



Project co-funded by the European Commission within the ICT Policy Support Programme

ReAAL

CIP ICT PSP – 2012 - 325189

www.cip-reaal.eu

universAAL compliance guidelines

[Deliverable D3.3, Revision 1.0]

Key Information from the DoW

Due Date	15-October-2015
Type	Report
Security	Public

Description:

This deliverable is a compilation of the coaching experiences in ReAAL, providing a monitoring and evaluation concept for the adaptation of products and services to the universAAL platform.

Lead Editor: Alvaro Fides (UPVLC)

Internal Reviewer: Ad van Berlo

Versioning and contribution history

Version	Date	Author	Partner	Description
0.1	24-Jul-2015	Pilar Sala	UPVLC	Structure of the document and task assignments
0.2	28-Sep-2015	Álvaro Fides, Michele Girolami	UPVLC, CNR	Aggregated all draft content so far
0.3	29-Sep-2015	Álvaro Fides, Michele Girolami	UPVLC, CNR	Added all available contributions. Rewritten common sections.
0.4	30-Sep-2015	Pilar Sala	UPVLC	Finalized Executive summary. Preliminary release for Review
0.5	30-Sep-2015	Saied Tazari	Fh-IGD	Intermediate version for project review
0.6	25-Oct-2015	Pilar Sala	UPVLC	Restructured content
0.7	9-Nov-2015	Pilar Sala	UPVLC	Updated section 4.1
0.8	11-Nov-2015	Saied Tazari	Fh-IGD	Updated section 2
0.9	12-Nov-2015	Alvaro Fides	UPVLC	Updated section 4
0.95	12-Nov-2015	Pilar Sala	UPVLC	Integrated contributions in final version, ready for internal review
0.96	14-Nov-2015	Pilar Sala	UPVLC	Applied reviewer comments. Final version ready for release
1.0	15-Nov-2015	Helmi Ben Hmida	Fh-IGD	Official release

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Executive Summary

This document is the result of the work in task T3.3 Coaching of application providers by platform experts. Within this task the platform technical experts have accompanied the application porting process providing training and technical support to the application providers and pilot investors with the aim of ensuring that the porting is done with good quality. The ultimate goal is that both the related knowledge is spread more widely and failure risks during deployment and operation are minimized.

To facilitate this process, platform technical experts have been assigned to each pilot site according to several factors, such as the proposed piloting concept, the geographical proximity or the shared idiom to facilitate communication.

Coaching basically relies on technical partners having a frequent communication with pilots and their technical staff, through many different means, such as teleconferences, physical meetings or asynchronous support; and it can be originated either by the technical expert, to support basic training or discuss project strategies, or by the application provider, to solve issues encountered during development.

The experiences in coaching have been quite varied, due to the fact that each pilot site has its own context and particularities, some have multiple providers that need coaching, others only one provider, some have more complex deployments than others, etc. For part of the pilots, off-line coaching was provided during the initial phase when companies started learning the universAAL basic concepts, while for others, more close, face to face sessions were organized. In some cases, pilots developers worked mostly autonomously until deploying the application while in others close cooperation was requested including support during deployment.

The lessons learned through this process have been useful to derive development patterns and best practices that will be shared with the community of developers to support anyone interested in creating universAAL compliant applications and services.

Table of Contents

1.	About this document	5
1.1.	Deliverable context.....	5
2.	Terminology.....	7
2.1.	Common universAAL terms.....	7
2.2.	universAAL – The open platform for open distributed systems a la IoT and Ambient Intelligence	8
2.3.	universAALization, what does it mean?.....	10
3.	The experience in ReAAL	13
3.1.	Lessons learned	13
3.2.	Development patterns and best practices	14
3.3.	Frequently asked questions.....	17
4.	Coaching process in ReAAL	20
4.1.	Methods and tools for coaching.....	20
4.2.	Pilots experiences	22
4.2.1.	AJT – SL	22
4.2.2.	AJT – WQZ	22
4.2.3.	BSA	24
4.2.4.	BRM	25
4.2.5.	IBR	25
4.2.6.	ODE	25
4.2.7.	PER	26
4.2.8.	PUG	27
4.2.9.	RNT	29
4.2.10.	TEA	30
4.2.11.	Associated Pilots	31
4.3.	Conclusion.....	32

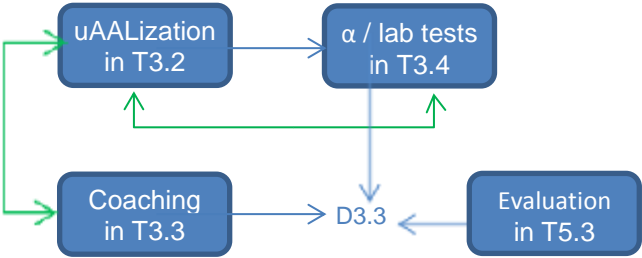
1. About this document

This document is reporting on the process and progress of task T3.3 Coaching of application providers by platform experts. Within this task, the platform technical experts have accompanied the porting process from T3.2 by providing training and technical support to the application providers and pilot investors with the aim of ensuring that the porting is done with good quality. The ultimate goal is that both the related knowledge is spread more widely and failure risks during deployment and operation are minimized.

The document provides first, an introduction about the terminology used in relation to the universAAL platform, describing the basics of the platform and the principles to adapt an application to be used on top of universAAL platform, which in the context of the project is known as *universAALization*. Then, an overview is given of the experience gained in ReAAL, that has enabled to develop compliance guidelines in the form of development patterns and best practices.

Finally, the information about how the coaching process in ReAAL has been established is provided, reporting on the methods and tools used by the different pilots and platform experts as well as the lessons learned from the process.

1.1. Deliverable context

Project item	Relationship
Objectives	Contributes to the following project objectives: O1: deploy at least 7 universAALized applications and services O3: an initial universAAL ecosystem O5: knowledge sharing O8: sustainable exploitation
Work plan	<p>The deliverable D3.3 universAAL compliance guidelines is an outcome of T3.3 Coaching of application providers by platform experts. The relationship with the other parts of the work plan is as follows:</p>  <pre> graph TD T32[uAALization in T3.2] --> T34[α / lab tests in T3.4] T32 --> D33[D3.3] T33[Coaching in T3.3] --> D33 T34 --> D33 T53[Evaluation in T5.3] --> D33 T34 -.-> T32 D33 -.-> T33 </pre>
Milestones	Contributes to the achievement of the following project milestones: MS4 – Final corrective actions
Deliverables	This is a single deliverable, however it receives input from the experiences in several tasks in addition to T3.3, such as T3.4 Lab Tests, or T5.3 Evaluation execution.

