Contract No. IST 2005-034891

Hydra

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

Validation Report for IDE 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 - D10.4 Validation Report for IDE 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)	18-05-2010	Document organisation, objectives and first structure
0.2	A. Gugliotta, D. Ferulli, A. Guarise (INN) M. Ahlsen (CNet), M. Ingstrup (UAAR), M. Jahn (FIT), T. Wahl (SIT), F. Milagro (TID)	30-06-2010	Description of the approach and list of requirements for the third cycle of validation with contributions from WP leaders
0.3	A. Gugliotta, D. Ferulli, A. Guarise (INN) M. Ahlsen (CNet), M. Ingstrup (UAAR), M. Jahn (FIT), T. Wahl (SIT), F. Milagro (TID)	15-07-2010	Added text descriptions in sections 1,2 and 3 with contributions from WP leaders
0.4	A. Gugliotta, D. Ferulli, A. Guarise (INN) M. Ahlsen (CNet), M. Ingstrup (UAAR), M. Jahn (FIT), T. Wahl (SIT), F. Milagro (TID)	30-07-2010	Added text descriptions in section 4 with contributions from WP leaders
0.5	A. Gugliotta, A. Guarise, S. Galizia (INN)	01-09-2010	Included last contributions and finalisation of sections 4 and 5
1.0	A. Gugliotta (INN), A. Guarise, S. Galizia (INN)	27-09-2010	Addressed internal reviewers' comments

Internal review history:

Reviewed by	Date	Comments
M. Crouch (UR)	22-09-2010	Approved with comments
Helene Udsen (In-Jet)	26-09-2010	Minor general editing, comments/questions

Index

List of figures

List of tables

1. Introduction

1.1 Purpose and context

This deliverable provides the results of the third iteration validation phase focused on the IDE 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 IDE prototype and be able to deploy effective applications based on Hydra. The validation tests have been fulfilled during the implementation phase and in a specific testing activity and it involved both the third Hydra prototypes (IDE) 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.

The iterative approach followed in Hydra consists of successive improvements of the software package, and the validation fulfilled during this third 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 previous validation phases (reported in D10.2) but resulted in either "partly supported" or "not yet supported".

1.2 Outline

The present validation report represents the third 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 (D10.3). 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 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 with 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 phases.

Section 4 reports the results obtained while applying the assessment procedures for the evaluation of the fit criterion 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; this means (since we are in the last stages of the project) that an additional effort will be made to achieve such a requirement in the very last release of the software before the end of the project. To conclude the section, a table summarises all the requirements tested throughout the project (first, second and third cycles) 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 the expected progress by the end of the project.

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.

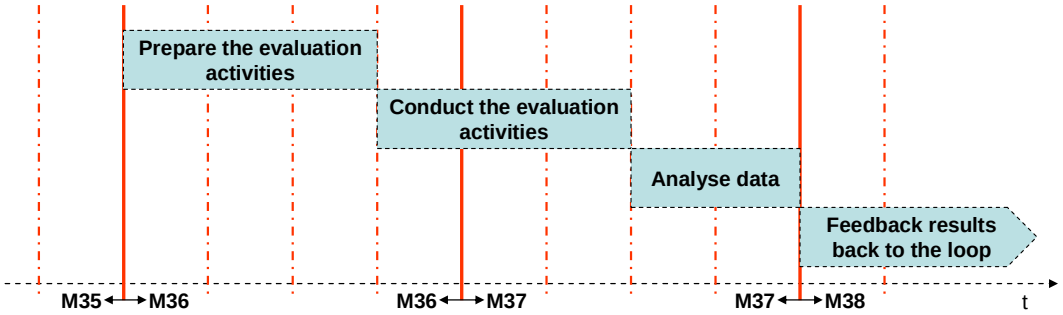

Figure 1: User validation second iteration - time plan

In the third validation cycle, the objects of the evaluation are the IDE and middleware prototypes. However, because some requirements, that resulted in being partly supported or not yet supported in the previous iterations, need to be validated again, the third iteration also considers the last version of the IDE and DDK. 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 who 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 who 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 successful commercialisation of the product.

