Contract No. IST 2005-034891

Hydra

Networked Embedded System middleware for Heterogeneous physical devices in a distributed architecture

Validation Report for DDK Prototype

Integrated Project
SO 2.5.3 Embedded systems

Project start date: 1st July 2006 **Duration: 52 months**

Published by the Hydra Consortium **31.08.2009**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level: PUBLIC

Document file: Hydra - D10.3 Validation Report for DDK Prototype

Work package: WP 10 – Validation & business modelling

Task: T10.1 – User validation

Document owner: INN

Document history:

Version	Author(s)	Date	Changes made
0.1	A. Gugliotta, A. Guarise (INN)	04-06-2009	Document organisation, objectives and first structure
0.2	A. Gugliotta (INN), M. Ahlsen (CNet), M. Ingstrup (UAAR), M. Jahn (FIT), T. Wahl (SIT), F. Milagro (TID)	06-07-2009	Included list of requirements for the second cycle of validation with contributions from WP leaders
0.3	A. Gugliotta (INN), M. Ahlsen (CNet), M. Ingstrup (UAAR), M. Jahn (FIT), T. Wahl (SIT), F. Milagro (TID)	07-07-2009	Added text descriptions in sections 1,2 and 3 with contributions from WP leaders
0.4	A. Gugliotta, A. Guarise (INN)	29-07-2009	Added text descriptions in section 4 with contributions from WP leaders
0.5	A. Gugliotta (INN)	05-08-2009	Finalized the deliverable
1.0	A. Gugliotta (INN)	31-08-2009	Addressed the comments of internal reviewers

Internal review history:

Reviewed by	Date	Comments
Trine Fuglkjær Sørensen (IN-JET)	18-08-2009	Approved with comments
Adedayo Adetoye (UR)	26-08-2009	Approved with comments

Index

List of figures

List of tables

1. Introduction

1.1 Purpose and context

This deliverable provides the results of the second iteration validation phase focussed on the DDK prototype. The objective is to show the major outcomes found after applying the validation concepts described in the revised Deliverable 10.1 "Validation Plan for prototypes" in order to better understand the middleware and DDK prototype and be able to feed the lessons learnt back into the next development iteration. The validation tests have been fulfilled during the implementation phase and in a specific testing activity and it involved both the second Hydra prototypes (DDK) but also the middleware which is under continuous development. The critical outcomes and those not planned or foreseen to occur during the software code writing are also highlighted.

It is important to underline at this stage, that the level of implementation gathered for the DDK has not yet reached the release phase. Indeed, the iterative approach followed in Hydra consists of successive improvements of the software package, and the validation fulfilled during this second loop involves the software components that have been developed so far and:

1) the group of requirements that have been considered as a reference and guiding specification for the actual implementation;

2) the group of requirements that have been considered in the first validation phase (reported in D10.2) but resulted in either "partial supported" or "not yet supported".

1.2 Outline

The present validation report represents the second document of a series of three different assessment studies, one per prototype and iteration, organised and structured as explained in the validation plan (D10.1), which is considered as an input document. Therefore, this document follows the same structure introduced in the report of the previous validation phase (D10.2). Section 2 recalls the objects of the validation, the targeted users and the reasoning for explaining the requirements selection. As Hydra foresees to meet more than 450 requirements, it is necessary to limit the number with a careful selection of the most important ones, given the fact that not all of them have already been implemented at this point of the project and that a large part of the technical (functional) specifications are considered to be met at debug level.

Section 3 briefly describes the assessment methodology applied to each requirement and summarises them into groups divided per work package. The tables in this section (selection of requirements) are revised in respect to those that were indicated in the Deliverable D10.1 and D10.2, because (i) a short list has been made considering the most important ones among those so far implemented and (ii) we distinguish between requirements tested in this and the previous validation phase.

Section 4 reports the results obtained while applying the assessment procedures for the evaluation of the fit criteria fulfilment. Each WP leader and the validation participants decided together on how to give proof of the requirement verification (fit criterion) and the threshold level below which the requirement is considered not met. In the worst case (requirement not reaching the threshold) the requirement is marked to be re-evaluated at the next validation iteration, so that all requirements are submitted to a continuous improvement. To conclude the section, a table summarises all the requirements tested so far (first and second cycle) and their current state (supported, not yet supported or partly supported).

Section 5 draws the major conclusions of the report. It gives some figures on the obtained results, indicates the open issues and expected progress in the assessment procedure and how the validation process shall improve the implementation part.

2. Object of the validation

The foreseen planning from the validation plan (considering also the revision made to D10.1) is depicted in Figure 1. The validation started at due date, while second and third steps were adapted as per partners' request.

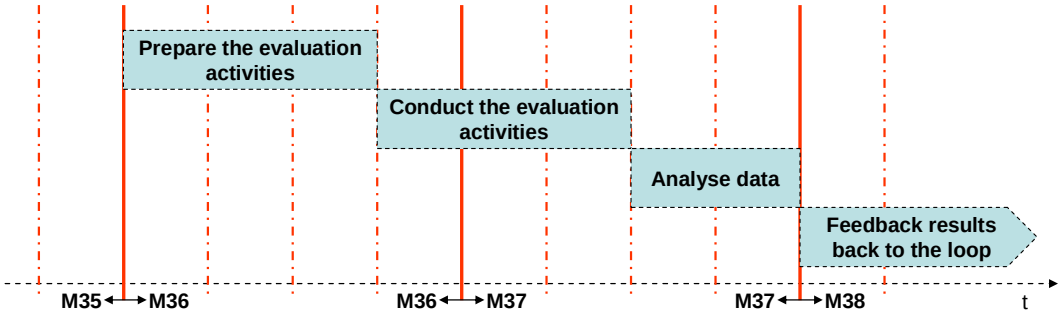

Figure 1: User validation second iteration - time plan

In the second validation cycle, the object of the evaluation is the DDK and middleware prototypes. However, because some requirements, that resulted in being partial supported or not yet supported in the first iteration, need to be validated again, the second iteration also considers the last version of the SDK. In fact, every validation cycle assesses components that are not considered as the final ones, but as the partial release of a subsequent delivery of improved prototypes.

Moreover, the requirements selection, even if based on the initial Deliverable 10.1 list, has also been updated with the intention first of all to fine-tune the group of requirements towards a higher number, and secondly to consider those already implemented so as to make the testing possible. The tables in the next section eventually consider the new important specifications introduced after the completion of the Validation Plan.

2.1 Target users

Hydra identified along the previous deliverables two main groups of users:

- developers that will use the middleware, considered as the major focus for the validation report due to their direct involvement in the SW development process, which is the aim and the reason why Hydra middleware has been conceived;
- end-users that will benefit from the Hydra enabled services created by the previous group, the developers, and also considered as a major source of feedback due to their role in the value chain and their fundamental part in the product´s successful commercialisation.

Therefore, we differentiate between the term developer-user from end-users, as the same difference existing from those who create a product (developers, first group) from the real users of the product itself. The validation plan divided the task activity into three different parts related to each project iteration conclusion and depicted in the next table.

Type of user	Object of the evaluation	Start of the user validation (month)
Developer user	SDK + middleware vers. 1	M24
Developer user	DDK + middleware vers. 2	M36
Developer user	IDE + middleware vers. 3	M48
End user	Applications	M48

Table 1: Validation plan milestones

As the actual object of the assessment is the middleware and the DDK, during this validation cycle the target users are the application developers, from the first group indicated above.

The developer users are identified among Hydra internal resources where possible. This is done mainly because it is difficult to find the commitment from companies not directly involved in the Hydra consortium, especially from an economical point of view (external experts who are not Hydra partners asking for a fee shall be paid with the means of subcontracting). This is also a challenge because we must consider that evaluation with developer-users may or may not lead to new issues if compared to traditional user validation. In order to diminish this risk the selected developers were chosen from among those who were not directly involved in the Hydra implementation, otherwise their judgement would be biased.

2.2 Quality dimensions and assessment criteria

Similarly to the previous validation cycle, the validation is made through the comparison between an expected impact (requirement) and how the real application works. In Hydra the expected impact is described with the means of the user requirements, derived in WP2 and collected throughout all WPs. The user requirements consist of a list of features and properties of the Hydra middleware including quality criteria, which are considered relevant by the users. Deliverable 3.2 "Updated system requirements report" contains an updated overview of the requirements that shall be necessary to the Hydra developed system as emerged in several focus groups with developer users.

Every requirement statement is composed of six fields to briefly describe it, as shown in the next example.

ID:	31
Type:	Non-functional / look and feel
Priority:	Critical
(Short) description:	An easy-to-use programming framework should be provided
Rationale:	The programming framework provided by the prototype should be easy to use in the sense that it is intuitive
Fit Criteria:	9 out of 10 developers recognise the IDE as intuitive

As quality is a relative or personal issue to be measured, a value must be attached to the cost and benefit of quality-oriented actions. Features and properties requested by stakeholders have to determine how to implement and what the optimal investment is.

There are different frameworks analysing quality attributes, with differing vocabulary, metrics etc. that are relevant to software architecture design. Quality attributes are essential to the design of software architecture, but it is a challenge to describe quality attribute (requirements) on a common form. For this reason, together with the Volere schema for drafting user requirements, the SEI quality framework (Bass et al., 2003) and the ISO 9126 (2001) international standard have been studied. The SEI quality framework, also known as Quality Attribute Scenarios, is a well-established way of defining architectural requirements in a uniform way and introduces the concept of considering quality attribute requirements on a fixed and precise scenario form. This approach has been integrated in the context of the Hydra project with the ISO 9126 international standard defining a comprehensive quality model for software products. Deliverable 6.1 "Quality Attribute Scenarios" gives a detailed and clear overview of the two frameworks.

The second validation report follows the same schema defined in the previous report on how to measure the fit criteria pertaining to each different requirement. In particular, the assessment procedures summarised in Section 3 tables and then applied in Section 4 have been identified in D10.2 "Validation Report for SDK Prototype" to be used in the following validation cycles and, thus, simplify the evaluation effort and for improving also the single requirement evaluation, in case some of them were not satisfying the threshold condition.

2.3 Requirements for the second iteration

Developer-users are interested in requirements fulfilment, the technical aspects related to the software instrument they want to use: a middleware, DDK, or another prototype. For this deliverable the validation is applied through requirements technical tests and assessment fulfilled at the end of the DDK cycle implementation.

The first group of requirements was identified in Deliverable 10.1, as the total number of specifications had reached a large quantity. In the Validation Plan all major functional and non-functional requirements were chosen, but the overall tables have been revised or updated during project activity and in this report. As a major observation, the largest part of functional requirements were considered to be verified during the debugging phase, otherwise the middleware component would not work, so just the most important among them were taken into account for the validation process. The specifications have been confirmed depending on their implementation status at the time of the validation, and eventually substituted with those that have been already considered at this stage of the project.

The final selection of requirements was performed by each work package leader in agreement with the WP participants. Starting from the initial group, each WP first confirmed the possibility to assess or not each requirement and then identified the major ones on which to apply the testing procedure, eventually integrating or substituting the initial list in case new requirements were added, old important ones had been left out or the previous selected group was not adequate or sufficient. The need to have a short list of final requirements was due to the large number of entries so far identified during the project course (more than 450) as the validation shall be completed into a defined time frame (i.e. 2 months) for allowing the provision of the results back into the loop.

The requirement refinement is strongly related to two factors: the software development process, which requires different needs for different components, and the iterative approach, which adds the latest requirements at every implementation update.

The final list of the requirements selected for the second iteration is presented in the next section, through tables divided depending on the particular WP. Differently to the previous report about the first validation cycle (D10.2), in this validation report we define two tables for each WP:

- The first one collects all requirements that were not supported or partially supported after the first validation cycle; these requirements have been tested again on the current middleware and SDK, as well as tested for the first time (if applicable) on the DDK.
- The second one collects the requirements that have not been selected in the previous validation cycle and need to be tested on the current middleware and the DDK.

3. Description of the validation methods

Once the validation testing procedures are defined, the tester has to follow the indications given to perform the validation, which can be a laboratory test or a trial of the middleware/DDK. Different expert evaluators do not find the same defects, and not in the same order. It is therefore advisable to use at least two or three experts (even more if available).

The developer user can be assisted by colleagues actively involved in the Hydra project in case something is not clear or misleading. The process of the validation by the software developer should be linear if the planning is done carefully and the validation procedures are prepared with sufficient details.

Experience shows that the more immature an implementation is, the faster defects will be found. Users who are confronted with incomplete and faulty software become frustrated and cannot provide much constructive feedback. So it is preferable to proceed with the first middleware evaluation at an advanced stage, when the implementation of software has already reached certain robustness. As the prototypes are recursively improved, the middleware assessment is repeated in all iterations. The collected feedback allows having a constant improvement of the implemented system.