Exploitable results	Contributes to the following exploitable results of the project: Res 6: Guidelines for monitoring and evaluating the adaptation of products and services to the universAAL platform
Risks	Contributes to the clearance of the following project risks: Rk3: Difficulties and delays in the integration with the platform Rk4: Interoperability problems between different subsystems. Rk5: universAAL runtime platform failure during pilots

2. Terminology

2.1. Common universAAL terms

In order to better understand the following sections, the definitions of the most common terms are provided here:

Concept	Definition
AAL Space	A smart environment (in the sense of a physical space, such as a home, a car, an office, a supermarket, a hospital, an airport, etc.) with defined boundaries of hardware, software, and human resources (in terms of, e.g., access control, multimodal user interfaces, etc.), which is capable of responding to the needs of its users in an adaptive way by providing relevant assistance.
AAL Service	A concrete assistance value brought to people to help them live a full and independent life. To bring such a value, any combination of hardware, software, and human effort may be incorporated.
AAL Application	Software that implements certain use cases involved in the provision of AAL Services. It may consist of several software modules and / or utilize special-purpose devices (e.g., sensors, actuators, medical devices, etc.).
AAL Platform	Usually used to refer to the software that supports the creation of integrated AAL systems by providing a logical layer that facilitates the integration of interoperable AAL applications as well as a standard set of capabilities commonly needed by AAL applications.
Ontology	<p>In information technology, an Ontology is an explicit specification of a set of concepts and their relationships within a domain in a formal language that is developed with the goal to be shared by humans and machines as a <u>shared understanding of the domain</u>. Hence, ontologies are <u>explicit</u> and <u>sharable domain models</u> specified in a <u>formal language</u>.</p> <p>Being based on a formal language, ontologies become machine-readable, which enables computer programs to benefit from ontologies in data and information processing.</p> <p>Conventional computer programs usually include domain models, as well; but such models are not suitable for explicit sharing because (1) due to linkage to an associated program code, some assumptions in the code find no reflection in the model itself and remain implicit, (2) the formalism of programming language is not powerful enough for creating standalone models, and (3) share-ability remains confined in the scope of a certain technology.</p>
universAAL	The core part of universAAL that provides the logical horizontal

<p>Middleware</p>	<p>integration layer for creating integrated systems across distributed components and subsystems. It hides distribution and possible heterogeneity of the components and acts as a broker between the communicating parties. This way, it facilitates integration and communication and effectively provides for interoperability among components that may have been developed widely independently from each other. The kind of interoperability enabled by the universAAL Middleware is called “semantic interoperability” because it eliminates syntactical dependencies among the communicating components and reduces their dependencies to shared domain models or ontologies.</p>
<p>universAAL (communication) Buses</p>	<p>Part of the universAAL Middleware with the ultimate integration and communication logic that implements the semantic brokerage mechanisms and handles the flow of messages among the communicating parties. For each type of communication, there is a specific bus with its own model, strategy and match-making: (1) the Context Bus for event-based communication based on the subscribe / publish pattern, (2) the Service Bus for the general call-based communication based on the request / response pattern, (3) the User Interaction Bus (UI Bus) for the specific call-based communication when utilizing interaction services, and (4) the Control Bus for the specific call-based communication with regard to configuration management. It is these Buses to which applications connect in the process of integration.</p>
<p>universAAL Managers</p>	<p>A universAAL Manager is a software component that runs on top of the universAAL Middleware as part of the platform itself. universAAL is distributed with a set of such manager components that are necessary for its proper operation or provide relevant basic services and / or events to the other managers and / or to the application layer.</p>

2.2. universAAL – The open platform for open distributed systems a la IoT and Ambient Intelligence

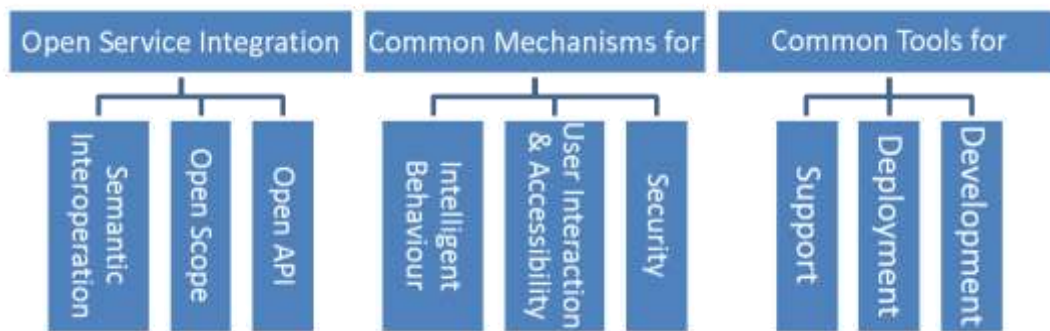
With its worldwide unique implementation of semantic interoperability for SoA at the level of communication protocol (existing since 2008), universAAL provides an open horizontal service integration layer at the highest abstraction layer, across all possible verticals. By avoiding domain-specific APIs (reduction of all possible syntactical dependencies to one single message brokerage API), universAAL has a unique future-proof contribution to managing the IoT complexity.

The open source software – distributed with the Apache Software License 2.0 – consists of:

- the distributed implementation of three "group-level" brokers (the context, service and UI buses) that ensure integration and semantic interoperability while hiding distribution and heterogeneity,

- a fourth broker supporting configuration management at node level (the control bus),
- a set of "managers" on top of that (providing for shared mechanisms for security, user interaction & accessibility¹, system memory, intelligent & adaptive system behaviour², remote and inter-group (also multiple with or without hierarchies) interoperability, and configuration management),
- a set of concrete ontologies,
- a set of development, deployment, and administration tools,
- example applications,
- documentation, wiki pages and training material.

Major features of universAAL can be summarized the following way:



With its 30+ applications, 100+ services offered by these applications to 5000+ users in 13 pilot sites in eight countries, ReAAL is serving as stress-tester of universAAL in real life. The socio-economic impact of universAAL is being evaluated in the first half of 2016 during operation in real life. In other words, universAAL as a domain-independent integration platform is, through ReAAL, getting ready for mission-critical deployments.

Available resources:

- The universAAL project in GitHub <<https://github.com/universAAL/>>: Holding the current source code and documentation wikis, including:
 - The main wiki of the 'platform' repo <<https://github.com/universAAL/platform/wiki>>
 - The Developer Handbook <<https://github.com/universAAL/platform/wiki/Developer-Handbook>> (previously known as Reference Documentation)
 - The Quick Developer Guide <<https://github.com/universAAL/platform/wiki/Developer-Guide>>

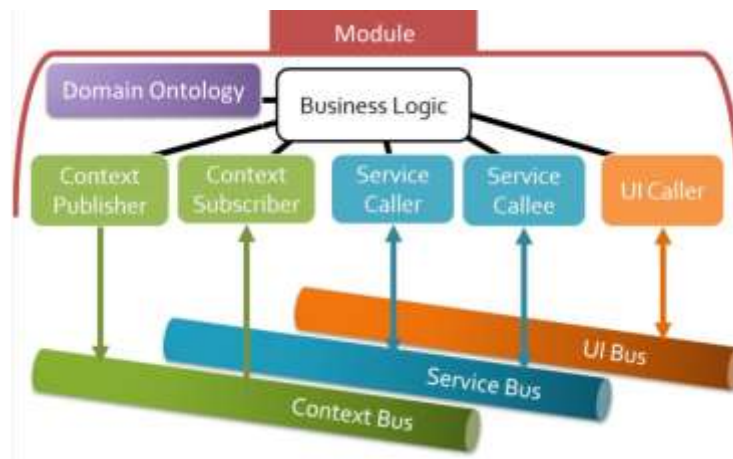
¹ The user interaction framework is specifically designed and developed for smart environments and already has an internationally recognized status [IEC/PAS 62883 Ed. 1.0] as an open approach with considerable potential to become an important standard in near future.

² Architecturally, the framework supporting context-awareness and profiling has a very open approach that enables to achieve an intelligent and adaptive system behaviour.