Therefore, we differentiate between the term developer user from end user, 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	M50

Table 1: Validation plan milestones

As the actual object of the assessment is the middleware and the IDE, 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 by 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 criterion

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 criterion, which are considered relevant by the users. Deliverable 3.2 "Updated system requirements report" contains an updated overview of the requirements needed for the development of the Hydra system as they 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 Criterion:	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 third validation report follows the same schema defined in the previous reports on how to measure the fit criterion 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 D10.3 "Validation Report for DDK Prototype" to be used in the following validation cycles and, thus, simplify the evaluation effort and improve also the single requirement evaluation, in case some of them were not satisfying the threshold condition.

2.3 Requirements for the third iteration

Developer-users are interested in requirements fulfilment, the technical aspects related to the software instrument they want to use: a middleware, IDE, or another prototype. For this deliverable the validation is applied through requirements technical tests and assessment fulfilled at the end of the IDE 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 ones 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 within a defined time frame (i.e. 2 months) to allow feedback of the results 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 third iteration is presented in the next section, through tables divided depending on the particular WP. Similarly to the previous report about the second validation cycle (D10.3), in this validation report we define two tables for each WP:

- The first one collects all requirements that were not supported or partly supported after the second validation cycle; these requirements have been tested again on the current middleware and DDK, as well as tested for the first time (if applicable) on the IDE.
- 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 IDE.

3. Description of the validation methods

Once the validation testing procedures are depicted, the tester has to follow the indications given to perform the validation, which can be a laboratory test or a trial of the middleware/IDE. 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 conduction 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 IDE prototype assessment is performed in the same way as 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 criterion 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 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 No. 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 Criterion:	**9 out of 10** developers recognise the IDE as intuitive

3.1 WP3 – Evaluated requirements

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome 2nd Cycle			Outcome 3rd Cycle			
					Middle ware	SDK	DDK	Middle ware	SDK	DDK	IDE
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.	Partly supported	n.a.	n.a.	supported	supported	supported
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	Partly supported	supported	supported	supported	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.	Partly supported	n.a.	n.a.	supported	n.a.	n.a.	n.a.
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 interface or replaceable by 3rd-party modules with equivalent functionality.	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,	Partly supported	Partly supported	Partly supported	supported	supported	supported	supported

				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	n.a	supported	n.a.	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.	n.a.	supported	n.a.	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 Middleware.	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	supported	supported	supported	supported
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.	n.a	supported	n.a.	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.	n.a.	supported	n.a.	n.a.	n.a.

Table 2: WP3 requirements found not or partly supported in the 2nd cycle.

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome	
					Middleware	IDE
17	When applicable, middleware interfaces are exposed by WSA-compatible services	Web Service Architecture (WSA; http://www.w3.org/TR/ws-arch/) introduces a common definition of what a web service is and describes minimal characteristics of what is common to all web services. When web services are used in HYDRA, they should comply to WSA	In min. 90% of all cases, HYDRA web service interfaces are realised as WSA-compatible web services. In the remaining cases, web services use proprietary formats.	Assess the HYDRA Web Service implementation with regard to WSA compatibility.	supported	n.a.
19	Support of low-end devices	HYDRA must support low-end devices like RFID tags. Therefore, HYDRA must be compatible with at least 32-bit devices with < 512 KB RAM/FLASH or less. For smaller devices, HYDRA provides proxies.	Middleware is able to be installed and run on low-end 32-bit devices with 512 KB RAM/FLASH in 90% of all cases. . Proxies can be created to support more limited devices in 40% of all cases.	Deploy minimal part of the HYDRA middleware on required low-end devices.	supported	supported
21	HYDRA should be a service-oriented architecture	HYDRA should be a SOA per the Description of Work of the project	HYDRA is compatible to the SOA-definition by OASIS.	Evaluate HYDRA architecture against OASIS Reference Model for Service-Oriented Architecture	supported	n.a.
23	Configuration by open, human readable languages	Flexibility and robustness of the configuration process	95% of the middleware's functionality is configurable by open / human readable languages, for example by editing XML files.	Assess configuration process of each HYDRA manager.	supported	supported