First there will be a collection of data as a result of laboratory test by considering each requirement referring to the middleware. This will be the case for those quality dimensions that need a specific measurement (for example, an efficiency performance test). On the other hand requirements that need a special evaluation, not feasible with a simple measurement, will be assessed through a complete description of the reasoning developer users.

The DDK assessment is performed in the same way as it is done for the middleware, but differentiating the domain applicability. The assessment used laboratory measurements, software procedures and an assessment analysis completed by the developer users who exploited the Hydra components.

Assessment procedure for verifying the fit criteria fulfilment

The assessment procedures for the requirement evaluation were deployed by the WP leader in agreement with other WP partners. The testing has been decided in order to assure that the methodology is able to verify that the fit condition is met with limited uncertainties. In case of functional requirements usually this is proved by the means of a (numerical) threshold level; in case of a non functional requirement where there is no clear indication of the expected result, the assessment procedure contains the background methodology and the proper conditions able to demonstrate the criterion verification.

As an example, requirement n. 31 mentioned above has already a fit criterion identifying the numerical indication for which the requirement is considered as met.

ID:	31
Type:	Non-functional / look and feel
Priority:	Critical
(Short) description:	An easy-to-use programming framework should be provided
Rationale:	The programming framework provided by the prototype should be easy to use in the sense that it is intuitive
Fit Criteria:	**9 out of 10** developers recognise the IDE as intuitive

3.1 WP3 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome 1st Cycle		Outcome 2nd Cycle		
					Middle ware	SDK	Middle ware	SDK	DDK
31	An easy-to-use programming framework should be provided	The programming framework provided by the SDK should be easy to use in the sense that it is intuitive.	9 out of 10 developers recognise the SDK as intuitive.	Conduction of a software-walkthrough and a validation session with developers specifically addressing the ease of use.	n.a.	Not yet supported	n.a.	Partly supported	n.a.
41	Hydra Developer's Companion	Complete and comprehensible documentation is very important to the Hydra software developer.	Complete documentation is available. It is considered "very helpful" by at least 8 out of 10 developers.	Conduction of a technical review of the documentation. Run a software walkthrough as a preparation for the training activities.	n.a.	Partly supported	n.a.	Partly supported	Partly supported
136	Dynamic architecture	The architecture of a running Hydra system can be easily modified by increasing or decreasing the degree of centralisation in order to balance the utilisation of available resources.	In 95% of all cases, Hydra supports dynamic migration of components to realise centralised and decentralised systems.	Implement and run a test application and test whether it can be reconfigured or not.	Not yet supported	n.a.	Partly supported	n.a.	n.a.
185	Middleware provides basic services	In order to program AmI applications, the middleware must provide basic services. This makes life easier for application developers. Basic services provide e.g. methods to query available devices and services or to pass messages between components.	Middleware provides a set of basic services that at least contain basic functionality that is needed by all services, like communication and a service / device registry.	Conduct a technical review of the core Hydra services with developers. This review aims to define the setup of a basic Hydra infrastructure, querying available devices and passing messages between devices.	Partly supported	n.a.	Supported	n.a	Supported
199	Modules should be extendable	Hydra modules should be extendable in their functionality by 3rd-party solutions.	80% of all Hydra modules are extendable in their functionality by integrating 3rd-party code via a standard	An assessment procedure that measures the extensibility of software is part of current research. One	Partly supported	Partly supported	Partly supported	Partly supported	Partly supported

			interface or replaceable by 3rd-party modules with equivalent functionality.	approach could be to count the number of hooks that allow for the modification of existing modules or the addition of new ones. Another approach could be to let a number of developers implement extensions to the Hydra middleware and to assess the result. Thus, a formal assessment procedure remains an open issue.					
207	Service selection by context	In order to select an appropriate service for a specific task, contextual information, like the spatial position, must be taken into account. Hydra must provide a method to specify a desired service by contextual parameters. For example, if a certain room in a building is specified in a search request for a service, only services that are relevant in the current user's location and context are returned.	In search requests for a specific service, contextual information like a spatial position is allowed.	Build a prototype, which combines location and other context constraints to select an appropriate service. An example scenario would be: A user wishes to print a coloured document to the nearest printer during a presentation.	Partly supported	n.a.	Partly supported	n.a	n.a
217	The middleware should ensure high robustness of services	In order to ensure the service support of important components in the system, the middleware should provide a highly robust service structure.	Breakdown of crucial services of the middleware in less than 1 case per 100 hours of operation.	Identify the crucial services of the Hydra middleware, build a test application that is based on that set of services and conduct a long-term operation stress test.	Partly supported	n.a.	Partly supported	n.a.	n.a.
234	The middleware should be self descriptive	The developer should be enabled to understand all components and their interplay of the system in order to take full advantage of the Hydra	Nine out of ten developers have a clear understanding of the Hydra middleware after one week of experience.	Conduct a software peer review with developers.	Not yet supported	Not yet supported	Not yet supported	Not yet supported	Not yet supported

		Middleware.							
320	Separate domain-oriented services and user interface services architecturally	This is a standard architectural design tactic to enhance modifiability.	90% of the modules of the architecture properly separate layers for domain services and interfaces.	Analyse the SVN repository which contains all Hydra managers and modules and identify those that mesh interface and control logic.	Not yet supported	n.a.	Not yet supported	n.a.	n.a
335	Location awareness / positioning support	Hydra should enable developers to write applications that depend on context, especially spatial context.	A component for acquiring spatial context exists. At any time, min. 75% of all devices attached to a Hydra system can be spatially located. Also, there is a programming model for using spatial context.	Build a location-aware application based on the Hydra middleware.	Partly supported	n.a.	Partly supported	n.a.	n.a.

Table 2: WP3 requirements resulted not/partly supported in the 1st cycle.

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome	
					Middle ware	DDK
515	Support of domain-specific ontologies	To establish knowledge or application domain interoperability, HYDRA should be able to support domain-specific ontologies on a structural level. Interoperability can only be established to the degree external ontology support exists.	HYDRA is able to support domain-specific ontologies or not.	Build a prototype that makes use of domain-specific ontologies.	Partly supported	n.a.
518	No external standards should dictate the virtual layer.	Hydra manages internal standards in the virtual layer. These cannot be dictated by external standards.	External standards do not create limitations for HYDRA internal. All access to the virtual layer is done through HYDRA middleware.	Set up a prototype to verify that Hydra middleware handles all access to the virtual layer itself.	Supported	n.a.
519	It should be possible to implement managers in either programming model.	The architecture should be fairly independent of any specific programming model. It should be possible to implement managers in either programming model.	It is possible to implement managers in either programming model or not.	Build a Hydra application that plugs together managers of different programming languages.	Supported	Supported
522	All HYDRA entities	If interoperability and	A hydra-enabled entity must	Build a prototype to ensure that a Hydra-enabled	Supported	Supported

	must have a semantic model description	security are to be possible, an entity must have a semantic model description. Otherwise other devices are not able to discover if they can communicate with the device or if the device security can be resolved according to the security policy. Devices or applications that are unable to present a semantic model description cannot be expected to be able to pass a security resolution according to security policies.	have a semantic model description.	device has a semantic model description.		
524	Determination and Description of the dependencies among Hydra Managers.	Some core managers exhibit a type of predefined collaboration between them; others offer their functionality to all components of the entire Hydra software architecture. Managers of the first group actually demand direct inter-manager calls or a refactoring of the software architecture focussing on the fusion of functionality. Managers of this second group provide functionality to all managers of the other groups. Therefore, the managers of the second group offer functionality that runs orthogonally with respect to the basis functionality. In addition, this orthogonal functionality cannot be separated from the existing components.	The dependencies of all Hydra Managers must be determined clearly and described in detail.	Examine relevant documentation and make sure all Hydra managers and dependencies are covered.	Supported	n.a.
525	Delimitation between Application and Device Elements.	In the first two cycles we found that we need clarification on the delimitation between	No interdependencies between Application and Device Elements.	Make sure the Hydra architecture specification clarifies this aspect and that it is applied to the managers.	Supported	n.a.

		application and device elements. The delimitation between Application and Device Elements seems to blur.				
526	Delineation between middleware and application in terms of context provision.	The Context Manager is mainly connected to the application itself and not to the middleware (as agreed in discussion with the partners); it was withdrawn from the scope of the ontology manager.	In terms of context provision the middleware and the application itself must be delineated.	Check whether context- and ontology managers are clearly delimitated.	Supported	n.a.
528	Specification of the information flow among Hydra Managers.	During software integration of the first year prototype some problems were attributed to the event management, which has been overly used. The application of JAX-WS and Axis for event-driven application worked fine, although some latency has been identified due to multiple concurrent function calls. In addition, the use of web applications as Event Manager in the role of both publisher and consumer works fine. However, the development of web applications for small devices such as PDAs, limits the usage of HTML, JavaScript and CSS.	Complete specification that clearly defines how the information shall flow among Hydra Managers.	Check the relevant documentation.	Supported	n.a.

Table 3: WP3 selected requirements for the 2nd validation cycle.

3.2 WP4 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome 1st Cycle		Outcome 2nd Cycle		
					Middle ware	SDK	Middle ware	SDK	DDK
312	Support profiling of devices' performance	The middleware should contain services that allow monitoring and reaction on what devices are doing. This includes monitoring response time, device load (e.g. CPU), and message interchanges per second.	Said services available in Hydra.	See § 4.2	Partly supported	n.a.	Partly supported	Partly supported	Partly supported
314	Faults should be intercepted by middleware, notified to interested services	To create reliable and available systems it is essential to catch faults/partial failures before they become failures/complete failures. There needs to be uniformity in how this is done; thus it should be supported by the middleware.	The middleware has support (through components/services) for sending and receiving notifications for partial failures.	Experiment with behaviour when services become available, tested with agriculture scenarios, weather station scenarios.	Partly supported	n.a.	supported	supported	TBD
317	Support runtime reconfiguration	To support monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime.	Services and devices can be connected in new ways during runtime in Hydra-based applications.	Test an example application's ability to be reconfigured according to specific scenarios, tested with configurations of Hydra middleware according to QoS requirements.	Not yet supported	Not yet supported	supported	supported	TBD
334	There should be support for developing auto-configuration of certain devices	A number of use scenarios calls for the ability to bring a device home, turn it on, and have it function reasonably.	The middleware supports defining auto-configuration properties and using these at runtime. This is not in conflict with security.	Test execution of configuration script for network manager on osgi.	Not yet supported	Not yet supported	Supported	Supported	supported
366	Web services should run on embedded devices	Service-orientation is a good match for many embedded devices. Web services will provide a	Hydra supports web services on embedded devices (Initial target should	See § 4.2	Not yet supported	n.a.	supported	supported	supported

		gateway to many applications and it would be beneficial to be able to structure all of the communication in a system using the same primitives.	be Develco's DevCom 02 ZigBee module)						

Table 4: WP4 requirements resulted not/partly supported in the 1st cycle.

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome	
					Middle ware	DDK
479	The EventManager should support event prioritisation	The EventManager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system.	Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort.	See § 4.2	supported	supported

Table 5: WP4 selected requirements for the 2nd validation cycle.

3.3 WP5 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome 1st Cycle		Outcome 2nd Cycle		
					Middle ware	SDK	Middle ware	SDK	DDK
276	New communication technologies	New communication technologies might be added to the system, so that Hydra should provide the means to facilitate this inclusion.	80% of new technologies are supported.	Integration of the ZigBee protocol and discovery mechanism.	Supported. Although percentage yet to be validated.	n.a.	Supported. Although percentage yet to be validated	n.a	n.a
407	Storage Manager – Gateways information stored synchronization	The information stored in the Gateway must be synchronized with the information inside the devices. The dumping of devices information could be either initiated by the device or controlled by the Gateway.	90% of the information stored in the Gateway is synchronized with the information stored inside the devices.	Data will be annotated with timing information, which will be used to evaluate applied (soft) real-time constraints.	Not yet supported	n.a.	Not yet supported	n.a	n.a
465	Networks overlapping	If two users of the Hydra system wear a personal Hydra Body Area Network (HBAN) and meet each other in the same place, the HBAN of one user doesn't have to add the devices of the HBAN of the other user. The middleware must provide criteria to distinguish when a "new" device is authorized to be added to an existing Hydra network and when it belongs to another Hydra network which is temporary near to the former device.	A device is not to be added to an existing Hydra network if it is unauthorised or when it belongs to another Hydra network, which is temporarily near to the former device.	Validation session with developers.	Not yet supported. Security not in place.	n.a.	Not yet supported. Security not in place.	n.a.	n. a.

Table 6: WP5 requirements resulted not/partly supported in the 1st cycle.

