronized and in an asynchronous way. The same SPARQL-based interface can be used for querying the registered service profiles (not implemented fully) as well as for making “availability subscriptions” (implemented). • The matchmaking is based on the general reasoning capabilities provided by mw.data.representation but also particularly tailored to matching service requests against registered service profiles. • The whole bus strategy for hiding the distribution is based on a configurable permanent coordinator instance that might pose a single point of failure in AAL spaces.
<p>Development agenda</p> <p>Fh-IGD: 2 PMs</p> <p>CERTH: 2.5 PMs</p>	<ul style="list-style-type: none"> • M18: add support for declaring the required permissions in service profiles and enhance the matchmaking to check granted permissions against the required ones (in coordination with enhancements in mw.bus.model and in cooperation with the security expert group) • M18: enhance the interfaces to allow direct work with OWL and SPARQL as strings in addition to the equivalent Java interfaces • M21: Improve support for non-functional parameters (OWL-S service parameters) and improve the documentation to push their usage • M21: complete the implementations for querying service profiles • M24: Revise bus strategy to avoid single point of failure • M24: enhance the mechanisms and the Java interface for SPARQL-based service requests and queries to handle also the case of “composition by query”

3.2.2 AAL Space Orchestrator

<p>Black-box description</p>	<p>This component is based on the idea of support for Service Composition using an appropriate “scripting” language, such as BPEL and OWL-S, with a workflow engine for executing composite services defined in those languages. However, the AAL Space Orchestrator should work on the basis of a novel scripting language that allows to utilize the whole functionality available in AAL spaces, including interaction with human users and context-aware activation. It should be implemented in a modular way,</p>
------------------------------	---

	<p>consisting at least four artefacts:</p> <ol style="list-style-type: none"> 1. An interpreter for the scripting language² 2. The actual AAL Space Orchestrator that manages a configurable database of the installed scripts and starts the interpreter to run such scripts in a situation-aware way 3. A shell added to the OSGi shell that enables developers to run commands by typing them in the OSGi console, at least for simulating context events (related to Task 3.1?) 4. A graphically interactive tool to facilitate the development of orchestrating scripts based on the knowledge about available functionality and / or loaded ontologies (related to Task 3.5?)
Wiki link	http://depot.universaal.org/wiki/sieg:Composition
SVN link	No existing software is reused
Nexus link	ditto
Current status/ functionalities	Start from scratch!
Development agenda	<ul style="list-style-type: none"> • M27: Scripting language specification (Lead by IBM with 3 PMs, contribution by CERTH, Fh-IGD, and FZI) • M33: Realization of the language interpreter (IBM 2.5 PMs) • M33: Realization of the Orchestrator with context-aware activation (Fh-IGD 2 PMs, design contribution by FZI)

3.3 UIM – User Interaction Management

This group has finished the procedure defined in Section 3 of Part I of D2.1-A successfully.

The group has decided to adopt the software artefacts developed within the PERSONA project after comparing all the candidates in the light of the group design decisions. There was no other input project that provided a whole and consistent concept for user interaction in AAL spaces in terms of a UI framework (each project had covered though certain aspects, such as some support for context-aware and personalized interaction). This was already encountered when the group started to “define itself” (the scope of its work). For this reason, the group tried to consider external related work: portable UI from Open Health Tools and Universal Remote Console (URC – proposed to ISO in 2002 and standardized in 2008: ISO/IEC 24752). The first one showed to be in start-up without concrete results so far; the corresponding working group showed very interested to look at the universAAL solution once it is completed. In case of URC, several sessions were dedicated to discuss the two

² We might call this language ASOR (Aal Space ORchestrator); according to Wikipedia “The asor (Hebrew: עֶשׂוֹר; Hebrew for "ten" 'eśer [ayin shin resh עשר) was a musical instrument "of ten strings" mentioned in the Bible.” Then, it would be a good match, not only because of relationship between orchestration and musical instruments but also because it is going to be specified under the leadership of IBM Research in Haifa, Israel.

solutions in comparison, with the inventor of the URC involved. Finally, UIM reached the understanding that URC is not related to the core business of the group (user interaction in AAL spaces as smart environments) but to separating user interface development from the business logic of service components. However, modern service-oriented architectures, such as universAAL, already provide for this feature per se. Please refer to <http://depot.universaal.org/wiki/uim:Overview> for more details.