241	Middleware should open source	We have stated in the DoW that we will produce open source software.	The core components of the Software are open source.	Core components are publicly available under an open source license.	supported	supported
324	Systems built using HYDRA should be scalable in terms of devices communicating	In large installations (such as in the apartment complex example) there will be many devices per apartment and a huge amount of embedded devices in total. HYDRA should support the development of such big systems.	The HYDRA middleware supports applications in which more than 100,000 devices exist.	Develop and run respective HYDRA applications that incorporate large numbers of devices.	supported	n.a.
535	Reduce number of rule engines	Several managers employ different rules engines. Identify all rule engines and the managers that use them. Investigate whether the existing rule engines can be conjoined into one single and common rule engine.	Survey of used rule engines and an assessment if they can be conjoined.	Identify all rule engines used by HYDRA managers and conduct an assessment if these can be conjoined	supported	n.a.

Table 3: WP3 selected requirements.

3.2 WP4 – Evaluated requirements

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome 2nd Cycle			Outcome 3rd Cycle			
					Middle ware	SDK	DDK	Middle ware	SDK	DDK	IDE
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	Partly supported	Partly supported	Partly supported	Partly supported	Partly supported	n.a.
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.	Supported	Supported	TBD	Supported	Supported	n.a	n.a
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.	Supported	Supported	TBD	Supported	Supported	Supported	Partly supported

Table 4: WP4 requirements found not or partly supported in the 2nd cycle.

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome	
					Middle ware	IDE
543	The Self-* Manager should exhibit good performance.	Performance is important as it is fundamentally concerned with efficient utilization of resources. It thus has effects as well for both usability and power consumption.	The Self-* Manager should have acceptable turnaround times for reconfiguration actions involving all components. The criterion for acceptable performance is inevitably application specific, but 5 seconds would be acceptable for interactive applications given the user does not interact directly with and is therefore not held up by self-management, but instead experience it as change in a system's qualitative properties, e.g. switching to a faster protocol for a given connection.	Measure performance on individual subcomponents of the Self-* Manager, to locate potential bottlenecks and enable computation of their combined performance, ie that of the Self-* Manager as a whole.	Supported	n.a.
544	The tracing tool should allow developers to inspect interactions among Hydra OSGi bundles	When developers set up rules that guide self-management of the middleware, they will need to understand the high-level interactions taking place among hydra components. A tracing tool that shows interactions among bundles as sequence diagrams can provide this understanding.	This requirement is functional and just concerns the ability for bundle interactions to be displayed in a reasonably useful manner. As such the requirement is satisfied if 3 developers are able to use the tool as intended. Specific requirements for usability is outside the scope of this requirement, but such qualitative requirements can arise from the test results arising from validating this requirement.	The tool has been testedby checking the ability for bundle interactions to be displayed in a reasonably useful manner.	Not supported	Partly supported

Table 5: WP4 selected requirements.

3.3 WP5 – Evaluated requirements

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome 2nd Cycle			Outcome 3rd Cycle			
					Middle ware	SDK	DDK	Middle ware	SDK	DDK	IDE
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	n.a	Supported	n.a	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	n.a	Supported	n.a	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 criterion 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.	n. a.	Not yet supported	n.a	n.a	n.a
506	It should be possible to lock	For many reasons it can be important to know	All write access is aborted if a	A Lock Manager will be implemented that	Not yet supported	n.a.	n.a.	Supported	n.a	n.a	n.a

	files (Storage Manager)	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.	file is locked.	provides the ability to get and release locks on entities like files and directories. Validation will be done by trying to sequentalize multiple access.							
504	It should be possible to add and remove physical storage 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.	n.a.	Supported	n.a	n.a	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.	n.a.	Supported	n.a	n.a	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.	n.a.	Supported	n.a	n.a	n.a
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 of these groups share the same credentials and can communicate isolated from non-group-members.	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.	n.a.	Not yet Supported	n.a	n.a	n.a

Table 6: WP5 requirements found not or partly supported in the 2nd cycle.