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome	
					Middle ware	DDK
506	It should be possible to lock files (Storage Manager)	For many reasons it can be important to know that an application is updating data, so that other applications will wait using it until the update is done. There should be a read/write locking.	All write access is aborted if a file is locked.	A Lock Manager will be implemented that provides the ability to get and release locks on entities like files and directories. Validation will be done by trying to sequentalize multiple access.	Not yet supported	n.a.
505	It should be possible to access data in Storage Manager using a well defined protocol (e. g. WebDav)	Using external Applications it should also be possible to access data without to much trouble. Exporting storage using WebDav gives the User the ability to access it as network devices on most operating systems.	50% of the storage can be accessed using WebDav.	The File System Device will be mountable using Fuse. This will be tested by mounting a File System Device on a Linux host and accessing it using desktop applications.	Supported. Storage can be mounted under Linux using FUSE	n.a.
504	It should be possible to add and remove physical stoage from a Mirror/Striping-Set	If there is some striped storage and it is not big enough, it should be possible to increase its size by adding new physical storage.	All striped devices can be enlarged by adding new physical storage.	Adding 10 devices to a striped and a mirrored storage and removing them.	Not yet supported.	n.a.
503	It should be possible to combine different storage for mirroring or striping	To get better storage we need to implement some RAID-Technologies inside Hydra to mirror data over different Storage Manager or to stripe data.	10% of the storage are striped or mirrored.	Building a striped storage on top of two mirrored ones and a mirrored one on top of two striped ones.	Partly supported (Only replicated device is implemented now)	n.a.
502	It should be possible to store simple key/value pairs	Not every Application storing data like sensor data want to use the full overhead of a file system and files. The idea behind this issue is to store something like cookies in a browser.	Storing and receiving cookies to a given Manager does not need more than 3 requests.	Building a test application not sending more than 3 requests per access.	Not yet supported.	n.a.
488	Modular and standard device integration	In order to simplify and speed up the integration of new wireless devices in Hydra, the discovery and proxy creation process has to be standardized and be	30% of proxy modules rely on common kernels.	Limbo tests on modularization.	n. a.	Supported

		as modular as possible, so common parts can be reused by proxies for different wireless devices.				
487	Improve handshake protocol between Network Managers for exchanging certificates	Current protocol is quite low level, just sending certificates to other partners, we should use s.th. like SSL protocol mechanisms, we also have to consider the other trust models, like Web of Trust and user interaction.	In 95% of cases simple protocol would work.	Use different managers, with different keys, and different protocol standards to exchange data between them. Turn on handshake protocol and detect errors either in encryption or keystore methods.	Supported.	
486	Hydra proprietary supernodes are needed to support D2D communication between networks	At the moment, public supernodes are used to act as relays in D2D communication. If these supernodes are down, communication between networks is impossible. Thus, we need to manage our own supernodes in partners' servers.	80% of the time, own supernodes are up and running.	Supernode deployment on CNET and FIT.	Supported.	
446	Security parameters negotiation	Since different applications/ devices request different security parameters, it is not advisable to use fixed parameters for communication but flexible ones.	In 90% of all cases the parameters should be flexible.	Make use of handshake protocol and the security related mangers also with the security ontology to test these negotiation mechanisms. Rules set to different levels of security. Test minimum and maximum levels.	Supported.	
442	Proxy – Gateways can filter and react to data received from associated non-hydra devices	Part of the proxy functionality may include support for filtering of the received data and possibly a reaction to high or low values. Non-hydra devices can not be expected to analyze the data themselves, so the gateways could take care of this.	50 % of Gateways support filtering and reaction to received data.	Generated device proxy using Limbo and using context manager on the gateway to test the data acquisition.	n. a.	Supported
427	D2D communication – Group management	The D2D communication system has to allow the Hydra enabled device to create, join and leave groups of Hydra enabled devices, so the components	90% of the devices involved in the D2D communication system can create, join and leave groups.	Test group creation for applications.	Not yet supported	n.a.

		of these groups share the same credentials and can communicate isolated from non-group-members.				
396	Hydra-enabled devices – May be mobile or fixed equipment	A subset of the Hydra middleware (mainly Network Manager) can be deployed in mobile (PDA, Smartphone) and in resource constraint devices (Home Gateway).	30% of State of the Art PDAs, Smartphones and Home Gateways can host part of the Hydra middleware.	Test the implementation on Home Gateway (Play Station 3) and Android mobile phone.	Supported	n.a.

Table 7: WP5 selected requirements for the 2nd validation cycle.

3.4 WP6 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome 1st Cycle		Outcome 2nd Cycle		
					Middle ware	SDK	Middle ware	SDK	DDK
91	Any Hydra device should have an associated description	For management, search and discovery purposes, all Hydra enabled devices should be described (classified) according to the Hydra device ontology.	Any device associated to a Hydra application is also included in the Hydra device ontology, and its description can be retrieved.	Check that a newly discovered device has/gets a corresponding representation in the Device Ontology.	Not yet supported.	Not yet supported.	Supported	Supported	Supported
110	Device Categorisation in runtime	Middleware should after discovery of device be able to categorise a device based on device ontology information.	7 of 10 devices are correctly categorised and described.	Enter new devices into a Hydra network, locally and remotely.	Partly supported. Devices executable in application	Partly supported. Devices executable in application	Supported	Supported	Supported
114	Semantic enabling of device web services	Middleware should be able to attach semantic descriptions to device web services based on device ontology.	7 of 10 devices are semantically enabled.	Enter new devices into a Hydra network, locally and remotely.	Not yet supported. Requires manual interventio n.	Not yet supported. Requires manual interventio n.	Partly	Partly	Supported
122	Configurable and easy to install middleware	The middleware should be configurable and easy to install/deploy.	The average installation time is less than 1 hour.	Time a middleware installation.	Not yet supported. Installation still manual.	Not yet supported. Installation still manual.	Not yet supported. Installation still manual.	Not yet supported. Installation still manual.	Not yet supported. Installation still manual.
376	Security requirements must be part of the Hydra MDA	Security must be defined to be resolved semantically.	Security model can be defined semantically.	A semantic security model exists, check resolution process.	Not yet supported. The resolution process is not in place.	Not yet supported. The resolution process is not in place.	Supported	Supported	Supported

Table 8: WP6 requirements resulted not/partly supported in the 1st cycle.

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome	
					Middle ware	DDK
501	A Hydra enabled device must support UPnP discovery	UPnP has been proven as a well-functioning network discovery mechanism in HYDRA.	All HYDRA enabled devices support UPnP.	All Hydra devices are found thru UPnP.	supported	supported
500	Semantic annotations of devices using SAWSDL	Device developers should via the DDK be able to produce (SAWSDL) annotations for devices, in order to facilitate device discovery and ontology update.	For a given UPnP discoverable device, it is possible to create an SAWSDL annotation which can be accessed from the UPnP discovery information.	Annotations can be attached and retrieved for any device.	supported	supported
477	Device proxies should make use of available security features for "last mile" communication	If non-Hydra-enabled devices are communicated to the Hydra network by a proxy, security features of the protocol supported by the device should be used.	Device proxies must support WEP and WPA for WiFi-connections as well as Bluetooth authentication and encryption.	Device proxies can be created that use WEP and WPA for WiFi-connections as well as Bluetooth authentication and encryption.	supported	supported
126	Automatic Device ontology updates	The device ontology should automatically update its device descriptions.	The device ontology can detect device updates and handle that in 7 of 10 cases.	Enter new devices into a Hydra network, locally and remotely, device discovery results in an ontology update.	N/A	Partly supported
122	Configurable and easy to install middleware	The middleware should be configurable and easy to install/deploy.	The average installation time is less than 1 hour.	Time a middleware installation.	Not yet supported. Installation still manual.	Not yet supported. Installation still manual.
120	Multiple Device Virtualisations	It should be possible to have several different views/virtualisations of a device depending on context and applications.	Multiple Device Virtualisations.	A developer is able to create at least two different views onto the same physical device.	supported	supported
117	HYDRA component ontology	In order to support automatic device proxy creation, a HYDRA middleware manager ontology is needed. The ontology will facilitate the selection of the appropriate device and service managers to implement the proxy, depending on the discovery protocol and	HYDRA device and service managers can be identified and selected through a software component ontology.	HYDRA device and service managers can be identified and materialized/displayed.	Partly	Partly

		device types.				
114	Semantic enabling of device web services	Middleware should be able to attach semantic descriptions to device web services based on device ontology.	7 of 10 devices are semantically enabled.	Enter new devices into a Hydra network, locally and remotely.	Partly	Supported
113	Composition (of services and devices)	In order to enhance or replace application level functions it will be useful to be able to compose services and devices from different providers and/or manufacturers into high level services/devices.	Service composition during design-time is possible.	Design an application composed of at least two different devices of different type and with different services.	supported	supported
112	Dynamic Web Service Generation	Configuration tool that is able to generate the necessary interfaces to wrap the device functionality as a web service.	7 of 10 device functionalities are automatically represented as web services.	Enter new devices into a Hydra network, locally and remotely.	supported	supported
104	Automatic Discovery of Services	It should be possible to configure the middleware to discover available services that meets defined criteria.	8 of 10 services are automatically discovered.	Enter new devices into a Hydra network, locally and remotely.	Partly supported	Partly supported

Table 9: WP6 selected requirements for the 2nd validation cycle.

3.5 WP7 – Evaluated requirements

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome 1st Cycle		Outcome 2nd Cycle		
					Middle ware	SDK	Middle ware	SDK	DDK
308	The Security Level of an existing network should be determinable	For a device entering an existing network it can be useful to determine the security level of that network. Depending on the provided security level the device can decide to enter the network or not.	Hydra middleware provides at least one mechanism enabling devices to determine the security level of an existing network.	Evaluation of the current status of the middleware security architecture.	Not yet supported	n.a.	Supported (best effort)	n.a.	n.a.
468	Different levels of security must be supported	In the healthcare scenario there are 2 communication types: - the inter-BAN communication - the internet communication Each of them could implement a different security criterion. The middleware could support different security levels during communications with wireless devices. For example, a simple accounting procedure for devices near to the user (a BAN in the healthcare scenario) and a harder codification for long distance communications where identity data are transmitted are supported.	It must always be possible to implement at least two different security levels for an application.	Evaluation of the current status of the middleware security architecture.	Not yet supported	n.a.	Ambiguous, no further assessment	n.a.	n.a.

Table 10: WP7 requirements resulted not/partly supported in the 1st cycle.

ID	Description	Rationale	Fit Criteria	Assessment procedure	Outcome	
					Middle ware	DDK
364	Hydra's Access-Control policies support credential based authentication	Instead of identifying the user or device, a session may be authenticated through credentials recognised by the application such as blinded certificates, direct anonymous attestation, previously agreed tickets, reuse of previous accepted keys (e.g., PGP keys). That means the network can operate with authentication schemes using credentials without having to identify the device and/or user. The point is that identification of people or devices MUST NOT be MANDATORY. Alternative mechanisms such as credential based authentication MUST be ALLOWED.	Access-control can be based on credentials.	Evaluation of the Access Control Policy Framework.	supported	supported
498	Mechanisms used for communication security should be replaceable by configuration	Cryptographic algorithms, protocols and authentication mechanism might become insecure after a Hydra-based application has been deployed. In that case, it should be possible to exchange security modules without having to recompile/deploy the middleware.	For at least two of the communication protection mechanisms (Core / Inside / Outside Hydra) it should be possible to replace security modules without recompiling the middleware. Evaluation of the current status of the middleware security architecture.	Evaluation of the current status of the middleware security architecture.	supported	supported
509	Enforcement of Access-control policies	Access control decisions must be enforced.	Policy enforcement points can be attached to Hydra web services so that access control decisions can be enforced.	Evaluation of the Access Control Policy Framework.	supported	supported
510	Enforcement of obligation policies	Security obligation policies dictate certain actions that have to be taken upon occurrence of an event trigger. Components that are part of a policy domain must negotiate on the action they can enforce and must provide the respective enforcement mechanism.	Hydra components negotiate their capability to enforce different actions with the policy decision point and provide an enforcement mechanism for at least one action type.	Evaluation of the current status of the middleware security architecture.	supported	supported

Table 11: WP7 selected requirements for the 2nd validation cycle.

4. Validation results

This section contains the description of the applied assessment procedures and outcomes, highlighting the major findings emerged during the validation fulfilment. The results are divided depicting the analysis carried on for each single requirement evaluated and grouped depending on their relative work package.

4.1 WP3 validation results

Requirements from the previous cycle

Req. ID: **31**

Description: An easy-to-use programming framework should be provided.

Fit criteria: 9 out of 10 developers recognise the SDK as intuitive.

Assessment procedure: Conduction of a software-walkthrough and validation sessions with developers specifically addressing the ease of use.