- The Quick Setup & Start guide
<<https://github.com/universAAL/platform/wiki/DG-Quick-setup-and-start-guide>> for use with the downloadable package below
- The universAAL depot <<http://depot.universaal.org/>>: A complementary website for developers who want to develop applications on top of universAAL or contribute to its maintenance.
- The universAAL GForge website <<http://forge.universaal.org/>>: The old project management website for universAAL software, which still holds the issue trackers, including
 - The "support" issue tracker, the main entry point for general issues
<<http://forge.universaal.org/gf/project/support/tracker/>>
 - The public forum for "help" issues
<http://forge.universaal.org/gf/project/support/forum/?action=ForumBrowse&forum_id=8>
- compact overviews: <http://tinyurl.com/future-proof-AAL-spaces> and <http://www.cip-reaal.eu/about/the-platform/>
- Mailing-List <<http://universaal.aaloa.org/mailman/listinfo/universaal-dev>>: The universAAL mailing list for both users and contributors
- universAAL demo <<http://universaal.org/index.php/en/demo-corner>>: Demos produced at the end of the universAAL EU project
- Maintenance sponsors <<http://universaal.org/index.php/en/consultancy-services>>: Organizations contributing to the universAAL maintenance and providing support, training, and consulting services with regard to universAALization processes and the development of specific versions of universAAL, either for specific runtime environments or for any other specific requirements.

2.3. universAALization, what does it mean?

universAALization is a term used in ReAAL to refer to the integration of all functional components to be deployed with the universAAL software platform prior to their deployment. Goal of universAALization is to resolve the dependencies between these components for the exchange of data and functionality based on an open platform in order to achieve future-proof interoperability, adaptability, and extensibility to a wider extent.



universAALization consists of two major steps: (1) selection and completion of a set of application-related ontologies, and (2) using the application programming interface (API) of the universAAL middleware together with the selected ontologies for the actual integration of the components.

With the latest versions of universAAL used in the ReAAL project, the following checklist could be used for creating a universAALization plan:

General approach

<please check-mark the applicable options (multiple choices possible)>

- Using universAAL runtime with the Java-OSGi API of the middleware
- Using universAAL runtime with the Java-Android API of the middleware
- Using universAAL runtime with ASOR (Java from within JavaScript)
- Using other runtime with the Java Remote-API of the middleware

Usage of the middleware buses

<please check-mark the applicable options (multiple choices encouraged)>

- Context bus (for publishing events and / or subscribing to events)
- Service bus (for calling and / or providing services)
- User Interaction bus (for using universAAL's UI description package and leaving the rendering to situation-aware UI handlers)

Additional middleware features planned to be used

<please check-mark the applicable options (multiple choices possible; the first two are encouraged to be used)>

- Multi-language support
- Configurability API (mainly management of config parameters and config home directories for storing files and resources and accessing them)
- Logging mechanisms

- Multi-tenancy support (server-based usage of universAAL to connect to and serve several homes)
 - Functional Manifest (each module can contain a digitally signed “functional manifest” that is used for getting user consent – similar to the Android permissions system that gets active when you decide to install a new app, then Android lists the permissions that the app claims to need and you decide if you will install the app or not)
- The AAL Space Management API (info about the available middleware instances, installed modules, etc.)
- Serialization and parsing API (currently only for RDF Turtle syntax)

Using universAAL “Manager” components (platform services)

<please check-mark the applicable options (multiple choices encouraged)>

- Context History Entrepôt (CHE) services (querying data gathered in the home)
- Profiling services, including storage and retrieval of data that describes users, objects, locations (builds on CHE)
 - Resource Manager (important only if at the middleware level, you plan to use the UI bus; in that case, you can achieve a higher level of adaptability if you let the Resource Manager store your media objects that you want to be used when interacting with the user)
- Situation Reasoner services (builds on CHE – the Situation Reasoner can store SPARQL CONSTRUCT-queries as rules to automatically generate new context events whenever certain conditions hold; this can be used to recognize situations; e.g., if you want that a context event is published whenever the user is sleeping, a solution could be to tell the SR to publish this event whenever in the night the user is in the sleeping room in the bed and the lights are off)
- The Drools Engine (a second reasoning engine using the JBoss rules)
 - ASOR Scripting (ASOR stands for AAL Space Orchestrator; with ASOR scripts, you can create composite services – combinations of existing services – and define rules that specify the automatic reaction of the system to certain situations)

3. The experience in ReAAL

3.1. Lessons learned

As it can be seen from section 4, the coaching experiences are as varied as the pilots themselves. In this section, a summary of the main conclusions is drawn from the report of the pilots.

Related to the process of universAALization and coaching support:

- Developers approaching to a new platform such as universAAL need first an off-line phase during which they can learn about the platform and the related component before starting any (pre)development steps. The quality of the documentation is key at this stage to keep the motivation of the developer.
- As the platform is very rich in number of features and help methods, the technical team must well learn about the platform and looks for the available features before deciding developing his owns
- The learning curve is steep at the beginning and need some effort to learn the basics. However, the ability of developing applications on the top of the platform quickly evolve with the practice.
- Training and coaching are key activities to ensure the success in adapting/developing uAAL applications, but they need to be done by a universAAL expert, otherwise common mistakes and pitfalls translate into the final product.
- The most efficient way to coach a team that requires platform knowledge is using continuous communication. The forum of the platform is vital for knowledge sharing but on-line seminars and Skype calls are most effective tools for answering to questions from developers.
- There is the need to ease/customize the uAAL applications development environment to maximize its acceptance from the developer side. Developer support tools for ontology design, test and debugging, as well as for configuration, deployment and maintenance are crucial for wide adoption of the platform.
- Main topics for support has been requested to setup uAAL server and environment, and showing how to use the platform without using Eclipse, as well as modelling ontologies for the pilot applications.

Related to ontologies:

- It is very important to study which exact ontologies to use, and how to extend them, to achieve an expressive and generic ontology that fit the pilot/application needs, before starting any implementation activity.
 - For the coaching team, it is important to keep a look on the updates during development, especially during the ontology implementation, their optimization is key to the success of the universAALization and for the capability of implementing the desired level of interoperability.
-

- Typical mistakes have included the disregard of existing ontologies that could help develop a more complete modelling of the solution as well as the definition of very technical ontologies, falling in the category of data models rather than full semantic framework for the system. Common mistakes included the direct translation of interfaces to ontological concepts.

3.2. Development patterns and best practices

Analysis and design

The universAAL platform supports both local and server-based (cloud) scenarios, or a combination of these. The first analytical step of deployers will be to identify which of these options corresponds to its initial architecture.

- **Client-based:** All logic is run in devices deployed at the client side, whether that is fixed or mobile devices at the user's home or in mobile device he carries around.
- **Server-based:** All significant logic is run in a server run by the deployer. It may be the case that some amount of processing is done at devices in at the client-side, but only for the purpose of sending the information to the server (e.g. Sensors sending data), not performing any processing on it.
- **Client-Server:** Some data processing or business logic is performed in devices at the client side, while other happens in a server. This may be working in conjunction or separately in independent solutions (e.g. deploying together client-based applications and server-based applications).