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome	
					Middle ware	IDE
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 standarized and be as modular as possible, so common parts can be reused by proxies for different wireless devices	30% of a proxy modules rely on common kernels.	Use of limbo tool in combination with OSGi framework as common kernel	supported	supported
487	Improve handshake protocol between Network Managers for exchanging certificates	current protocol is quite low level, just sending certificates to other partner, we should use s.th. like SSL protocol mechanisms, we have also to consider the other trust models like, Web of Trust and user interaction	In 95% of cases simple protocol would work	Implementation of improved handshake protocol to be included in the Network Manager distribution	supported	n.a
486	Hydra propietary 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	Deployment of private Hydra supernodes at CNET premises (Sweden)	supported	n.a
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	Configurable Security Manager	Supported	supported

Table 7: WP5 selected requirements.

3.4 WP6 – Evaluated requirements

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome 2nd Cycle			Outcome 3rd Cycle			
					Middle ware	SDK	DDK	Middle ware	SDK	DDK	IDE
104	Automatic Discovery of Services	It should be possible to configure the middleware to discover available services that meets defined criterion.	8 of 10 services are automatically discovered.	Enter new devices into a Hydra network, locally and remotely.	Partly supported	n/a	Partly supported	Partly supported	Partly supported	Partly supported	Partly 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.	Partly	n/a	Supported	n.a.	Supported	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 device types.	HYDRA device and service managers can be identified and selected through a software component ontology.	HYDRA device and service managers can be identified based on device discovery data and automatically included in a device proxy.	Partly	n/a	Partly	Supported	n/a	Supported	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.	Supported	Supported	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	n/a	Partly supported	n/a	n/a	Supported	Supported

Table 8: WP6 requirements found not or partly supported in the 2nd cycle.

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome			
					MW	SDK	DDK	IDE
92	Rule-based configuration of devices	The possibility for the developer to specify device behavior using rules. It should be possible to derive and re-use rules from pre-existing or generic rule sets for application domains	The functionality (services) of a device is accessible (by user or application) thru a rule-based interface.	Use SDK/DDK to write rule-based expressions over devices and their services. This can be done by using various policies, such as energy policies (expressed in XML/XSL-t) and access control policies (expressed in XACML). The Semantic Device construct also allows rules (in SPARQL or XSL-t) to be expressed over constituent devices.				Supported by the Semantic Device construct, and policy languages. Rule languages used are SPARQL, XSL-t, XACML.
94	Simulation environment	Use of a simulation environment is important for validating the rules/software interaction with devices. It can also be used for replaying the event log in order to examine unwanted system behaviour.	Simulation environment is available	Use SDK/DDK to write stubs for device based on the HYDRA class libraries and device descriptions in the device ontology.		Supported by the use of stubs and abstract device classes from the SDK and DDK class libraries.		
102	Device Ontology with user interface	Tool that allows browsing, searching, navigating device classes and their capabilities.	Tool for browsing device ontology exists	Browse device ontology using the administrators tool UI (Eclipse version)		Supported		
103	Automatic device ontology construction	The construction of a device ontology should be facilitated through finding and parsing product or device descriptions to annotate and produce ontology entries.	5 of 10 device descriptions can be successfully processed	Test the discovery process with a range of different UPnP based devices. Also use SAWSDL to annotate ay web service device.				Partly Supported. Device descriptions in UPnP can be parsed. SAWSDL can also be used for annotations of a device.
106	Persistent storage	Settings, configuration and other data should be persistently stored in the system.	Data can be persistently stored.	Write an application that uses the DAC and a set of Hydra devices. Retrieve device and application specific data between sessions. Use the Application Bindings				Supported by the DAC, The Hydra device descriptions (device XML) and by the Storage

				configurations file to define device specific persistent identifiers.				Manager.
119	Domain modelling support	The middleware and IDE should be able to interface with application domain frameworks representing core concepts and functions of specific application domains. These could in the most basic form be represented by UML Profiles, or domain ontologies.	The HYDRA IDE supports at min 2 defined domain modelling frameworks.	Use Ontology Manager interface (or tool) to define an OWL model of a selected domain, e.g., home device control.				Supported. The Ontology Manager can host domain specific models (expressed in RDF/OWL) as defined by developers (Application Ontologies).
124	Automatic downloadable updates over the Internet	The middleware and IDE should have automatic update facilities that allows downloading and installation of latest security and functional updates. This should be configurable.	Automatic updates works without disruption.	Deploy the system and provide a new version of a specific HYDRA manager and verify that the update is performed.				Supported in coming Open Source release.
248	Definition of Virtual Devices	In order to ensure flexibility, protecting weak devices and manage differentiated access to device and information, the developer or advanced users should be able to define virtual devices that replace/represent physical devices.	Separation of physical and logical device definition. A virtual device can fully replace a physical device	Use the DDK to HYDRA enable a specific physical device.				Supported by the basic HYDRA Device architecture and by several developer constructs, like semantic devices.
391	Device and service exception handling	The development and run-time environment should support exception handling constructs that the developer user can employ to manage service and device availability and malfunctioning, isolated from the main	SDK provides exception handling constructs that the developer can use to specify exception conditions.	Use the SDK to write an application that subscribes to events corresponding to exceptions.				Supported, by programming constructs in the IDE/SDK and by eventing.