Description of the assessment result: Partly supported. The SDK is still under development.

Req. ID: **41**

Description: Hydra Developer's Companion.

Fit criteria: Complete documentation is available. It is considered "very helpful" by at least 8 out of 10 developers.

Assessment procedure: Conduction of a technical review of the documentation. Run a software walkthrough as a preparation for the training activities.

Description of the assessment result: Partly supported. Deliverables covering developer-centric issues exist, describing SDK as well as DDK. Yet, an exhaustive developer's companion is not available.

Req. ID: **136**

Description: Dynamic architecture.

Fit criteria: In 95% of all cases, Hydra supports dynamic migration of components to realise centralised and decentralised systems.

Assessment procedure: Implement and run a test application and test whether it can be reconfigured or not.

Description of the assessment result: Partly supported. This requirement mainly depends on requirement 317 (WP4). From the architectural point of view, this can be seen as supported. A Hydra application can be run inside an OSGi container, to allow dynamic reconfiguration of components. This has been tested in the eHealth demonstrator.

Req. ID: **185**

Description: Middleware provides basic services.

Fit criteria: Middleware provides a set of basic services that at least contain basic functionality that is needed by all services, like communication and a service / device registry.

Assessment procedure: Conduct a technical review of the core Hydra services with developers. This review aims at the setup of a basic Hydra infrastructure, querying available devices and passing messages between devices.

Description of the assessment result: Supported. The core Hydra services exist and are consolidate: the services and devices can be registered at the Hydra network. Experiments and demonstrators show that these core elements work together properly.

Req. ID: **199**

Description: Modules should be extendable.

Fit criteria: 80% of all Hydra modules are extendable in their functionality by integrating 3rd-party code via a standard interface or replaceable by 3rd-party modules with equivalent functionality.

Assessment procedure: An assessment procedure that measures the extensibility of software is part of current research. One approach could be to count the number of hooks that allow for the modification of existing modules or the addition of new ones. Another approach could be to let a number of developers implement extensions to the Hydra middleware and to assess the result. Thus, a formal assessment procedure remains an open issue.

Description of the assessment result: Partly supported. The Hydra SDK will be published under an open source licence, which guarantees at least maximum modifiability. Furthermore, the software architecture follows several design patterns (see deliverable D3.9) that aim at a good extensibility. Also DDK components are extendable. However, a formal assessment procedure could not be conducted so far.

Req. ID: **207**

Description: Service selection by context.

Fit criteria: In search requests for a specific service, contextual information like a spatial position is allowed.

Assessment procedure: Build a prototype, which combines location and other context constraint to select an appropriate service. An example scenario would be: A user wishes to print a coloured document to the nearest printer during a presentation.

Description of the assessment result: Partly supported. Respective components are under development but have yet to be applied in demonstrators.

Req. ID: **217**

Description: The middleware should ensure high robustness of services.

Fit criteria: Breakdown of crucial services of the middleware in less than 1 case per 100 hours of operation.

Assessment procedure: Identify the crucial services of the Hydra middleware, build a test application that is based on that set of services and conduct a long term operation stress test.

Description of the assessment result: Partly supported. As well as the building automation demonstrator, the eHealth demonstrator proved to be a robust Hydra application. Nevertheless, formal stress tests still need to be conducted.

Req. ID: **234**

Description: The middleware should be self descriptive.

Fit criteria: Nine out of ten developers have a clear understanding of the Hydra middleware after one week of experience.

Assessment procedure: Conduct a software peer review with developers.

Description of the assessment result: This requirement has not been fulfilled, yet. Such a software peer review will be scheduled after the developments on the Hydra middleware are finished.

Req. ID: **320**

Description: Separate domain-oriented services and user interface services architecturally.

Fit criteria: 90% of the modules of the architecture properly separate layers for domain services and interfaces.

Assessment procedure: Analyse the SVN repository which contains all Hydra managers and modules and identify those that mesh interface and control logic.

Description of the assessment result: No assessment results so far, since the set of Hydra managers is not complete yet.

Req. ID: **335**

Description: Location awareness / positioning support.

Fit criteria: A component for acquiring spatial context exists. At any time, min. 75% of all devices attached to a Hydra system can be spatially located. Also, there is a programming model for using spatial context.

Assessment procedure: Build a location-aware application based on the Hydra middleware.

Description of the assessment result: Partly supported. Please refer to requirement n. 207.

Requirements for this cycle

Req. ID: **515**

Description: Support of domain-specific ontologies.

Fit criteria: Hydra is able to support domain-specific ontologies or not.

Assessment procedure: Build a prototype that makes use of domain-specific ontologies.

Description of the assessment result: Partly supported. The Ontology Manager provides support for domain specific ontologies. Yet this has to be tested in a prototype.

Req. ID: **518**

Description: No external standards should dictate the virtual layer.

Fit criteria: External standards do not create limitations for Hydra internal. All access to the virtual layer is done through Hydra middleware.

Assessment procedure: Set up a prototype to verify that Hydra middleware handles all access to the virtual layer itself.

Description of the assessment result: Supported. All access to the virtual layer is enabled via the Hydra concept of HIDs. This is applied in the eHealth demonstrator.

Req. ID: **519**

Description: It should be possible to implement managers in either programming model.

Fit criteria: It is possible to implement managers in either programming model or not.

Assessment procedure: Build a Hydra application that plugs together managers of different programming languages.

Description of the assessment result: Supported. Hydra provides managers that are implemented in different languages. The eHealth demonstrator shows that these managers work together smoothly.

Req. ID: **522**

Description: All Hydra entities must have a semantic model description.

Fit criteria: A Hydra-enabled entity must have a semantic model description.

Assessment procedure: Build a prototype to ensure that a Hydra-enabled device must have a semantic model description.

Description of the assessment result: Supported. In order for a device to be discoverable it must have a semantic model description. This concept is applied in the eHealth demonstrator.

Req. ID: **524**

Description: Determination and Description of the dependencies among Hydra managers.

Fit criteria: The dependencies of all Hydra managers must be determined and clearly described in detail.

Assessment procedure: Examine relevant documentation and make sure all Hydra managers and dependencies are covered.

Description of the assessment result: Supported. D3.9 Updated System Architecture Specification provides comprehensive descriptions on the dependences among managers.

Req. ID: **525**

Description: Delimitation between Application and Device elements.

Fit criteria: No interdependencies between Application and Device elements.

Assessment procedure: Make sure the Hydra architecture specification clarifies this aspect and that it is applied to the managers.

Description of the assessment result: Supported. The updated system's architecture specification describes how managers are positioned regarding application and device elements. Since Hydra supports a centralized approach as well as a decentralized, a strong distinction between the two sides is no longer supported.

Req. ID: **526**

Description: Delineation between middleware and application in terms of context provision.

Fit criteria: In terms of context provision, middleware and application must be delineated.

Assessment procedure: Check whether context- and ontology managers are clearly delimitated.

Description of the assessment result: Supported. The Context Manager is responsible for collecting information from various context providers and performs low-level reasoning. The Ontology Manager takes context data as input and in turn performs higher-level reasoning. The Context Manager was withdrawn from the scope of the Ontology Manager.

Req. ID: **528**

Description: Specification of the information flow among Hydra managers.

Fit criteria: Complete specification that clearly defines how the information shall flow among Hydra managers.

Assessment procedure: Check the relevant documentation.

Description of the assessment result: Supported. D3.9 describes the information flow among all dependent managers.

4.2 WP4 validation results

Requirements from the previous cycle

Req. ID: **312**

Description: Support of profiling device performance. The middleware should contain services that allow monitoring and reaction on what devices are doing. This includes monitoring response time including service execution time, round trip invocation time, and device calling relationships. It also includes the profiling of device load (e.g., CPU) and memory, and power consumption.

Fit criteria: The services are available for Hydra developer.

Assessment procedure:

The calculation of service execution time, round trip invocation time, and device calling relationships is realized as service message probe, a plugin for Limbo, supported by a Messageprobe Ontology to calculate these parameters. The profiling of device load (e.g., CPU) and memory is achieved through the using of Nokia energy profiler, which support S60 3rd devices. The power consumption is measured through multimeter. These measurements are then encoded in the QoS ontologies, and are used by Self-management components and QoS manager if possible.

Description of the assessment result: supported, partly supported or not yet supported. The usage of Messageprobe Ontology is described in a SASO paper, and also deliverable D4.8. The measurements of service execution time, round trip time (RTT) are reported in D5.9. It was shown that the Limbo generated probe code for services and clients are useful for QoS parameter monitoring. And all the performance metrics are important as they are used for self-management.

Req. ID: **314**

Faults should be intercepted by middleware, notified to interested services. To create reliable and available systems it is essential to catch faults/partial failures before they become failures/complete failures. There needs to be uniformity in how this is done; thus it should be supported by the middleware.

Fit criteria: The middleware has support (through components/services) for sending and receiving notifications for partial failures.

Assessment procedure: Scenarios are first developed for fault detection, based on the real requirements for pig farm monitoring systems. Then the self-management components are used to detect possible failures, both device level and system level. We then evaluated the performances of our approach, both the Semantic Web based approach and the CPN based approach.

Description of the assessment result: It was shown that both approaches have acceptable performances, and CPN is better in this aspect, but the Semantic web based approach has bigger intelligence potential. The experiments have also shown that the extensibility is good for adding new features to the system. The description of these tests is reported in the SASO papers, and a SEKE paper.

Req. ID: **317**

Description: support for run-time reconfiguration. To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime.

Fit criteria: Services and devices can be connected in new ways during runtime in Hydra-based applications.

Assessment procedure: First the re-configuration scenario for Hydra middleware is developed, and then reconfiguration based on QoS requirements are used to calculate the components for Hydra middleware. Another scenario for self-protection is used related to run time reconfiguration, in terms that they configure the security protocols based on the QoS objectives.

Description of the assessment result: The evaluations are published in an ICECCS paper and a submission in ICSoC 2009. The tests shown that the performance for run time configuration is acceptable, which can potentially find a global optimized solution, which has acceptable solution quality.

Req. ID: **334**

Description: There should be support for developing auto-configuration of certain devices. A number of user scenarios calls for the ability to bring a device home, turn it on, and have it function reasonably.

Fit criteria: The middleware supports defining auto-configuration properties and using these at runtime. This is not in conflict with security.

Assessment procedure: Test execution of configuration script for network manager on osgi. An ASL script describes in a procedural way how to get from one configuration to another. If the start configuration is a bare device configuration then a script can capture and, when interpreted, enact a desired default configuration. As an example default configuration we consider an instance of the OSGi platform, in the case where the desired initial configuration is a running network manager (including crypto manager) connected to the Hydra network.