Identifying which scenario is being considered will help answer the following questions about the deployment of universAAL-aware nodes (devices that can run universAAL):

- **Where do applications run?** Identify in which devices and execution environments the applications (business logic) are running.
 - **Where can universAAL run?** Right now, the universAAL platform can be executed in devices running Android 2.3 or above, and in Java Virtual Machines (JDK 7) through an OSGi Container. In the latter case, the hardware requirements are as low as mini-computers like the Raspberry Pi, while for servers the requirements will depend on the number of clients to service.
 - **Can applications be universAALized?** In cases where universAAL can be run in the same device like the deployed applications, consider if it would be possible to universAALize these applications. The "universAALization" (or "uAALization") consists on having the applications communicate with a universAAL instance, whether through "native" interfaces (Android apps and OSGi bundles can do this) or through alternative methods (universAAL's Remote API).
 - **How are universAAL instances connected?** If universAAL instances can be deployed in more than one node, determine if and how this communication will happen. Instances in the same network can communicate directly to each other. In different networks (or through the internet) they can communicate with Gateways or Remote API. Or not be connected at all.
-

Answering these questions does not yet provide the right choice for universAALizing, but will help taking the final decision on which of the 4 common universAALization options, and how to implement it. These uAALization options provide value to the deployer in different ways; it is how the deployer benefits from having universAAL. It is possible to combine more than one. What options to take will depend on the aim of the deployer, the client/server scenario identified above, and the restrictions identified by the previous questions.

- **Decouple devices from applications:** Instead of hardwiring devices to applications, the communication is abstracted thanks to uAAL, which allows to connect newer devices from other technologies without modifying the applications, and allow other applications to benefit from existing devices.
- **Decouple applications from each other:** Instead of hardwiring the communication between applications, it is performed through universAAL, which allows to connect new or different applications providing or consuming the same type of information, without having to change them.
- **Decouple applications from servers:** Instead of hardwiring the communication of client-side applications towards the server side, the communication is performed by universAAL, so for the client application it is like the server application is being run locally.
- **Virtualize applications in the server:** If universAAL cannot run in the client side, an instance of universAAL run in the server can virtualize this client, allowing client applications to benefit from universAAL even if it is not run locally.

To consider that a solution (application, service or a combination) has been uAALized, at least one of the options must be implemented.

Implementation

Once it is known where to run universAAL and how and what to connect its instances to, it will be clear which application modules have to be uAALized (understanding application modules as pieces of executable business logic, not the overall application, which can be comprised of many).

The process of uAALizing an application module involves making it run in a uAAL-compatible container (Android, OSGi) and perform/allow some interaction through uAAL buses. When an application module cannot run in those containers but still has to be uAALized, the Remote API has to be used. In any case, it is imperative that the interaction with uAAL is made in terms of ontologies.

Design ontologies.

Ontologies are the data model in universAAL. In order to communicate with universAAL the data exchanged must be represented by ontologies. Deployers must analyse the original data model of the application, and extract the basic information needed for the business logic, to a bare minimum, need-to-know basis. This helps identify the meaningful parts of the data that need to be shared (the “semantics”).

Ontologies are application- and hardware- agnostic, which means they should not model things like raw values of sensors, application-specific constants and the like, nor used to stream data (even at low frequencies). It must always be taken into account that other deployers should be able to understand the data model to work

with it without knowing anything about how the application or hardware work (even if nobody else will use it in the end). Deployer should aim at having an ontology that is as future-proof as possible to reduce later needs of updates.

Once it has been identified what kind of data the application is going to handle, it is necessary to inspect existing uAAL ontologies to see if there is one that can already be used. It could be that:

- There already is an ontology that covers all the needs of the application: Use that one.
- There is one that covers the needs of the application but would need more concepts: Use that one and create an extension for it. Ontologies allow extensions: additional modules that can define new concepts that “inherit” or “extend” existing concepts in other ontologies.
- There is none that covers the needs of an application: Create a new ontology specifically for the application, but trying to also make it as generic as possible so that other applications in the future can benefit from reusing it.

There are links on how to create ontologies form scratch or extending existing ones (which is practically the same). If planning to use the Service Bus, a service ontology must be included (if an existing one cannot be reused, which is recommended).

Modifying application modules

When all needed ontologies are identified and ready, the application module must be modified to make use of them and the universAAL APIs for connecting to the buses. This process depends on the type of universAAL-compatible container the application module will run in:

A) OSGi Container

- 1) Application code must be turned into an OSGi bundle (or set of bundles depending on application module architecture). Bundles contain an Activator that starts up the logic. UniversAAL tools and documentation are based on Maven projects. In these, POM files configure the project and dependencies. Dependencies should be taken from existing bundles (like Maven repository or uAAL Nexus) or else provide them in an alternative way. POM file can be standalone, extend its own parent, or extend a uAAL parent (not recommended unless for special cases or ontologies).
- 2) If possible, deployers should make sure the app still works as expected when packed as a bundle and run in OSGi, before trying to uAALize the code.
- 3) Import dependencies as needed from uAAL bundles and the right ontologies.
- 4) Create the needed uAAL wrappers (SCee, SCer, Csub, Cpub, UICaller). Make sure you use the right ontologies in the right way.

B) Android container

- 1) Choose whether to run uAAL app separately, or embedded in your app
- 2) Decide if modifying uAAL app with default configuration and onts, or set it up from your app
- 3) universAALize the app following the instructions in the wiki.

Once the application modules are uAALized, perform tests to check that works properly, checking logs looking for expected messages.

3.3. Frequently asked questions

Here are some of the recurring issues the pilots have come across. These have been brought up to the attention of the coaches or other support members through many sources: Support forums, Issue Trackers, Developers mailing list, direct contact with coaches... (Not counting bugs or feature requests).

The intention here is not to answer the questions themselves but how they were answered and why the issues they represent were found so often. The questions are sorted by frequency (more frequent first) but in a very loose way: it is difficult to determine actual amount of occurrences given all the different sources of the questions, and the fact that some may be related to others, or from different sources.

How can I migrate existing ontologies to universAAL (and back)?

There were many requests for explaining how to move from existing OWL files (files for defining ontologies) to universAAL-compatible ontology representations (OSGi bundles), and also the other way around. This is a topic of interest because in the cases where developers already know ontologies, they usually have some existing OWL files they would like to use. And another reason for the question to pop up is that the process that allows this is a bit convoluted, requiring separate steps with some of the tools of the AAL Studio plugins – whose documentation was hard to find and follow. In addition, there were a few bugs, too.

The actions taken to minimize these issues were to update and fix the tools, rework the documentation, and lead to the right links in the wikis. However, since a full refactor of the migration tool has been planned (whenever there is time), the usual recommendation has been to implement the code of the ontology manually – taking the chance to refine the ontology. Which leads to the next frequent question.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?forum_action=ForumMessageBrowse&thread_id=120&action=ForumBrowse&forum_id=7

How do I model a new ontology?

Ontologies are one of the most common entry barriers for using universAAL, as they require changing the way developers think about data models. This is why developers have asked for support on how to design the ontologies they need and implement them in universAAL-compatible bundles.

Regarding design, it is only possible to assist developers on a per-case basis, and give general advices (some are found in this deliverable, for instance). Regarding implementation, developers were redirected to appropriate documentation links.

Example tracker link:

http://forge.universaal.org/gf/project/ontologies/forum/?action=ForumBrowse&forum_id=126

Where can I find an example and how can I run it?