		application logic.						
392	Rules for selection of alternative devices	The developer user should be able to specify how devices can replace or complement each other. This is relevant in situations when a device fail and another device exists which can provide a replacement service, or, when different levels of quality of service are available.	In the SDK, there are constructs available allowing the developer to specify rules for when and how devices and services/devices can be interchanged and combined.	Use the SDK to write an application that selects alternative device services depending if selected device is not available.			n/a	Supported, by programming constructs in the IDE/SDK, and thru semantic devices, i.e. aggregations of devices.

Table 9: WP6 selected requirements.

3.5 WP7 – Evaluated requirements

In WP7, no requirements found not or partly supported in the 2nd cycle. The table below reports the new tested requirements.

ID	Description	Rationale	Fit Criterion	Assessment procedure	Outcome	
					Middle ware	IDE
521	Linking Security Policy Language and Policy Manager to the Security Ontology	If Semantic Interoperability of security shall be made possible it is critical that policies can be linked and resolved to the meta-layer security objectives and assertions mapping the capability to the security objectives. This means that security policies can be made independent of specific security implementations.	It must be possible to express and resolve security policies linked to assertion providers evaluation of security capabilities linked to the meta-model security objectives as defined in the security ontology.	Test of SemanticPIP.	Supported	n.a.
507	Policy Editor for Access-Control Policies	The IDE must provide a tool to create Access-Control policies. Features like syntax-highlighting, on-the-fly-error creation etc. are optional.	An editor for the creation of Access-Control-policies exists in the Hydra IDE.	Test of Policy Editor.	n.a.	Supported
497	Analysis of conflicts between policy domains (dynamic analysis.	Conflicts may occur between different policy domains. Hydra should provide the developer with tools that reveal potential conflicts and their impacts	A tool exists that reveals potential cross-domain conflicts. A protocol and further mechanisms exist that resolve these conflicts if they occur.	Analysis of a prototypical implementation.	Partly Supported	
491	Authorisation based on semantic information	Policies regulating access control and authorisation should make use of semantic information about devices and users.	Semantic information and inferred knowledge is used for policy decisions.	Test of SemanticPIP.	Supported	
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	HYDRA middleware provides at least one mechanism enabling devices to determin the security level of an existing network.	Analysis of security ontology, Policy Framework and Secure Session Protocol.	Supported	

		provided security level the device can decide to enter the network or not.				
107	Tool for managing access rights of services	Tool that allows setting and managing access rights of services and resources.	Access rights can be configured and managed.	Test of Policy IDE.	n.a.	Supported
66	Access control for context data	Since the users don't want others to have full access to their data, context awareness control must be provided. For example there is no need for the technician to read the health related files of his customer.	It is possible to control the access to context data of a user either during runtime or when setting up the middleware.	Test of Policy Framework.	Supported	n.a.

Table 10: WP7 selected requirements.

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 out for each single requirement evaluated and grouped by 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 criterion:

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:

Supported. The HYDRA SDK, consisting of .NET and Java components is made available to developers by providing both, Eclipse and .NET integration. This provides a very convenient and efficient way for HYDRA application developers to utilize the complete set of available functionality. During training sessions held with 20 developers, all of them considered the SDK and the related IDE integration as intuitive and useful.

Req. ID: 41

Description:

Hydra Developer's Companion.

Fit criterion:

Complete documentation is available. It is at least 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 preparation for the training activities.