The script that accomplishes this is:

```
init_device(local);
init_component(CryptoManager,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/Crypto
Manager_1.0.0.jar);
deploy_component(local,CryptoManager);
init_service(local,CryptoManager,local_CryptoManager);
init_component(javax.servlet,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/javax.
servlet_2.4.0.v200806031604.jar);
deploy_component(local,javax.servlet);
init_service(local,javax.servlet,local_javax.servlet);
init_component(Log4j,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/Log4j_1.1.0.ja
r);
deploy_component(local,Log4j);
init_service(local,Log4j,local_Log4j);
init_component(Network_Manager_Bundle,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmja
rs/Network_Manager_Bundle_1.6.0.jar);
deploy_component(local,Network_Manager_Bundle);
init_service(local,Network_Manager_Bundle,local_Network_Manager_Bundle);
init_component(NetworkManagerConfigurator,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/
nmjars/NetworkManagerConfigurator_1.0.1.jar);
deploy_component(local,NetworkManagerConfigurator);
init_service(local,NetworkManagerConfigurator,local_NetworkManagerConfigurator);
init_component(org.apache.commons.logging,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/
nmjars/org.apache.commons.logging_1.0.4.v20080605-1930.jar);
deploy_component(local,org.apache.commons.logging);
init_service(local,org.apache.commons.logging,local_org.apache.commons.logging);
init_component(org.apache.log4j,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/org
.apache.log4j_1.2.13.v200706111418.jar);
deploy_component(local,org.apache.log4j);
init_service(local,org.apache.log4j,local_org.apache.log4j);
init_component(org.apache.xml.serializer,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/n
mjars/org.apache.xml.serializer_2.7.1.v200902170519.jar);
deploy_component(local,org.apache.xml.serializer);
init_service(local,org.apache.xml.serializer,local_org.apache.xml.serializer);
init_component(org.eclipse.core.jobs,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjar
s/org.eclipse.core.jobs_3.4.1.R34x_v20081128.jar);
deploy_component(local,org.eclipse.core.jobs);
init_service(local,org.eclipse.core.jobs,local_org.eclipse.core.jobs);
init_component(org.eclipse.equinox.cm,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmja
rs/org.eclipse.equinox.cm_1.0.0.v20080121.jar);
deploy_component(local,org.eclipse.equinox.cm);
init_service(local,org.eclipse.equinox.cm,local_org.eclipse.equinox.cm);
```

```
init_component(org.eclipse.equinox.common,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/
nmjars/org.eclipse.equinox.common_3.4.0.v20080421-2006.jar);
deploy_component(local,org.eclipse.equinox.common);
init_service(local,org.eclipse.equinox.common,local_org.eclipse.equinox.common);
init_component(org.eclipse.equinox.ds,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmja
rs/org.eclipse.equinox.ds_1.0.0.v20060828.jar);
deploy_component(local,org.eclipse.equinox.ds);
init_service(local,org.eclipse.equinox.ds,local_org.eclipse.equinox.ds);
init_component(org.eclipse.equinox.http.jetty,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resour
ces/nmjars/org.eclipse.equinox.http.jetty_1.1.0.v20080425.jar);
deploy_component(local,org.eclipse.equinox.http.jetty);
init_service(local,org.eclipse.equinox.http.jetty,local_org.eclipse.equinox.http.jetty);
init_component(org.eclipse.equinox.http.servlet,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/reso
urces/nmjars/org.eclipse.equinox.http.servlet_1.0.100.v20080427-0830.jar);
deploy_component(local,org.eclipse.equinox.http.servlet);
init_service(local,org.eclipse.equinox.http.servlet,local_org.eclipse.equinox.http.servlet);
init_component(org.eclipse.equinox.preferences,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resou
rces/nmjars/org.eclipse.equinox.preferences_3.2.201.R34x_v20080709.jar);
deploy_component(local,org.eclipse.equinox.preferences);
init_service(local,org.eclipse.equinox.preferences,local_org.eclipse.equinox.preferences);
init_component(org.eclipse.equinox.registry,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resource
s/nmjars/org.eclipse.equinox.registry_3.4.0.v20080516-0950.jar);
deploy_component(local,org.eclipse.equinox.registry);
init_service(local,org.eclipse.equinox.registry,local_org.eclipse.equinox.registry);
init_component(org.eclipse.osgi.services,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/n
mjars/org.eclipse.osgi.services_3.1.200.v20071203.jar);
deploy_component(local,org.eclipse.osgi.services);
init_service(local,org.eclipse.osgi.services,local_org.eclipse.osgi.services);
init_component(org.mortbay.jetty,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/or
g.mortbay.jetty_5.1.14.v200806031611.jar);
deploy_component(local,org.mortbay.jetty);
init_service(local,org.mortbay.jetty,local_org.mortbay.jetty);
init_component(org.os4os.forge.axisbundle,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/
nmjars/org.os4os.forge.axisbundle_3.0.6.jar);
deploy_component(local,org.os4os.forge.axisbundle);
init_service(local,org.os4os.forge.axisbundle,local_org.os4os.forge.axisbundle);
init_component(XMLSecurity,/Users/ingstrup/Documents/workspaces/Hydra2/ASL/resources/nmjars/XMLSecur
ity_1.0.0.jar);
deploy_component(local,XMLSecurity);
init_service(local,XMLSecurity,local_XMLSecurity);
start_service(local_org.eclipse.equinox.ds);
start_service(local_org.os4os.forge.axisbundle);
start_service(local_org.eclipse.equinox.registry);
start_service(local_org.apache.xml.serializer);
start_service(local_XMLSecurity);
start_service(local_NetworkManagerConfigurator);
start_service(local_Network_Manager_Bundle);
start_service(local_org.apache.commons.logging);
start_service(local_org.eclipse.osgi.services);
start_service(local_org.eclipse.equinox.preferences);
start_service(local_org.eclipse.core.jobs);
start_service(local_CryptoManager);
start_service(local_org.eclipse.equinox.http.servlet);
start_service(local_org.apache.log4j);
start_service(local_javax.servlet);
start_service(local_org.eclipse.equinox.cm);
start_service(local_org.mortbay.jetty);
start_service(local_org.eclipse.equinox.http.jetty);
start_service(local_org.eclipse.equinox.common);
```

The bare device configuration is in this case the Equinox osgi platform with the ASL interpreter bundle installed. This configuration can be reached as a default by using the existing configuration features of the equinox osgi platform. The script is device specific because it references files in a local file system.

Description of the assessment result: The script executes without faults, and manual inspection of the configuration was performed through (1) the osgi console to see what bundles were installed and that their status was as intended (2) the axis html site http://localhost:8082/axis/services showing that all web services were available as intended (3) the network manager status page http://localhost:8082/NetworkManagerStatus showed the network manager to be running and properly connected to the hydra overlay network. As such, inspecting the result reveals that the configuration script correctly transforms the platform into the intended configuration with the

network manager running and available on the local network. As a side remark it should be mentioned that debugging the script prior to performing the experiment showed that better error messages are needed from the ASL interpreter. Some of enhancements were implemented to this end, including printing line-numbers and indicating what step in the interpretation process went wrong (parsing, executing etc) in case of error.

Req. ID: **366**

Description: Services should run on embedded devices. Service-orientation is a good match for many embedded devices. Web services will provide a gateway to many applications and it would be beneficial to be able to structure all of the communication in a system using the same primitives. Depending on the resources (energy, processing capacity) available such a service may run on the device or on a proxy.

Fit criteria: Validate support for JME, SUN Spot and Lego NXT.

Assessment procedure: Experience report on using the Limbo compiler to generate web services for specific devices.

Description of the assessment result: Limbo has not been used to generate web services for ZigBee devices, because the device we have does not have JVM. As long as it has JVM, we can run web services directly on it. Web services for SUN Spot is also used (which does run iEEE 802.15.4), since focus on the Hydra has been on a middleware stack for integrating small devices. In general, our work so far has been on Java-based embedded devices: standard (JSE, JME) and product-specific (SUN Spot, NXT) variants.

Requirements for this cycle

Req. ID: **479**

Description: The Event Manager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system.

Fit criteria: Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort.

Assessment procedure: Stress test of the event notification system.

Description of the assessment result: The system has been tested and found to satisfy the fit criteria. There is a lower threshold in that the first events in a series of mixed-priority events published may not be handled according to priority, but at most one event out of order is handled this way.

4.3 WP5 validation results

Requirements from the previous cycle

Req. ID: **276**

Description: New communication technologies.

Fit criteria: 80% of new technologies are supported.

Assessment procedure: Integration of the ZigBee protocol and discovery mechanism.

Description of the assessment result: The validation of this requirement was done based on the integration of the ZigBee protocol and discovery mechanism into the existing set of technologies already in the Hydra middleware (at the time of validation), i.e., Bluetooth, Radio, Serial, RFID, UPnP/DLNA, IP/WIFI. Although this requirement is considered supported, the percentage expressed in the fit criteria is yet to be met.

Req. ID: **407**

Description: Storage Manager – Gateways information stored synchronization.

Fit criteria: 90% of the information stored in the Gateway is synchronized with the information stored inside the devices.

Assessment procedure: Data will be annotated with timing information, which will be used to evaluate applied (soft) real-time constraints.

Description of the assessment result: Not yet supported. Will be implemented.

Req. ID: **465**

Description: Networks overlapping.

Fit criteria: Device is not to be added to an existing Hydra network if it is unauthorised or when it belongs to another Hydra network, which is temporary near to the other present Hydra network.

Assessment procedure: Validation session with developers.

Description of the assessment result: This requirement is not yet supported. Resolution and enforcement of authorization is not in place yet.

Requirements for this cycle

Req. ID: **502**

Description: It should be possible to store simple key/value pairs.

Fit criteria: Storing and receiving cookies to a given Manager do not need more than 3 requests.

Assessment procedure: Building a test application that does not send more than 3 requests per access.

Description of the assessment result: Supported. Using the Cookie Manager, it is at first necessary to create a container to hold the key/value pairs. This can be done by one request. Storing a key/value pair is then done with one request. Reading a value can also be done in one request. Figure 2 describes the required requests.

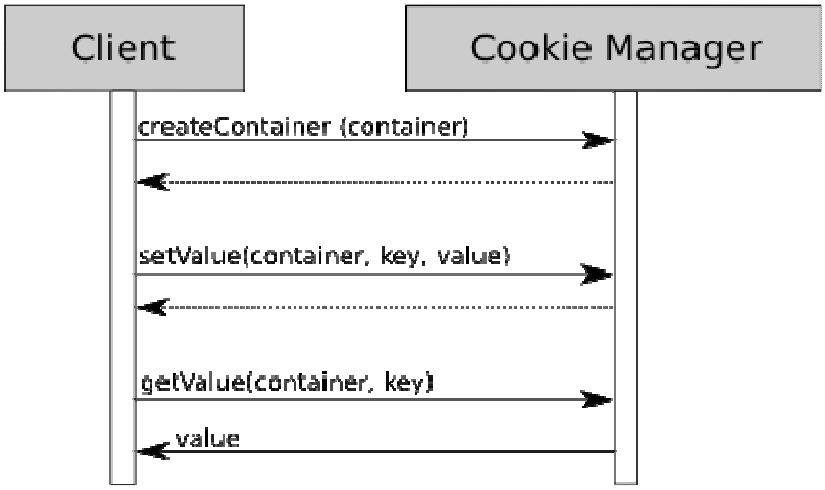

Figure 2: Storing and receiving cookies process

Req. ID: **503**

Description: It should be possible to combine different storage for mirroring or striping.

Fit criteria: Replicated and Striped devices can be built on top of each other.

Assessment procedure: Building a striped storage on top of two mirrored ones and a mirrored one on top of two striped ones.

Description of the assessment result: Partly supported. At the moment only the Replicated File System Device is implemented. This device can work on arbitrary devices, at the moment local and replicated. This has been tested by building a Replicated device on top of two other replicated devices as shown in Figure 3, each using two local devices as backend. The device was tested by using it as backend for the assessment described in 505.

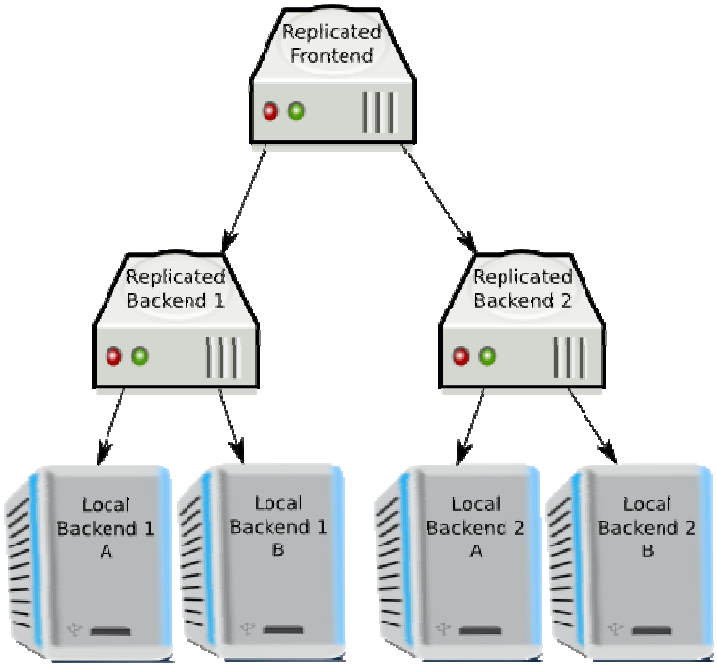

Figure 3: Replicated File System Device

Req. ID: **504**

Description: It should be possible to add and remove physical storage from a Mirror/Striping-Set.

Fit criteria: All striped devices can be enlarged by adding new physical storage.

Assessment procedure: Adding 10 devices to a storage device and removing them.

Description of the assessment result: Not yet supported.

Req. ID: **505**

Description: It should be possible to access data in Storage Manager using a well-defined protocol (e. g. WebDav).

Fit criteria: 50% of the storage can be accessed by non-Hydra applications.

Assessment procedure: The File System Device will be mountable using Fuse. This will be tested by mounting a File System Device on a Linux host and accessing it using desktop applications.

Description of the assessment result: Supported. Using the File System Device FUSE Client any File System Device can be mounted on Linux systems. We mounted a Local and a replicated device and tested it by following procedure:

- Read Root Directory
- Create a directory
- Create a File in the directory
- Write data to the File
- Read the File
- Read the directory
- Remove File
- Remove directory

All steps succeeded.

Req. ID: **506**

Description: It should be possible to lock files (Storage Manager).

Fit criteria: All write access is aborted if a file is locked.

Assessment procedure: A Lock Manager will be implemented that provides the ability to get and release locks on entities like files and directories. Validation will be done by trying to sequentalize multiple accesses.

Description of the assessment result: Not yet supported.