Some developers expect to find an example that is ready to download and run. This is a common practice with all libraries, platforms and open source projects. In universAAL there is the “Lighting Example”, but documentation is often focused on code development, not on how to simply run it.

New documentation pages were created that explain the minimum steps needed to get a universAAL instance running, and how to add modules (such as the lighting example) into it.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?_forum_action=ForumMessageBrowse&thread_id=135&action=ForumBrowse&forum_id=8

Where can I find the module [X] and how do I use it?

There were several questions asking for information about specific modules (managers like CHE or Gateway, ontologies, examples...). These are usually solved by redirecting developers to the right wiki page that explains the module. After the rework of the documentation pages this should be easier to find by developers themselves.

Where do I start?

In some cases new developers do not know exactly where to start reading to know how to use universAAL. This is part of the ongoing trouble with documentation clarity. After the rework of the documentation wiki pages, they are usually redirected to the GitHub Platform repository Wiki page, which is now highlighted in all places as the main entry point for technical documentation.

How do I prepare my own deployable instance of universAAL with my applications?

The final step in any deployment of the pilots is to actually deploy the universAALized applications. In this case it means preparing a Karaf folder with every needed bundle already installed, or, in case of deploying on Android, packaging the APKs.

There are already some documentation pages dedicated to these issues, but since every pilot is considerably different from each other, some required more specialized instructions, case by case.

How do I convert my existing application code to something I can run in uAAL? And how do I deploy it?

This is part of the implementation phase of the universAALization process, explained in this deliverable too. It is part of the coaching tasks to assist pilot developers with this. Again, because existing code varies so much from pilot to pilot, there is no generic answer.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?_forum_action=ForumMessageBrowse&thread_id=148&action=ForumBrowse&forum_id=7

What hardware sensor technologies does uAAL support?

One for the aspects of universAAL that raises more attention from developers is its support for sensor networks and domotic devices. Each hardware technology requires an adaptation (an “Exporter”), and in the universAAL platform there are already some existing exporters, listed and explained in the LDDI repository documentation.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?_forum_action=ForumMessageBrowse&thread_id=125&action=ForumBrowse&forum_id=7

Can I run uAAL in this particular execution environment or device?

One of the questions, described in this document, that need to be answered when planning the universAALization of applications is if it is actually possible to run universAAL in whatever device will be used for deployment. The universAAL platform can run in most JVM, so any device with a running environment capable of this should be able to run universAAL (including Android). Of course, there are particularities with every environment and these have to be solved one by one.

How do I use multi-tenancy to connect universAAL instances remotely?

One of the features that were added to the platform specifically for ReAAL was multi-tenancy, which allows to connect or virtualize several “tenants” (or remote nodes) to a central server instance. Because it was made during ReAAL, until all documentation was ready, pilots would have to ask for support on how to make use of it.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?_forum_action=ForumMessageBrowse&thread_id=163&action=ForumBrowse&forum_id=8

How do I run the Android version of uAAL?

Some pilots deploy universAAL in Android devices. Because it is quite different from the usual Karaf-OSGi deployment, it requires special care. Developers were redirected to the specialized documentation, and such documentation was also improved.

How can I make sure a universAAL service call will succeed?

Because service calls are semantic, it is often possible that calls will fail to find the service they were looking for if the call or the service were not properly defined. It is one of the most common issues found while coding uAAL applications. Coaches were available on demand when developers found these issues and helped debug and solve the problem.

Can I develop universAAL applications on Mac?

Some use Mac computers to code. In the past, there have been issues when installing the recommended IDE (Eclipse) for developing with universAAL. These were due to issues with one of the build tools (Maven), which is not the responsibility of universAAL platform. There are other ways to code the OSGi bundles (or Android APKs) that don't require this IDE and build tool, but the universAAL documentation cannot cover all possibilities. However it seems that latest versions of the IDE may no longer face this problem. As for running universAAL itself for testing, it should work just like Linux.

Example forum link:

http://forge.universaal.org/gf/project/support/forum/?_forum_action=ForumMessageBrowse&thread_id=159&action=ForumBrowse&forum_id=8

4. Coaching process in ReAAL

Coaching process in ReAAL was established with the goal of supporting application providers in their way toward successfully porting selected applications to work on top of universAAL platform.

To facilitate this process, platform technical experts have been assigned to each pilot site in order to provide training and technical support to the application providers and pilot investors in all the matters related to the adaptation work, with a twofold aim, from one side to ensure the good quality of the porting and on the other side to widely spread the knowledge about the platform.

The matching of platform experts and pilots have been done according to several factors, such as the proposed piloting concept, the geographical proximity or the shared idiom to facilitate communication. The assignments are as follows:

Pilot site	Platform expert
AJT - SL	Fh-IGD
AJT - WQZ	Fh-IGD
BRM	SINTEF
BSA	UPV
IBR	UPV
ODE	MEDCOM, (UPM)
PER	TRIALOG
PUG	CNR
RNT	SmH
TEA	UPM
Associated Pilots	TRIALOG, (UPV)

Selection of different modalities for such coaching activities have been done according to the concrete needs of each pilot site, and in case of need a backup coach have been assigned to complement the expertise.

4.1. Methods and tools for coaching

Coaching consisted on technical partners having a frequent communication with pilots and their technical staff, through many different means. There are two types of events in which coaching was performed, identified by the originator of the communication:

- Coaches would arrange meetings with coached people during the “initial steps”, in order to introduce pilots to the technicalities of the platform, but also during any generic update on the strategies of the project, like for instance a revision of used ontologies to enhance the adaptation to universAAL.

- Coached people would arrange meetings with coaches whenever a technical problem appeared during the adaptation, or when further explanations or support were needed after one of the coach-triggered meetings.

Meetings were conducted through several means:

- **Teleconferences:** These would take place between individual coaches and their assigned pilot staff, or all together (although the latter was less usual). This was the most common type of coaching sessions. The tools used were mostly Skype and GoToMeeting, often using the screen-sharing features.
- **Physical meetings:** Whenever needed, usually on demand of the coached people; in addition, coaches would take advantage of physical presence of pilots in the regular project meetings proactively, be it plenary meetings or training events.
- **Asynchronous support:** Not actual meetings, but asking for help and getting response, through emails, issue trackers or support forums, making use of collaborative documents or sharing source code.

The archetypical process by which a coach would assist a pilot to perform the adaptation to the universAAL platform usually followed these steps:

1. Get in touch with the pilot representatives, which would point the coach to the staff and developers responsible of the actual adaptation.
 2. Point the developers to the universAAL documentation and have them read the basics to understand the platform. Then have them try to run the examples.
 3. Understand the application and proceed to designing an adaptation scheme and strategy. In ReAAL practice, the result of this step has been documented in D3.2a.
 4. At this point developers would start implementing the adaptation. At any point in time, if they found any trouble, they could ask the coaches for help. This could include:
 - a. Explain, if needed, how to turn existing code into universAALizable components (OSGi bundles/Android apps).
 - b. Assist in designing and developing ontologies if needed. Check that the ontologies designed by pilots (if any) are correct, compliant with the principles of the platform, and build on existing ontologies.
 - c. It may be necessary to assist in code development, especially with the Service Bus. To check that wrappers work as expected (received events, matching calls...)
 5. Assist with any advanced manager that may be used (CHE, Remote API, Gateway,...), pointing to the right documentation, providing examples or helping with the code.
-

4.2. Pilots experiences

4.2.1. AJT – SL

The Smart Living pilot in ReAAL has been coached by Fraunhofer IGD.