In the course of importing the related software artefacts from PERSONA, all artefacts were adapted to the changes posed by the middleware expert group. The group has also scheduled the following overall task that affect several artefacts:

- M27: Look for easier and more efficient options for configuration and setup

The software artefacts imported by UIM are described in the following subsections.

3.3.1 I/O Buses

Black-box description	The input and output buses provide for making applications independent from the concrete I/O infrastructure (set of concrete I/O channels) available in AAL spaces as they might differ in their occurrences considerably. This is achieved by distinguishing between an “application” layer and a presentation layer where the I/O buses sit in between. The components on the imaginary presentation layer can then be called I/O channel managers (or I/O Handlers, as they were called in PERSONA) as opposed to applications that reside on the imaginary application layer. For this purpose, this artefact (1) provides a UI model for the independence between application layer and presentation layer, (2) facilitates the integration of "applications" and I/O Handlers possibly running on different nodes in different execution environments while hiding the possible distribution and heterogeneity of the related HW/SW artefacts, and (3) provides a context-aware and personalized brokering mechanism between the presentation and application layers based on the UI model that effectively contributes to the adaptation of presentation to the user situation, capabilities, and preferences.
Wiki link	http://depot.universaal.org/wiki/uim:IO_Buses
SVN link	http://depot.universaal.org/svn/uim/trunk/bus.io/
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/uim/mw.bus.io
Current status/ functionalities	<p>I/O buses enable communication between applications and I/O handlers. The artefact consists of two buses: the output bus for selecting an appropriate I/O handler for presenting a dialog (prepared by an application) to the user and the input bus for transferring the user input in this dialog (captured by the I/O handler) back to the application. The output bus relies on the Dialog Manager (see the next artefact) for the realization of the situation-aware selection of the right I/O handler that is most suitable to process a certain output event.</p> <ul style="list-style-type: none"> • provide a dialog package for describing dialogs in a modality- and layout-neutral way (inspired by XForms) • handle explicit input by human users and system output addressing human users based on a publish/subscribe mechanism

	<ul style="list-style-type: none"> responsible for finding best components (subscribers) that are able to process certain events restrict their members and their messages to the ones implementing the right interfaces the output bus accepts registration parameters from I/O handlers that describe their capabilities in handling output events; I/O handlers can dynamically update their registration parameters if needed the output bus defines protocols for suspending and resuming dialogs, dynamic adaptation of “rendering” by a previously selected I/O handler during the dialog is running, and transfer of responsibility to another I/O handler (the latter two features as a function of changes in the user situation) the input bus automatically drops subscriptions when input event is delivered
<p>Development agenda</p> <p>ENT: 3PM</p> <p>Fh-IGD: 0.75PM</p>	<ul style="list-style-type: none"> M18: Upgrade the specification in all aspects including the dialog package, the adaptation parameters, and the protocols, especially for a security-enhanced matchmaking, support for redundant dialog managers, possibility for supporting modality fusion and fission on the buses, possible improvement of adaptation mechanisms, and addition of new features like “apply” and “overlay” protocols M21: finalize the provision of adaptation parameters, such as the concrete rules for determining the modality, the necessity to use private I/O channels resp. the freedom to use public I/O channels, etc. M24: finish the implementation of the upgraded specification

3.3.2 Dialog Manager

Black-box description	Dialog Manager is the main (application-independent) component for handling system dialogs. It (1) represents the whole system by providing system menus (a unified view of all services available), possibilities to search for specific services, and by handling context-free user input (user input that cannot be assigned to any running dialog), and (2) assists the I/O buses by (2.1) acting as a representative for the whole framework supporting context-awareness & personalization (i.e., the framework under the responsibility of the CM expert group), and by (2.2) providing user-specific management of dialogs initiated by different applications to protect the user against a mess of parallel dialogs.
Wiki link	http://depot.universaal.org/wiki/uim:Dialog_Manager
SVN link	http://depot.universaal.org/svn/uim/trunk/dm/
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/uim/ui.dm/
Current status/functionalities	<p>Handles system-wide dialogs:</p> <ul style="list-style-type: none"> Provides for navigation through available service classes based on user- and language-specific configuration files Provides means for users to suspend and resume dialogs (see also the