Req. ID: **488**

Description: In order to simplify and speed up the integration of new wireless devices in Hydra, the discovery and proxy creation process has to be standardized and be as modular as possible, so common parts can be reused by proxies for different wireless devices.

Fit criteria: 30% of proxy modules rely on common kernels.

Assessment procedure: Limbo tests on code reuse and modularization. We will create two proxies and check the code modularization and the reuse of code.

Description of the assessment result: We have created two proxies, one for a Door Trap and one for a Dimmer Switch using Limbo Tool. The generated code is very similar in both cases. They share the same core packages for the proxy (UPnP, BundleActivators, Servlets) except for the device-specific services (i.e. the code depends on the methods and services of each device). The main difference between the generated code is the implementation of the services specific for each device, in which the developer has to provide the implementation. Also, the different parts of the code are clearly separated and there is a specific package for each specific functionality (UPnP, Servlets, Parsers, WS code, etc.).

Therefore, we can state that the generated proxy code is reusable and modular so this requirement has been fulfilled and validated.

Req. ID: **486**

Description: At the moment, public supernodes are used to act as relays in D2D communication. If these supernodes are down, communication between networks is impossible. Thus, we need to manage our own supernodes in partner's servers.

Fit criteria: 80% of the time, own supernodes are up and running and Network Managers can join the Hydra Network (even when they are located behind firewalls or NATs).

Assessment procedure: Supernode deployment on CNET and FIT servers and test that Network Managers can join the Hydra Network.

Description of the assessment result: We have set up two super nodes, one in CNET premises and one in FIT premises, as it is shown in Figure 4. These special Network Managers help with the network creation and bootstrapping. After this setup, we have started several (up to 15) Network Managers and all of them, 100%, joined the Hydra Network successfully and were able to communicate with each other. Thus, this requirement has been validated successfully.

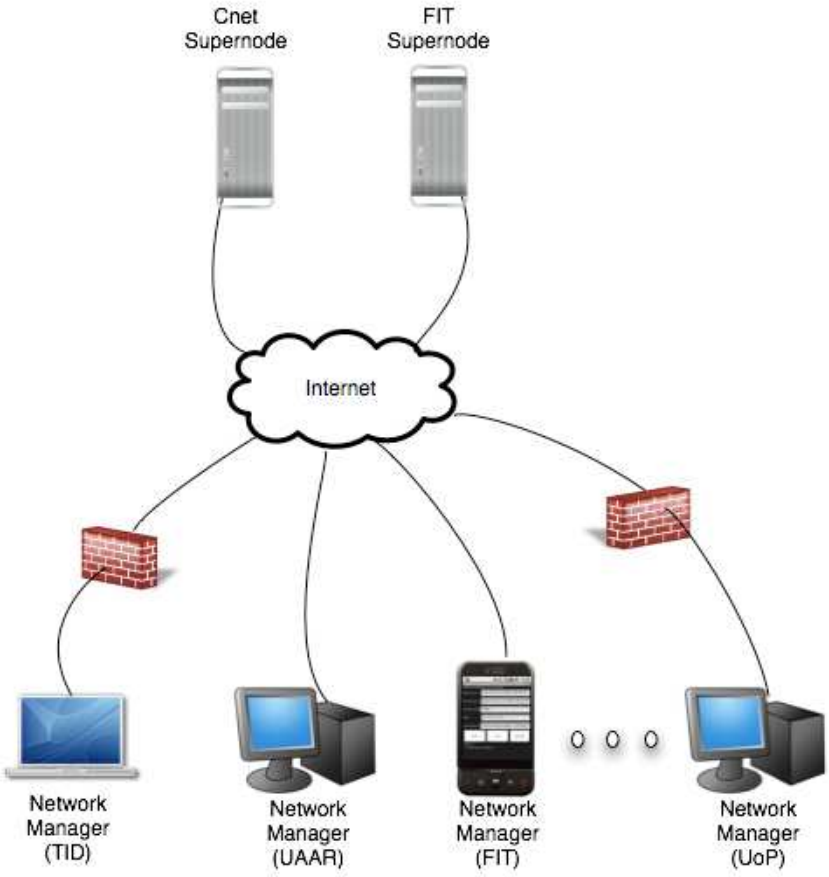

Figure 4: Supernodes deployment

Req. ID: **442**

Description: Proxy-Gateways can filter and react to data received from associated non-hydra devices. Part of the proxy functionality may include support for filtering of the received data and possibly a reaction to high or low values. Non-Hydra devices cannot be expected to analyze the data themselves, so the gateways could take care of this.

Fit criteria: 50 % of Gateways supports filtering and reaction to received data.

Assessment procedure: Generate device proxies using Limbo and use Context Manager on the gateway to test the Data Acquisition.

Description of the assessment result:

Figure 5 shows the interaction of a Non Hydra Device within the Hydra Data Acquisition Process, which is described in the dedicated deliverables about data acquisition and context management.

Basically the following steps are processed:

- Hydra Limbo service is generated to Hydra and enable the Non Hydra Device to deploy web services for accessing the device.
- The DAC is used by the Data Acquisition component to access the data produced by the device. The generic DAqC is configured through an interface which can be accessed by the Context Manager as well as by other Hydra components.
- Within the DAqC a first data validation is performed to check the plausibility of the incoming data.
- The Hydra Component or Application shown in the figure is usually the Context Manager which is configured through the application by developer. There the semantics of the data is further processed with the help of the Ontology Manager (not shown in the figure).

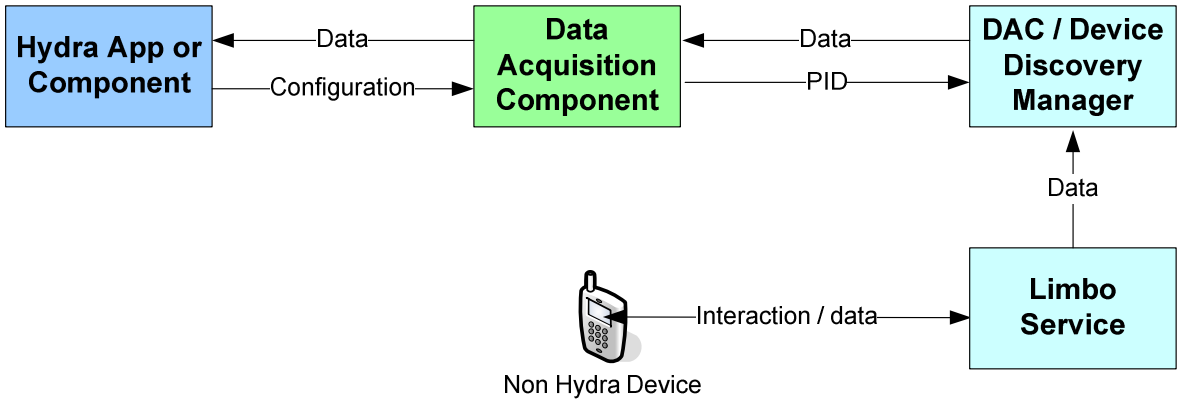

Figure 5: Non Hydra Device used within Hydra Discovery and Data Acquisition

Req. ID: **427**

Description: The D2D communication system has to allow the Hydra enabled device to create, join and leave groups of Hydra enabled devices, so the components of these groups share the same credentials and can communicate isolated from non-group-members.

Fit criteria: 90% of the devices involved in the D2D communication system can create, join and leave groups.

Assessment procedure: Test group creation for applications.

Description of the assessment result: Group creation is not yet supported in the current version of the middleware, so this requirement cannot be assessed.

Req. ID: **396**

Description: Hydra-enabled devices may be mobile or fixed equipment. A subset of the Hydra middleware (mainly Network Manager) can be deployed in mobile (PDA, Smartphone) and in resource-constrained devices (Home Gateways).

Fit criteria: 30% of state of the art PDAs, Smartphones and Home Gateways can host part of the Hydra middleware.

Assessment procedure: Test the implementation on Home Gateway, Play Station 3, and Android mobile phone.

Description of the assessment result: The Lite Version of Network Manager has been tested on a Play Station 3, Inaccess Home Gateway and Android mobile phone. In all the cases the Network Managers were able to communicate with each other and with standard Network Managers deployed on PCs.

Figure 6: PlayStation3

Figure 7: Inaccess Home Gateway

Figure 8: Android G1 phone

Therefore, it is tested that this version runs on mobile and resource constrained devices. The requirements for running these versions are:

Network Manager Version	Current version	CPU	Memory	Storage	Java VM
J2SE	2.0	1GHz	64 MB	20 MB	J2SE
Lite CDC	1.0	400 MHz	30 MB	8 MB	J2ME CDC
Lite CLDC	1.0	250 MHz	10 MB	1 MB	J2ME CLDC MIDP 2.0

Req. ID: **446**

Description: Security parameters negotiation.

Fit criteria: In 90% of all cases the parameters should be flexible.

Assessment procedure: Make use of handshake protocol and the security related mangers also with the security ontology to test these negotiation mechanisms. Rules set to different levels of security. Test minimum and maximum levels.

Description of the assessment result: The integration of the handshake protocol has been tested successfully. It is integrated in the Network Manager and communicates with Crypto and Trust Manager to use the services for encryption and decryption as well as for token verification. The obligation policy framework can be used to set up different security configurations.

Req. ID: **487**

Description: Improve handshake protocol between Network Managers for exchanging certificates.

Fit criteria: In 95% of the cases simple protocol would work.

Assessment procedure: Use different managers, with different keys, and different protocol standards to exchange data between them. Turn on handshake protocol and detect errors either in encryption or keystore methods.

Description of the assessment result: The handshake protocol has been integrated into the Network Manager. It can be used to exchange certificates between two managers to initialize a security session for inside hydra security.

4.4 WP6 validation results

Requirements from the previous cycle

Req. ID: **91**

Description: Any Hydra device should have an associated description.

Fit criteria: Any device associated to a Hydra application is also included in the Hydra device ontology, and its description can be retrieved.

Assessment procedure: Check that a newly discovered device has/gets a corresponding representation in the Device Ontology.

Description of the assessment result: As a part of the discovery process all devices will also be resolved against the devices ontology (semantic discovery). Interfaces exist that allows a developer to browse and retrieve the device descriptions in the device ontology. Devices can also be manually classified.

Req. ID: **110**

Description: Device categorization in run-time.

Fit criteria: 7 of 10 devices are correctly categorized and described.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result: The last step in the discovery process is to categorise a device based on device ontology information (aka the semantic discovery). The accuracy of this categorization depends on the completeness of the device descriptions and taxonomy in the device ontology. However, all devices that are discovered on the physical and network levels will also be resolved against the device ontology.

Req. ID: **114**

Description: Semantic enabling of web services.

Fit criteria: 7 of 10 devices are semantically enabled.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result: The intention here is that the system should be able to associate semantic descriptions to device (web) services based on the device ontology.

After a device has been discovered (physically and thru UPnP) the discovery process will try to resolve the device semantically against the device ontology. Depending on the result of this resolution, the discovery process will generate the necessary web service interfaces for the device. This requirement is currently partly supported, in that it may require manual intervention by updating the device ontology.

Req. ID: **122**

Description: Configurable and easy to install middleware.

Fit criteria: The average installation time is less than 1 hour.

Assessment procedure: Time of middleware installation.

Description of the assessment result: The final installation procedures and scripts are under development. The current Hydra implementation and configuration can meet the installation time constraint in most cases, but still requires manual intervention.

Req. ID: **376**

Description: Security requirements must be part of the Hydra MDA.

Fit criteria: Security model can be defined semantically.

Assessment procedure: A semantic security model exists, check resolution process.

Description of the assessment result: We can conclude that security requirements can be included in the MDA (the model driven architecture) of Hydra, i.e., the security meta model describes security requirements and policies, and a security ontology is in place. To validate this requirement, it is necessary to define security at the application and device levels and to resolve it semantically.

Requirements for this cycle

Req. ID: **501**

Description: A Hydra enabled device must support UPnP discovery

Fit criteria: All HYDRA enables devices support UPnP.

Assessment procedure: All Hydra devices are found thru UPnP.

Description of the assessment result: The UPnP is an integral part of the Hydra discovery process. Hence, any device discovered by Hydra will also be UPnP enabled.

Req. ID: **500**

Description: Semantic annotations of devices using SAWSDL.

Fit criteria: For a given UPnP discoverable device, it is possible to create an SAWSDL annotation which can be accessed from the UPnP discovery information.

Assessment procedure: Annotations can be attached and retrieved for any device.

Description of the assessment result: Interfaces exist that allow a (device) developer to annotate device WSDLs using SAWSDL. These SAWSDL annotations refer to the properties of device descriptions in the Device Ontology. SAWSDL annotations can also be generated from the device descriptions in the ontology.

Req. ID: **477**

Description: Device proxies should make use of available security features for "last mile" communication.