A SL developer attended universAAL training on January 16, 2014, in Barcelona. Then, two initial virtual meetings were organized, the first one for understanding the Smart Living system architecture better and the second one concerning the universAALization strategy.

In Two following face-to-face meetings, a concrete universAALization scheme was first agreed upon, as documented in D3.2a. Based on this scheme, it was obvious that SL had to make use of two new universAAL features, namely the Remote API and the scripting language ASOR (AAL Space Orchestrator). Therefore, a complementary training on these features as well as refreshments on other basic features, such as ontologies and the usage of the communication buses, were the second topic in the F2F meetings. Last but not least, the concrete ontologies in the context of the SL universAALization scheme were discussed thoroughly so that SL could start to develop them.

As a result of the F2F meetings, Fraunhofer IGD took over to demonstrate the universAALization of the SL services with one complete example. After delivering this example and discussing its details, the communication mode was changed from mainly meeting-based to mainly email-based in terms of continuous support, not only during the development phase, but also in the test and deployment phases.

After finishing the universAALization process at SL, a specific physical meeting was organized in order to discuss the criteria for selecting a concrete application from another pilot site as the “imported application” at SL. As a result, a ranked list of recommended alternatives was created so that SL can choose from among them.

The coaching experience with SL showed that it is not always obvious how to benefit from universAAL when a simple closed solution not depending so much on third party components is going to be ported to universAAL without having any concrete plans to make use of the cross-domain data analysis and service composition potential. In case of SL, the flexibility in future extensions and experimentation with service composition have been identified as the main motivation for making use of universAAL.

In addition, once more it was realized that it was very important to put the right emphasis on the design and development or selection of the right ontologies fitting the application context.

A recommendation that can be made to the universAAL developers is to consider alleviating the complexity of the development environment in order to improve its acceptance by new developers.

4.2.2. AJT – WQZ

Concerning the WQZ pilot side, it has coincided that the technical partner and the supplier of AJT are the same (FhG). Moreover, the coaching process has well taken place internally between the “FhG coaches” who present the main developers of the uAAL platforms and the “FhG-WQZ developers” who have universAALized the home

management and the CapFloor applications. In this context, the main coaching activities can be summarized as follow:

- 1- Common meeting with WQZ pilot site and its original technical supplier (inHaus) to discuss the original system architecture
- 2- 1 day internal meeting to agree on the uAALization schema and the uAAL system architecture
- 3- F2F discussion to extend the original ontology, thus to fit the WQZ “world” requirement: addition of the Posture and the activity ontologies, same extension of the Physical word and the AAL space ontologies.
- 4- Support for the conception and design of the WQZ created rules as an extension of the original system
- 5- Support of the system extension with 2 new protocols (SIP and SMTP) add to the original PLC one
- 6- Continuous support in case of bugs during the whole lifecycle of the WQZ pilot site

Lesson learned and experience:

- 1- It is easier to learn about the platform while being in touch with a core member of the uAAL developer teams.
- 2- It is very important to learn about the platform and the related component before starting any (pre)development steps
- 3- Training is a pillar toward a successful use of the Platform and its component
- 4- The uses of the platform at the beginning need some effort to learn the basics. In the meantime, the ability of developing applications on the top of the platform will quickly evolve... As the platform is very rich in number of features and help methods, the technical team must well learn about the platform and looks for the available features before deciding developing his owns.
- 5- The other very important lesson learned was mainly related to the required level of development to be able to experience the Platform advantages. In fact, while developing the initial modules based on the platform in order to satisfy the minimum requirement of the services, the developer experiences very small advantages compared with the original development environment. Once the minimum set of applications is created, the service provider, same the developer experience a huge advantage in using the platform to extend, adapt, adjust and transfer the created application.

Suggestion for improvement

- The re-activation of the uStore and the uCC as main two components for facilitating the marketing and the deployment of uAAL based solutions
-

4.2.3. BSA

The coaching was performed by UPV to the company subcontracted by BSA, which was in charge of actually developing the applications and their adaptation to universAAL. This company (TSB) already had certain experience on dealing with universAAL. At the same time, TSB had to deal with the technical staff of BSA remotely when it came to deployment of the actual applications and maintenance. There were several particularities and special considerations to take into account in this pilot, which made the adaptation to universAAL, and as a result the effort needed for coaching, a bit more demanding.

Ontology design has to be careful due to being like the “API” in universAAL, which means that critical changes in the ontologies may require changes in the apps using them. This is particularly crucial given the imposed inability of updating one of the applications (NOMHAD) once it was deployed at end users’ homes. This adapted application has to remain the same regardless of released versions of the platform due to a limitation in the remote updating functionality of the original commercial application – no changes can be made after it was initially deployed unless technical support people goes to end-users’ homes to do it manually.

It was the first pilot to attempt the combination of universAAL on Android devices in the user side connecting to universAAL in the server side through the Remote API. This means it was also the first to discover bugs in the operation of the systems involved. It must be noted that the Remote API was created by request of the ReAAL project and had not been tested before.

A consequence of the above is the divergence in versions and even codebase (for universAAL Android App) in the pilot against the official trunk. Bug fixes can only be promptly provided through SNAPSHOT versions, but SNAPSHOTS are prone to introduce failures and should not be used in deployment. But the pilot still uses early SNAPSHOTS because of the limitation in the app explained above.

The particularities of the Android version of the middleware make it, apparently, more difficult to adapt existing applications to universAAL. But not because of universAAL itself: Even if adapting an Android app can be a tricky task, some Android apps being used in this pilot were already adapted or developed precisely by universAAL project: the Help When Outdoor and the Oximeter/BloodPressure drivers of NOMHAD.

In the case of the Agenda application, it was developed by TSB from scratch for ReAAL, designed from the ground up to be universAAL-based. The fact that it is a new application increased the number of bugs encountered during its development and deployment which cannot be, when discovered, identified as being of either the application logic or universAAL platform.

In all cases, the adaptation of the Android apps was assisted by UPV in order to speed up the process – especially taking into account that UPV created the Android version of universAAL. However, this in turn was detrimental to the coaching of the pilot in that the partners did not undergo the same level of self-learning other pilots went when dealing with the Android version.

Another obstacle deploying in Android is the particularities of the OS itself, regarding libraries, cloud messaging (used by Remote API), packaging, publication or updates. The Android version of universAAL was designed to be an independent application that works as a hub for the apps in the device, but to facilitate the deployment it was turned into a library and embedded into one of the applications (the Agenda).

4.2.4. **BRM**

The coaching of BRM pilot site was performed by SINTEF. Before withdrawing from the project, BRM pilot site decided to import two of the applications from BSA, Help when Outdoor and Agenda, changing them to be tailored to BRM context and complement it with an additional application for monitoring users at home.

As the application provider was the same as in BSA case, the technical coaching that SINTEF performed was in supporting the pilot in defining the needed changes required to tailor the applications to the piloting context as well as in the requirements of the technical infrastructure needed to set up the deployment environment.