	<p>underlying mechanism in assisting I/O buses)</p> <ul style="list-style-type: none"> • Handles such user input that could not be assigned to any running dialog (the so-called context-free user input) <p>Manages system reactivity (in universAAL, this feature is delegated to the AAL Space Orchestrator):</p> <ul style="list-style-type: none"> • provides a database for associating service calls with situations • subscribes to the context bus for all situations in the above database that are associated with actions • upon occurrence of the above situations, constructs a service request from the associated action data and sends it to the service bus <p>Assists the I/O buses</p> <ul style="list-style-type: none"> • enriches output events by adding the (personal and situational part of) adaptation parameters to them • notifies the output bus to re-do match-making for the selection of the appropriate I/O handler for any running dialog whenever relevant adaptation parameters change • manages parallel dialogs for several users based on priority queues of published dialogs and takes care that only one dialog is presented to the user at each point in time • provides mechanisms for suspending and resuming dialogs
<p>Development agenda</p> <p>ENT: 0.8 PM</p> <p>Fh-IGD: 1.35 PM</p>	<ul style="list-style-type: none"> • M18: Upgrade the specification in all aspects including <ul style="list-style-type: none"> ○ overlaying of one dialog over another one to see how to upgrade current solution especially when dealing with important notifications ○ automatic main menus as an alternative to configuration files by taking into account access rights of different users ○ simulation mode to dialogs while changing adaptation parameters ○ seamless use and interaction by multiple users at once ○ possible additional "standard dialogs" beyond suspending running dialogs & resuming with them ○ possibility for proactively initiating dialogs that fit to a current situation, e.g. "service chaining" ○ different ordering of menus according to frequency of use or situational conditions • M21: Implement a first set of improvements • M27: finish the implementation of the upgraded specification • M33: remove the solution regarding "system reactivity" (see current functionality") as this is now delegated to the Service Orchestrator in the expert group for service infrastructure (wait with this action until the substitution is

	available)
--	------------

3.3.3 Different I/O Handlers

Black-box description	Assuming that the separation between presentation and application layers is achieved by the I/O buses (with the help of the Dialog Manager), I/O handlers and supporting UI services can be added, removed or replaced dynamically and freely; hence, they belong to the Common UIs building block of the reference architecture. This kind of plug-ins will be responsible for capturing input (with or without modality fusion) and presenting output (with or without modality fission) while complying with the protocols of the I/O buses. To capture user input, I/O handlers might decide to connect directly to certain input device types, use intermediate native managers that handle the low-level connections to input devices and are able to provide the data in its raw form or already interpreted in a specific context (like window managers in case of graphical user interfaces), or subscribe to certain context events that they expect to be published by input device controllers. Similarly, to present system output to users, I/O handlers might decide to connect directly to certain output device types, use intermediate native managers that handle the low-level connections to output devices and provide sophisticated API at different levels of abstraction (like window managers in case of graphical user interfaces), or utilize device services on the service bus when they expect such services to be provided by output device controllers.
Wiki link	http://depot.universaal.org/wiki/uim:GUI_Handler others to be added
SVN link	http://depot.universaal.org/svn/uim/trunk/handler.gui/ others to be added
Nexus link	http://depot.universaal.org/maven-repo/org/universAAL/uim/ui.handler.gui
Current status/ functionalities	<p>From PERSONA, at least the GUI I/O-Handler and the Web I/O-Handler are going to be used in universAAL. Currently, only the first one has been imported to the UIM repositories. It renders the dialogs described in terms of the dialog package of the I/O buses graphically using Java Swing as an intermediate native manager and hence has to be installed separately on each node that has both the appropriate I/O capabilities and the appropriate JVM, e.g. on computing devices with connected (or integrated) display, mouse, and keyboard or on “all-in-one” TVs. It manages the complete layout of the screen. Other characteristics are:</p> <ul style="list-style-type: none"> • divides layout in 4 areas: top (for title), bottom (for general purpose buttons), main (for controls form) and right (for submit buttons) • after receiving the data it manages the screen layout • layout properties can be altered via .properties files • 4 types of properties' values are: <ul style="list-style-type: none"> ○ None (if properties file does not exist default values are used)