Fit criteria: Device proxies must support WEP and WPA for WiFi-connections as well as Bluetooth authentication and encryption.

Assessment procedure: Device proxies can be created that use WEP and WPA for WiFi-connections as well as Bluetooth authentication and encryption.

Description of the assessment result: The proxy architecture of Hydra does not constrain or prescribe the use of the underlying protocol for communicating with a device. Any device proxy implementation can be designed to use the available security features of the current protocol.

Req. ID: **126**

Description: Automatic Device ontology updates.

Fit criteria: The device ontology can detect device updates and handle that in 7 of 10 cases.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely, device discovery results in an ontology update.

Description of the assessment result: The Ontology Manager supports this requirement by the automatic update of device descriptions from the parsing of WSDL and SAWSDL files associated to devices.

Req. ID: **120**

Description: Automatic Multiple Device Virtualisations.

Fit criteria: Multiple Device Virtualisations.

Assessment procedure: A developer is able to create at least two different views onto the same physical device.

Description of the assessment result: A Device Virtualization is here understood as the logical Hydra representation of a physical device. It is possible to create several different views/virtualisations of a physical device depending on network context and applications. The descriptions in the Device Ontology determine the initial device view after the device has been discovered.

Req. ID: **117**

Description: HYDRA component ontology.

Fit criteria: HYDRA device and service managers can be identified and selected through a software component ontology.

Assessment procedure: HYDRA device and service managers can be identified and materialized/displayed.

Description of the assessment result: The discovery process currently works with an implicit software component model. This model represents the device managers and service managers that are selected for automatic proxy generation. This model is currently not available to the developer, but the device and service manager objects are, and can be specialized.

This requirement is also supported by the ASL (Architecture Scripting Language) of WP4 which works with an explicit software component model for deploying Hydra component configurations.

Req. ID: **114**

Description: Semantic enabling of device web services.

Fit criteria: 7 of 10 devices are semantically enabled.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result: The Device Ontology includes a service model subset which can be used to for semantic web service description. The association of semantic descriptions to the device web services can be based on the parsing of device WSDL and SAWSDL files.

Req. ID: **113**

Description: Composition (of services and devices).

Fit criteria: Service composition during design-time is possible.

Assessment procedure: Design an application composed of at least two different devices of different types and with different services.

Description of the assessment result: A developer is able to compose services and devices from different providers and/or manufacturers into high level services/devices in an application. This is currently done using the programming language of the SDK/DDK. Composition and aggregation is

also supported by constructs for Semantic Devices, which are higher level programming constructs (devices), based on the combination existing Hydra Devices.

Req. ID: **112**

Description: Dynamic Web Service Generation.

Fit criteria: 7 of 10 device functionalities are automatically represented as web services.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result: As a result of the device discovery process and device proxy creation, web service interfaces are generated. There are three categories of services generated, Generic Hydra Services, Energy Profile Services, and the Device Specific Service.

Req. ID: **104**

Description: Automatic Discovery of Services.

Fit criteria: 8 of 10 services are automatically discovered.

Assessment procedure: Enter new devices into a Hydra network, locally and remotely.

Description of the assessment result: The current discovery model is based on the discovery of physical devices. The device services can be "discovered" in subsequent steps by the application. It is also possible to use the Ontology Managers search and browser interfaces to find services by various selection criteria.

4.5 WP7 validation results

Requirements from the previous cycle

Req. ID: **308**

Description: The Security Level of an existing network should be determinable.

Fit criteria: Hydra middleware provides at least one mechanism enabling devices to determine the security level of an existing network.

Assessment procedure: Evaluation of the current status of the middleware security architecture.

Description of the assessment result: As already stated in D10.2, the term "Security level" in this requirement is relatively fuzzy. In order to fulfil this requirement completely, it must be possible to observe the security mechanisms that are used in a network and to estimate their "Security level", i.e. their strength. The second point is fulfilled: Hydra's security ontology allows to reason about assurances that have been provided by certain institutions for different security algorithms and protocols. However, it is impossible to automatically observe security mechanisms that are used by entities in a network for a device that wants to join the network. Many security mechanisms, especially those which are implemented in layers below the Hydra middleware, are transparent to the middleware layer in general and so information about these layers has to be offered explicitly. One way to do this is to use the domain controller which can inform a joining device about the security level of his domain. The joining device might accept this information depending on the trust relationship to the domain controller. In detail, the domain controller might sign his "security level" claims with a certificate that is somehow trustable for the joining device.

Even though the arguments presented in D10.2 for the first unsuccessful assessment of this requirement are still valid and the implemented mechanisms regarding a determinability of "secure level" haven't changed significantly, we now declare this requirement as supported. The reason for the changed conclusion is that the implemented domain model now allows negotiating security mechanisms based on trust, and the security ontology is able to support the estimation of corresponding security level for a joining device.

Req. ID: **468**

Description: Different levels of security must be supported.

Fit criteria: It must always be possible to implement at least two different security levels for an application.

Assessment procedure: Evaluation of the current status of the middleware security architecture.

Description of the assessment result: In terms of cryptography for message protection, this requirement is fulfilled as the modules for Core Hydra and Inside Hydra communication protection are based on XMLEncryption[1] and XMLSignature[2]. Both standards define a message format for protected data but leave it up to the developer to use a suitable cryptographic algorithm from a list of recommended ones. In that way, different "security levels" in terms of "algorithms" and "key lengths" are supported.

Besides, "security level" could also be understood in the sense of a set of access-control policies. The "security level" could be higher the more access to different services is restricted by those policies. Even in that way, the requirement can be considered fulfilled as Hydra's policy framework will provide the basis for defining and enforcing such access-control rules.

[1] http://www.w3.org/TR/xmlenc-core/
[2] http://www.w3.org/TR/xmldsig-core/

D10.2 stated that it was not clear what the original intention of that requirement had been. In many aspects the requirement is supported by Hydra, but overall we marked this requirement ambiguous and won't continue further assessments.

Requirements for this cycle

Req. ID: **364**

Description: Hydra's Access-Control policies support credential based authentication.

Fit criteria: Access-control can be based on credentials.

Assessment procedure: Evaluation of the Access Control Policy Framework.

Description of the assessment result: Credentials are used for authentication in the Hydra Access-Control Policy Framework, with the use of various credentials relating to the subject and resource of a request that are supplied by the Network Manager. These credentials include the HIDs of the subject / resource, as well as the attributes associated with the cryptoHID (certificate bound to HID) of the subject / resource. This includes such (secure) credentials as a Service Identifier, Service Description and a Persistent Identifier. The implementation of this feature has been tested successfully, but integration and testing are ongoing.

Req. ID: **498**

Description: Mechanisms used for communication security should be replaceable by configuration.

Fit criteria: For at least two of the communication protection mechanisms (Core / Inside / Outside Hydra) it should be possible to replace security modules without recompiling the middleware.

Assessment procedure: Evaluation of the current status of the middleware security architecture.

Description of the assessment result: Both the communication mechanisms for core and inside Hydra security support the reconfiguration of security mechanisms without a need to recompile elements of the middleware. The basic functionality for communication security is provided by the security library and the communications protocols which are explained in D7.8 "Security and Privacy components for DDK prototype" and D7.7 "Security Architectural Models Design Specification".

Req. ID: **509**

Description: Enforcement of Access-control policies.

Fit criteria: Policy enforcement points can be attached to Hydra web services so that access control decisions can be enforced.

Assessment procedure: Evaluation of the Access Control Policy Framework.

Description of the assessment result: Policy Enforcement Points have been attached to Network Managers. These Network Managers can route the requests to its local services through the PEP, along with collected credentials about the subject and resource of the request, for an access decision.

Req. ID: **510**

Description: Enforcement of obligation policies.

Fit criteria: Hydra components negotiate their capability to enforce different actions with the policy decision.

Assessment procedure: Evaluation of the current status of the middleware security architecture.

Description of the assessment result: An obligation enforcement bundle has been implemented which is able to hand obligations to different types of enforcement plugins. Currently, two different types of enforcement plugins are available: one to manage and execute instructions of the OSGi framework and one to execute ASL-scripts. Therefore the requirement is supported. Further details of the obligation policy framework are available in deliverable D7.7.

4.6 Summary of the evaluated requirements

In the following table we summarise the results obtained for the validation of the selected requirements in both the first and second cycle.

WP3		I Cycle	II Cycle
18	Support for different software architectural patterns	Supported	
31	An easy-to-use programming framework should be provided	Not yet supported	Partly supported
33	Enable manufacturers to develop devices and applications that can be connected to Hydra	Supported	
41	Hydra Developer's Companion	Partly supported	Partly supported
136	Dynamic architecture	Not yet supported	Partly supported
185	Middleware provides basic services	Partly supported	Supported
186	GUI for configuring middleware parameters	Supported	
199	Modules should be extendable	Partly supported	Partly supported
207	Service selection by context	Partly supported	Partly supported
217	The middleware should ensure high robustness of services	Partly supported	Partly supported
234	The middleware should be self descriptive	Not yet supported	Not yet supported
320	Separate domain-oriented services and user interface services architecturally	Not yet supported	Not yet supported
327	The Hydra middleware should be flexible as to allow for opt-in and opt-out on parts	Supported	
329	Middleware provides domain-independent services	Supported	
335	Location awareness / positioning support	Partly supported	Partly supported
515	Support of domain-specific ontologies		Partly supported
518	No external standards should dictate the virtual layer		Supported
519	It should be possible to implement managers in either programming model.		Supported
522	All HYDRA entities must have a semantic model description		Supported
524	Determination and Description of the dependencies among Hydra Managers.		Supported
525	Delimitation between Application and Device Elements.		Supported
526	Delineation between middleware and application in terms of context provision		Supported
528	Specification of the information flow among Hydra Managers.		Supported
WP4			
312	Support profiling of devices' performance	Partly supported	Partly supported
314	Faults should be intercepted by middleware, notified to interested services	Partly supported	Supported
317	Support runtime reconfiguration	Not yet supported	Supported
318	Devices should be able to be added to the system at runtime	Supported	
334	There should be support for developing auto-configuration of certain devices	Not yet supported	Supported

366	Web services should run on embedded devices	Not yet supported	Supported
479	Event prioritisation	Supported	Supported
WP5			
264	Common message protocol	Supported	
276	New communication technologies	Supported	Supported
336	Discovery protocol should support multiple networks	Supported	
407	Storage Manager – Gateways information stored synchronization	Not yet supported	Not yet supported
419	Device services and resources provision through its Gateway	Supported	
425	D2D communication Overlay Hydra network	Supported	
445	The level of protection should be independent from the currently used low-layer protocol	Supported	
455	Identity - Update of the correspondences between identifier and physical addresses	Supported	
465	Networks overlapping	Not yet supported	Not yet supported
475	Multimedia streaming in the Hydra network	Supported	
476	Network Manager Configuration and Testing	Supported	
396	Hydra-enabled devices – May be mobile or fixed equipment		Supported
427	D2D communication – Group management		Not yet supported
442	Proxy – Gateways can filter and react to data received from associated non-hydra devices		Supported
446	Security parameters negotiation		Supported
486	Hydra proprietary supernodes are needed to support D2D communication between networks		Supported
487	Improve handshake protocol between Network Managers for exchanging certificates		Supported
488	Modular and standard device integration		Supported
502	It should be possible to store simple key/value pairs		Not yet supported
503	It should be possible to combine different storage for mirroring or striping		Partly supported
504	It should be possible to add and remove physical storage from a Mirror/Striping-Set		Not yet supported
505	It should be possible to access data in Storage Manager using a well defined protocol (e. g. WebDav)		Supported
506	It should be possible to lock files (Storage Manager)		Not yet supported
WP6			
91	Any HYDRA device should have an associated description	Not yet supported	Supported
101	Manual device ontology definition	Supported	
108	Device discovery	Supported	
110	Device Categorisation in runtime	Partly supported	Supported
111	Dynamic Web Service Binding	Supported	
114	Semantic enabling of device web services	Not yet supported	Partly supported

122	Configurable and easy to install middleware	Not yet supported	Not yet supported
129	Support for Semantic Web Standards for Device Communication	Supported	
210	Middleware should support different architectural styles	Supported	
376	Security requirements must be part of the Hydra MDA	Not yet supported	Supported
389	Service browsing in device ontology	Supported	
104	Automatic Discovery of Services		Partly supported
112	Dynamic Web Service Generation		Supported
113	Composition (of services and devices)		Supported
114	Semantic enabling of device web services		Partly supported
117	HYDRA component ontology		Partly supported
120	Multiple Device Virtualisations		Supported
122	Configurable and easy to install middleware		Not yet supported
126	Automatic Device ontology updates		Partly supported
477	Device proxies should make use of available security features for "last mile" communication		Supported
500	Semantic annotations of devices using SAWSDL		Supported
501	A Hydra enabled device must support UPnP discovery		Supported
WP7			
308	The Security Level of an existing network should be determinable	Not yet supported	Supported
468	Different levels of security must be supported	Not yet supported	**No further assessment**
472	Provide application developers with the functionality of checking tokens against a trust model	Supported	
473	Support of arbitrary trust models	Supported	
474	Core Hydra security mechanisms should run on embedded devices	Supported	
364	Hydra's Access-Control policies support credential based authentication		Supported
498	Mechanisms used for communication security should be replaceable by configuration		Supported
509	Enforcement of Access-control policies		Supported
510	Enforcement of obligation policies		Supported

Table 12: Summary of evaluation results

From the table it is possible to sketch the graphics on the successfulness rate for the actual validation, in terms of requirements' percentages reaching the threshold.