4.2.5. **IBR**

UPV coached the Ibermática without major particularities, following the steps described in the first section as usual. Several telcos were held whenever needed, with the developers and representatives of the pilot, concerning the following topics:

- A general introduction to how universAAL works, and how applications can be made to run in the same environment and interface with the buses.
- How to design ontologies – in particular studying which exact ontologies to use, and how to extend them. As a consequence, Ibermática identified existing ontologies that could be reused, with a minor extension to cover the data model of one of the applications.
- Since the device used at the client-side (Assisted Person's home) was a Raspberry Pi, Ibermática needed particular support for running OSGi and universAAL in its Operative System. Feedback from this issue helped refine documentation.
- Other application runs in Android, so the same support was needed for running the Android version of universAAL. Adapting the application to universAAL in this context is a bit trickier, so help was provided to adapt the code. It also helped that it was UPV the partner that had developed the Android version of universAAL.

After the initial phases of adaptation to universAAL, when it was requested that pilots should enhance such adaptation to be more compliant to universAAL development principles, some discussions took place to agree on possibilities for improvements.

During the process of determining imported applications, the technicalities of importing were discussed and a set of applications was determined that would be compatible with the existing pilot infrastructure.

As a minor side note, it was necessary in some cases to share “diagram-like” information. A tool would have been useful – instead the discussions relied on screen sharing and using whatever tool at hand.

4.2.6. **ODE**

Coaching for ODE pilot have been performed by MedCom, however, as it had no previous experience with universAAL, UPM has been assisting in the more complex

topics, such as the ontology design, what has been translated in a coaching more difficult from a managerial point of view.

An on-site two days training event was organized and carried out by universAAL experts, to support the application providers in setting up the development environments and understanding the basics of the platform.

From that moment on, the coaching was performed according to the usual methods described in the first section. Whenever a more complex issue has arisen that the main coach was not able to provide support, the backup coach was involved to speed up the solving process. In particular, additional support has been requested to setup uAAL server and environment, and showing how to use the platform without using Eclipse, as well as modelling ontologies for the pilot applications.

ODE managed to achieve adaptation by developing preliminary ontologies, however after the expert coaches reviewed these ontologies many changes were proposed. Preliminary ontologies were very technical, falling the category of data models rather than full semantic framework for the system. Common mistakes included the direct translation of interfaces to ontological concepts, this method is not always the most advisable, as normally interfaces represent a very specific interaction between two components rather than the description of the interaction itself or the context in which the interaction takes place. One way to pick up the ontologies where not correct was the presence of strings encoded in an object serialization standard (in this case JSON) which is indicative that the objects contained could be modelled by the ontology itself. Another give away was the use of fixed length arrays, where each index had different meaning, this is not an ontological model as it cannot assign different meaning to each index; the correct way to do this would be to model independently each index as its own concept and then grouping them in another container concept (if needed). Typical mistakes also included the disregard of existing ontologies that could help develop a more complete modelling of the solution.

ODE was always opened for suggestions and changes, in fact the new ontologies were developed using all experts' suggestions; they modelled two types of step counters (something it was challenging even for experts), and included existing ontologies such as the device or health profile ontologies. The new model helped developers and managers in the ODE pilot appreciate the value of universAAL, as their system is now compatible with other applications. The new model made the use of the platform far easier, as the lower level details were hidden on top level business logic.

ODE development was very proactive, and required little assistance in the implementation phase. Probably due to previous experience, but fundamentally due to the new ontological model.

One important lesson learned is that coaching needs to be done by a universAAL expert, otherwise common mistakes and pitfalls translate into the final product.

4.2.7. **PER**

The case of the PERCHE is quite specific compared to the others. Its late arrival prevents them to propose the same process (a complete universAALisation of existing application) but a new approach based on the "imported App" concept. Therefore, the goal of PERCHE was not directly to universAALize existing Apps but

to study how imported uAALised components (and Apps) can be integrated into an existing AAL service.

TRIALOG, the pilot leader, has a strong experience with AAL applications development (for instance they were the technical leader of the FP6 MonAMI) and also with uAAL as developers of AAL Application. Therefore, the analysis performed in July on the Agenda and HWO applications design has concluded to decouple and re-implement the User management and interface part (independent on uAAL) and to keep as it is the uAAL architecture of the App. After a first period of deployment and based on the user feedbacks, it is possible that Trialog in collaboration with his subcontractor will go deeper into uAAL stuff to extend the current functionalities of the Apps.

4.2.8. PUG

Introduction to the Italian Pilot Site

The Italian pilot site is hosted in the Puglia region. It represents one of the most articulated pilot sites joining the ReAAL project for two main reasons:

- Number of involved companies
- Number of AAL applications available in the ReAAL portfolio.

Differently from other countries in Italy we did not have one single company developing one or more universAAL-based applications, rather the Italian procedure for selecting the application providers had to be achieved with a public procurement.

The procurement has been published during 2014 at the end of which 6 different companies answered to the public call and now they are joining the ReAAL consortium as associate vendors. The companies are:

- Ingel
- SteelMinds
- Virtech
- Cupersafety
- eResults
- BioResult

Moreover, the applications (or modules) developed by the companies have been clustered in three main areas of interest: Safety-at-Home, Home Activity Monitoring and Easy Home Control. The following table reports the name of the modules developed by each of the companies:

Area	Module	Company
Safety-at-home	Safehome	Ingel
	Electrosafe	Steel minds
	iHelp	Virtech

Home Activity Monitoring	Indoor Monitoring System	ERESULT SRL
	Environmental Monitoring System	ERESULT SRL
	Omnicare Health Check	BIORESULT SRL
Easy Home Control	NewDom	CUPERSAFETY SRL

Coaching Experience

The number of companies involved as well as the number of modules developed led CNR-ISTI to approach the coaching activities with a structured approach. The coaching is split in three phases closed in a loop (the description of the coaching procedure is available [here](#) in Italian language):

1. Off-line training: during this phase the companies started learning the universAAL basic concepts including the available tutorials, examples and the API documentation. Such resources have been mostly prepared in the context of the universAAL project. CNR-ISTI collected in one single start-up document a step-by-step procedure for getting ready with universAAL.
2. On-line training: during this step CNR-ISTI prepared a simple mechanism to allow the companies to ask for questions, suggestions and requesting for live seminars. In particular, we adopted the following mechanism:
 - a. A company first opens a ticket on the Italian tracker hosted on the GForge platform. The tracker collects the tickets only from the Italian companies. For every new ticket, CNR-ISTI receives an email notification so that it is possible to understand the kind of question, get ready with the proper documentation and (hopefully) to group together different tickets concerning the same topic.
 - b. After the reception of the email notification CNR-ISTI answers to the companies posting a message on tracker or, if necessary, via skype calls. This last option (Skype calls) has been the most useful and practical mechanism to provide support to companies.
 - c. In addition, CNR-ISTI has prepared two live seminars concerning the some basic concepts of the universAAL platform as well as some live demos to better explain how to use the universAAL API.

After such training steps, each company had still the possibility of requesting for further support.