	<ul style="list-style-type: none"> ○ General values (define layout properties for all services) ○ Form values (define certain form's layout) ○ Group level values (used by the group of controls based on their depth level within the form)
Development agenda	<ul style="list-style-type: none"> • M24: develop a new I/O-Handler for Android smart phones based on the current GUI I/O-Handler (IBM: 1 PM) • M24 & M33: Provide two updates of the GUI I/O-Handler (ENT: 3PM, UPM: 0.2PM) • Provide at least one update of the Web I/O-Handler (not scheduled yet) • M33: develop a new I/O-Handler that works purely gesture-based (Fh-IGD: 3 PMs)

3.4 RIEG – Remote Interoperability Expert Group

Wiki Home: <http://depot.universaal.org/wiki/rieg:Overview>

This group has finished the procedure defined in Section 3 of Part I of D2.1-A partially: the scope of the group work and the related requirements have been specified; the design decisions regarding import and export of functionality between AAL spaces and remote assistance sites has been made; inspection of reusable code is also mostly done; but, the design decisions regarding support for remote admin, and gateway-gateway communication between two AAL spaces are not finalized yet; also, the group is still lacking a clear action plan so that the development plan has remained very roughly; accordingly, the wiki pages must reflect the latest results as the group makes progress with its design decisions and action plans.

The reason for a slower progress in comparison to most of the other expert groups is the fact that there has been little correspondence between what was done in the input projects with what is supposed to be covered in the scope of RIEG as a novel approach. This causes that the group is forced to reach to its own design decisions from scratch.

Some parts of the work in the input project OASIS, such as semi-automatic generation of semantic service descriptions from web service description languages (such as WSDL), seem to be relevant for RIEG but the group still needs to make more concrete design decisions, on one side, and reach a better understanding of the boundaries with tools possibly related to WP3, on the other side. Other OASIS work that was assumed in the beginning to be relevant for RIEG, however, has shown to be more related to the ontology management tasks in the Middleware expert group and / or to the tool support by WP3; examples are the OASIS ontology repository and mechanisms for ontology mapping.

The expert group has identified three major components with the following rough division of work but has not decided yet how to modularize the components in terms of concrete software artefacts in the SVN repositories:

Major Component	Work Item	Description	Partners' intentions already stated

AAL Space Gateway ³ (ASG)	importing subcomponent	When the inhabitants of an AAL space decide to use functionality provided by an external entity from within their AAL space in a seamless way (or the provider is interested that the functionality is provided to the AAL space in a seamless way) then the provider agent must be “proxied” locally. This subcomponent defines the related protocols, provides the necessary interfaces, and plays the proxy role for all such external service agents.	CNR-ISTI and Fh-IGD contribute to the specification.
	exporting subcomponent	When software running on remote assistance sites needs to access functionality from an AAL space (e.g., call services or receive context events) it must register to the service and context buses of the AAL space. This can be done by the same proxy as above by providing the conversion of access methods in the reverse direction. The security mechanisms defined by the Security expert group must also be considered.	AIT and Fh-IGD contribute to the specification. AIT plans also about 1 PM for the implementation.
	ASG-ASG communication	Provides for the needed level of connectivity and exchange of functionality and data between two AAL spaces.	CERTH, CNR-ISTI, and Fh-IGD contribute to the specification. CNR-ISTI plans also about 2 PMs for the implementation.
	Mobile version	User’s near-body mobile AAL space needs also such an ASG functionality that must be activated on a mobile device worn or carried, as soon as the user leaves the home environment.	none

³ Remote access by human users to AAL Spaces, either service personnel of a service provider or the AAL space inhabitants when outside or any other person with appropriate permissions, can be done by normal Web access to AAL spaces through the gateway, because the gateway can activate the Web I/O-Handler for this purpose (see UIM). Hence, there is no remarkable work item in this regard in RIEG.

Local agent for providing services that can be accessed by special service providers (through the ASG) for administrating the AAL space remotely	node-level admin	Administration services related to the configuration of individual nodes in an AAL space	CNR-ISTI contributes to the specification.
	space-level admin	Administration services related to the configuration of an AAL space as a whole	CNR-ISTI contributes to the specification.
	communication with uStore	When the user decides to download a certain AAL application from uStore, then the uStore software needs access to certain administrative services in the AAL space	CERTH and Fh-IGD contribute to the specification. CERTH also plans about 1 PM for the implementation.
Tools (WP3?)	service profile generation	In order for the importing subcomponent of the ASG to proxy remote service agents, it needs their set of service profiles. This tool should automate the generation of the service profiles from agent's original API (e.g., from WSDL).	CERTH plans 3 PMs for the design and implementation.