On the average, 86% of the tested requirements have been partly or completely covered, as it appears in Figure below.

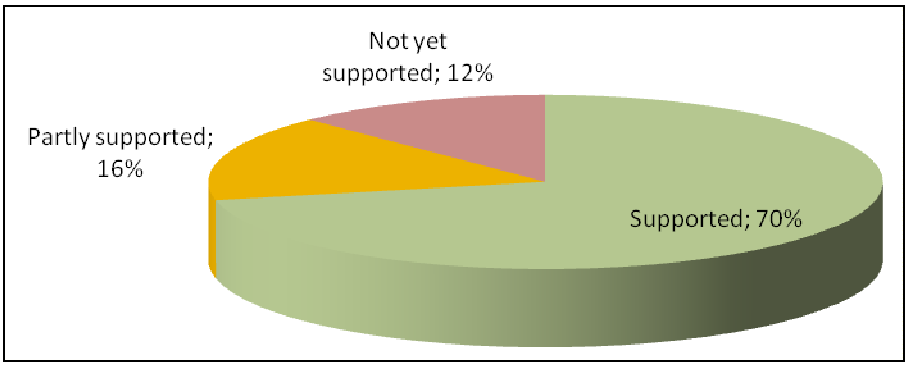

Figure 9: Overall success percentages after 2nd validation cycle

In the next summarising table we present the results obtained divided per WP. The indication is not relevant in terms of quantitative aspects, but it is considered as a basic reference for the future development and validation activities to be fulfilled during the next project iteration.

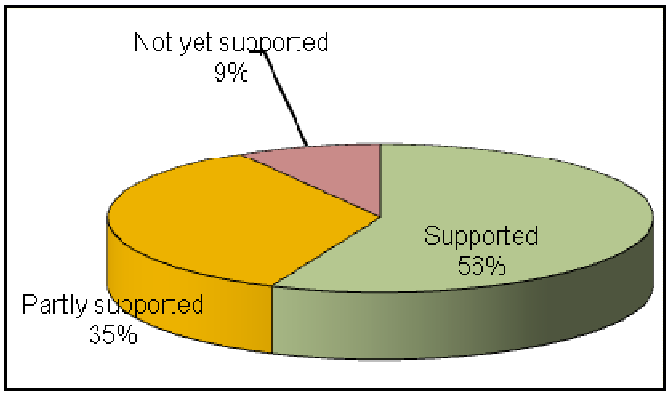

Figure 10: Requirements fulfilment for WP3

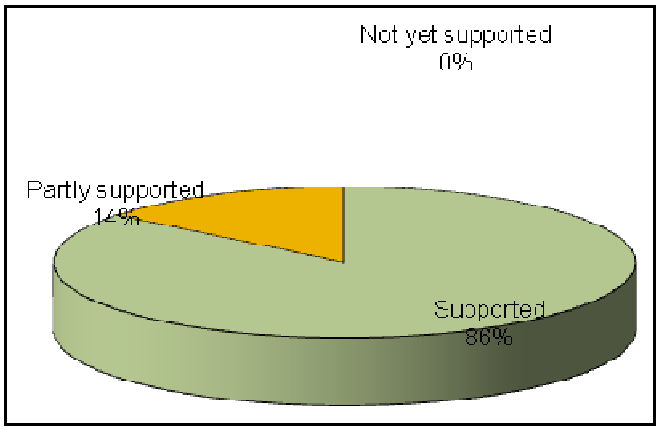

Figure 11: Requirements fulfilment for WP4

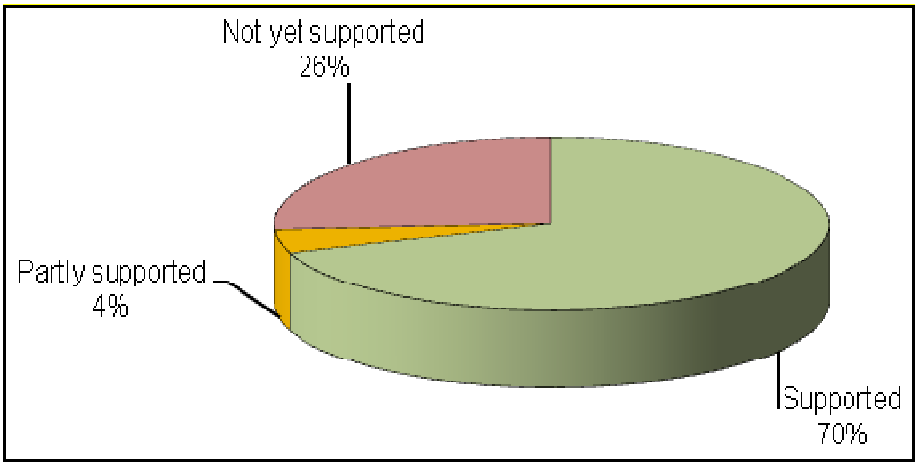

Figure 12: Requirements fulfilment for WP5

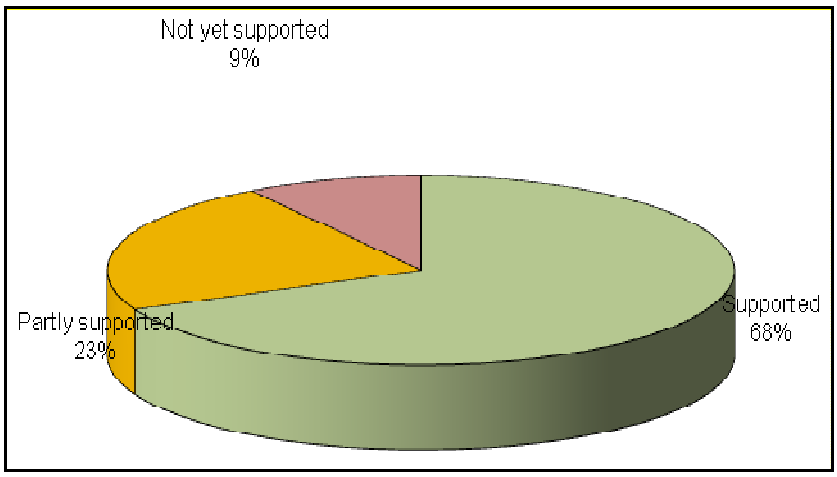

Figure 13: Requirements fulfilment for WP6

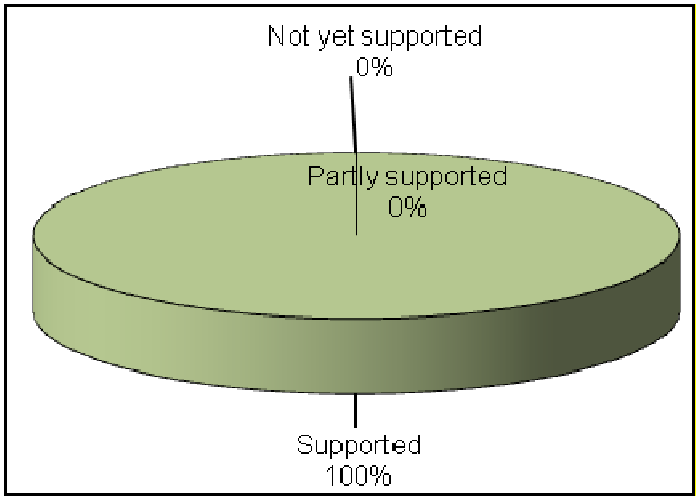

Figure 14: Requirements fulfilment for WP7

5. Conclusions

The validation methodology has been built and applied by the comparison between an expected impact (requirement) and how the real prototype or application behaves. The assessment procedure was applied from the (potential) Hydra user, who is a developer or a software expert able to recognise if the promised features and properties of the Hydra middleware are met. The environment selected for the validation was the software laboratory of the Hydra partners, where potential developer users, not previously working in the Hydra implementation, were selected and carried out the assessment.

More in details, the validation methodology consisted in the verification that each selected requirement fit criterion has reached the threshold level, or whether the requirement has been partially met or has not been met. The selection of the requirements to be validated has been fulfilled by considering the following parameters:

- effective implementation or not of the requirements (in respect to the actual timing or status of the project)
- relevance for the overall architecture (cross related features)
- requirement type and priority

In total, i.e. considering the first and the second validation cycles, 83 requirements have been assessed. The overall results are summarised into the following table.

Assessment threshold level	N. of requirements fulfilling the threshold
Supported	58 (70%)
Partly supported	13 (16%)
Not yet supported	10 (12%)

Table 13: Overall success rate.

Notice that the number of supported requirements is improved compared to previous cycle (it was 52%). In particular, the number of requirements "not yet supported" has decreased (it was 31%).

Specifically, in the second validation cycle, in total 57 requirements have been assessed:

- 22 requirements have been re-assessed, because they were not yet or partly supported in the first validation cycle.
- 35 requirements have been assessed for the first time.

Focusing on the re-assessed requirements only, we can state that 12 requirements out of 22 (54%) moved from not yet supported to supported or partly supported, or moved from partly supported to supported; i.e. we had a substantial improvement in the development of SDK and middleware in the last year.

Focusing on the newly assessed requirements only, the results are summarised into the following table.

Assessment threshold level	N. of requirements fulfilling the threshold
Supported	24 (69%)
Partly supported	6 (17%)
Not yet supported	5 (14%)

Table 14: New requirements success rate.

Notice that the success rate for the new requirements of this second validation cycle has improved compared to the success rate of the first validation cycle (it was 52%).

These validation outcomes clearly show that the Hydra platform implementation is properly (and with an increasing speed) pursuing the target objectives. Most likely, the improved know-how of researchers and developers about the involved technologies and features of the platform helped the achievement of these improved results.

However, it is worth to highlight that this is just an intermediate result. The next validation cycle should confirm (and possibly improve) the obtained results, following the current development trend.

Similarly to the previous validation cycle, the results of this phase and, thus, the user feedbacks, are given back to the developers of the system by continuing the iterative approach. The data emerged in the present analysis will be distributed to the Hydra consortium (starting from each technical Work Package, but also looped back to WP2), as a mean for refining the user requirements, better detailing the project lessons learnt and continuously improving the system characteristics.

6. References

Foglia, T. and Costa, N. (2007). *Validation Framework*. Technical Report D2.6, Hydra Consortium. EU Project IST 2005-034891

Guarise, A. and De Bona, M. (2008). *Validation Plan for prototypes*. Technical Report D10.1, Hydra Consortium. EU Project IST 2005-034891

Hansen, K.M. (2007). *Quality Attribute Scenarios*. Technical Report D6.1, Hydra Consortium. EU Project IST 2005-034891

Klaus Marius Hansen, Weishan Zhang, Mads Ingstrup: Towards Self-Managed Executable Petri Nets. SASO 2008: 287-296

Kupries, M. and Hansen, K.M. (2007). *Updated Systems Requirements Report*. Technical Report D3.2, Hydra Consortium. EU Project IST 2005-034891

International Standards Organisation website http://www.iso.org/

Wahl, T., Hoffmann, M. and Schütte, J. (2008). *Impact of architectural security implications*. Technical Report D7.6, Hydra Consortium. EU Project IST 2005-034891

Weishan Zhang, Klaus Marius Hansen: An Evaluation of the NSGA-II and MOCell Genetic Algorithms for Self-management Planning in a Pervasive Service Middleware. Proceedings of the 14th IEEE International Conference on Engineering of Complex Computer Systems.

Weishan Zhang, Julian Schütte, Mads Ingstrup, Klaus M. Hansen. A Genetic Algorithms-based Approach for Optimized Self-protection in a Pervasive Service Middleware.(submitted for ICSoC 2009).

Weishan Zhang, Klaus Marius Hansen: An OWL/SWRL Based Diagnosis Approach in a Pervasive Middleware. SEKE 2008: 893-898

Zimmermann, A. (2008). *Updated System Architecture Report*. Technical Report D3.9, Hydra Consortium. EU Project IST 2005-034891