During the coaching the activities we received 5 tickets as shown in Figure 1, that requested different level of support:

- Email exchange
- Skype calls
- Coding of specific examples
- On line seminars

ID	Sommario	Priorità
445	[Italy Pilot Sites - SteelMinds] Training: Context History Entreat	top
446	[Italy Pilot Sites - SteelMinds] Training: Reasoner Application	top
447	[Italy Pilot Sites - SteelMinds] Support request for ontology writing	top
452	[Italy Pilot Sites - SteelMinds] Training: Application Ontology and uAALutils library use	top
453	[Italy Pilot Sites - SteelMinds] Training Request for Application Ontology and Service Profiles	top

Figure 1 Tickets opened during coaching

The coaching activities are still in progress, however it is possible to draw two conclusions and lessons learned:

- Developers approaching to a new platform such as universAAL need first an off-line phase during which they can read and test the platform. This phase allows to the developers to become familiar with the universAAL APIs and terminology to be used.
- On line seminars and Skype calls are the most effective tools for answering to questions from companies. Email exchange are helpful only if short and concise.
- The quality of the documentation is the key-factor for the success of the diffusion of a software artefact such as a library or an execution platform.

4.2.9. RNT

Smart Homes (SmH) was RijnmondNet's (RNT) coach for this project. RNT is not a technical party itself and were not able to comprehend the complexity of the platform. This was solved by inviting the supplier Almende to the coaching meetings. Although none of the six suppliers had any experience with universAAL, Almende took the lead in trying to understand universAAL and was able to setup a test environment and getting familiar with universAAL. The other suppliers were:

- MedicineMen
- Netmedical
- Mibida
- MindDistrict
- Curavista

During this period the coaching of Smart Homes consisted on sharing experience during meetings, which we had on regular basis. The feedback from these meetings were reflected through SmH in the technical workshops and technical teleconferences that were organized by WP2 lead.

This resulted for example in better, understandable documentation and a quick-start guide for every new developer who wants to setup a universAAL developing environment. At a later stage, the forum of the platform was used to answer questions directly by the main developers of universAAL, which in most cases are members of WP2.

The corrective actions that were assigned to RNT by the management of the Make it ReAAL project made a more intense appeal to SmH as coach. During this period a strict time-schedule was created and a document with 4 milestones. Due to the strict time limitations RNT and SmH had at least weekly teleconference meetings in order to make sure all milestones were reached in time and good quality. SmH was responsible to setup meetings for discussing the ontologies with all suppliers amongst others. Every week a fixed timeslot was allocated for an open question round between suppliers and developers.

Lessons learned and experience

The forum of the platform is vital for knowledge sharing, however in a lot of cases this was seen as overhead and the path of e-mailing directly from supplier to developer was used as it was quicker and more easy. This resulted, in most cases, that SmH as coach was not involved in heavy discussions over any issue's that occurred.

RNT, SmH and all six suppliers had no experience in or with universAAL. Almende and SmH started to investigate the possibilities of the platform when the project started. The task of SmH was not to implement any applications and thus had the theoretical knowledge and no the hands-on knowledge. Almende quickly gained hands-on knowlegde as they were in fact implementing universAAL. This resulted in the fact that questions asked were highly technical and could only be answered by main universAAL developers. The role of SmH as coach changed gradually from a highly technical knowledgeable role, towards more organizational role, e.g. keeping track of deliverables that needed technical input from the suppliers, etc.

4.2.10. TEA

UPM was the assigned coach for TEA pilot site. Being located in the same city made it easy to follow up on the progress and profit from physical meetings to provide a more effective support.

In the case of TEA, the coaching proceeded without major issues according to the usual steps described in the first section. After the initial phases of adaptation to universAAL, suggestions for improvement were discussed and adopted in the context of the importing application process.

When dealing with developers that have a good understanding of how to model ontologies, it is easier to convey all the semantic functionalities of the platform. It is particularly useful to explain the intricacies of how services may or may not be matched, and how to use ontological modelling to the advantage of the design of the application.

Sometimes concepts have to be particularly agreed, sometimes the good understanding of the trainee leads to misunderstandings. For example, instead of reusing existing concepts, by adding properties, TEA had made a copy to customize of these concepts, effectively duplicating concepts unnecessarily. Thankfully, and yet again, the semantic features of mapping ontological concepts can be used to unite once again the duplicated concepts.

During Training it is critical to convey a broader understanding of the platform. universAAL is very complete and some trainees will pick some key concepts they

feel the platform can be used for, and disregard the rest of the properties of the platform.

For example, the most important lesson in this regard, is to show that platform can be used as a tool for developing within application and not just between applications. This concept of using universAAL as base for the system architecture of the application will also be useful in both cases: a) it will help optimize the design of the application itself by being able to use the features of the platform b) inner links are “exposed” for other applications to use and exploit the first application to different degrees.

Another misconception, which thankfully was identified early on, is about security. Without careful explanation trainees may understand that the busses of the platform operate at universal scale (probably confused by the name of the platform), where privacy and security concerns take them back from counting with the platform features for many application areas (especially those sensitive to privacy issues).

Generally, the most efficient way to coach a team which requires platform knowledge is using continuous communication. Being open to questions and participation on the application design, in most cases, helps trainees to get insight of the process of how to get the most advantage of the platform. It is also important to keep a look on the updates on the development, especially during the ontology implementation, as ontologies are the basis of the whole universAAL platform, their optimization is critical to the success of the universAALization and for the capability of implementing the different showcases correctly.

4.2.11. **Associated Pilots**

In February 2015, four Associated Pilots joined the ReAAL project with the goal of complementing and extending the experience of the member pilots and providing additional users. For them, TRIALOG is acting as their coaching partner, backed up by UPV.

The uAALisation process has been active from March to July for 3 pilots (IMA, SCUPS, NCSR) and until October for EIC-IL.

Two full coaching days for the associated pilots and associated vendors were organized in Eindhoven by end of February 2015, where all the associated pilots participated and therefore were able to start the design process of their universAALised applications. Raw videos with the theoretical approach are here:

<https://cloud.igd.fraunhofer.de/owncloud/public.php?service=files&t=57283396bca246d856c4307d332dce70>

The project **members** coaching them (mainly Trialog and UPV for the uAAL expertise) guided them through a step-by-step process inspired by the previous WP1 and WP3 works. First an overall template architecture document has been proposed. This template is based on the interoperability requirement deliverable D1.2 and the D3.2a. The architecture of each application is described per software components and per distributed nodes when required. Within this architecture, uAAL middleware is inserted where functional decoupling is suggested by the designer as well as a first description of the data involved. After discussing about the reasons of this decoupling, pilots started to work on the ontology. As for other pilots, it has been proved that the choice of the ontologies (reusing an existing one or creating a new one) is one the most important decision to ensure a reasonable adaptation.

4.3. Conclusion

One of the advantages of the universAAL platform is that it can be adapted in many different ways to many different scenarios, but in turn, it makes it difficult to define a holistic compliance methodology. There are, of course, certain “checks” that need to be certified in order to consider an application as being adapted to universAAL (universAALized), and these are reported in this deliverable. But beyond these, there is plenty of room for pilot developers to choose how exactly, and for what purpose, they incorporate universAAL into their solutions and how to benefit from it.

Consequently, there is no unique script that a coach can follow to give support to all pilots, given the differences between them. Experience showed that the most practical approach (and probably the best) was providing on-demand support through direct communication channels. While it is thought that face-to-face meetings (e.g. in plenaries) or workshops are fruitful (and they are), their limited duration and the effort required to attend the meetings make them an exceptional occasion, used for exceptional issues.

As a conclusion, pilots required a dedicated coaching support tailored to their specific architecture and intended usage of the platform. The compliance of their adapted solutions had to be evaluated starting from a common set of minimal requirements and continuing through each specificity of their deployment, to determine that the adaptation was correct and useful.