Description of the assessment result:

Supported. The Hydra developer’s companion consists of different parts. Each component consists of the following pieces:

- Code examples of how to use the respective component and how interplay with other components works.
- A how-to guide describing the usage of the component.
- A technical documentation describing the component’s functionality from a technical point of view.

Further, deliverable D3.9 – Final System Architecture Report provides an extensive description of the Hydra architecture, comprising issues like general design considerations, relations among

components, different architectural perspectives etc. Similar documents exist for the DDK (D5.10 – Wireless Network DDK Prototype), SKD (D5.7 – Wireless SDK Prototype) and IDE (D5.11 Wireless Network IDE Prototype).

Req. ID: **136**

Description:

Dynamic architecture.

Fit criterion:

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:

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 run inside an OSGi container, to allow dynamic reconfiguration of components. This has successfully been tested in the Hydra e-health demonstrator. Services can also be dynamically migrated, which is supported by the Network Manager and HIDs.

Req. ID: **199**

Description:

Modules should be extendable.

Fit criterion:

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:

Let developers implement extensions to the Hydra middleware and assess the result.

Description of the assessment result:

Supported. The Hydra SDK will be published under an open source licence, which guarantees 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. Further, as OSGi is component-oriented, 3rd-party code can be easily integrated by simply adding it to an OSGi configuration.

Req. ID: **207**

Description:

Service selection by context.

Fit criterion:

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:

Supported. This requirement has been validated within the context of a master thesis that takes the challenge of user-adapted suitable service selection in pervasive computing environments as a focus [Shi, 2009]. Based on the QoS and the Context Manager a selection framework has been developed, which matches user-side criterion against potential service properties. In three-stage experiments, the framework has been tested with 15 participants, and the selection performance has been validated in two given difficult selection situations using Paired T-Test.

Req. ID: **217**

Description:

The middleware should ensure high robustness of services.

Fit criterion:

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 based on this set of services and conduct a long-term operation stress test.

Description of the assessment result:

Supported. The Hydra energy efficiency demonstrator has been successfully tested during GSMA and CeBIT 2010. During CeBIT the demonstrator was running without breakdown of any services for 120 hours undergoing phases of very high stress during the day and no usage during the night. This has been a very good test in a near real-world scenario.

Req. ID: **234**

Description:

The middleware should be self-descriptive.

Fit criterion:

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:

Supported. Training sessions showed that the Hydra middleware is self-descriptive. Developers had a clear understanding of the concept of Hydra managers and OSGi bundles. During two-day training sessions held with 20 developers, all of them were able to explain the basic middleware concepts.

Req. ID: **320**

Description:

Separate domain-oriented services and user interface services architecturally.

Fit criterion:

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:

Supported. Hydra middleware services and application- or domain-oriented services are perfectly separated. Thanks to Hydra's service-oriented architecture, services developed for the Hydra demonstrators make use of middleware services with no strong coupling nor interweaving of both. As a middleware, Hydra does not provide any user interface services. User interfaces are highly domain dependent, thus, like all other domain-oriented services they are separated from the middleware services.

Req. ID: **335**

Description:

Location awareness / positioning support.

Fit criterion:

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:

Supported. The Context Manager provides rule-based functionality to deal with spatial information. Nevertheless, the implementation of concrete positioning systems is not part of the middleware as it can vary depending on application requirements, use cases etc. For example, the agriculture demonstrator makes use of RFID technology to implement location awareness. At the semantic level, the Ontology Manager provides means to annotate Hydra devices with location information. Like HYDRA-207, this requirement has been validated by implementing user-adapted suitable service selection in pervasive computing environments employing semantic context information [Shi, 2009].

Requirements for this cycle:

Req. ID: **17**

Description:

When applicable, middleware interfaces are exposed by WSA-compatible services

Fit criterion:

In min. 90% of all cases, HYDRA web service interfaces are realised as WSA-compatible web services. In the remaining cases, web services use proprietary formats.

Assessment procedure:

Assess the HYDRA Web Service implementation with regard to WSA compatibility.

Description of the assessment result:

Supported. The Hydra Web Service implementation has been successfully assessed towards WSA compliance. A detailed description of the assessment can be found in D3.9 Final System Architecture Report in Section 15 – Validation of the Hydra Software Architecture.

Req. ID: **19**

Description:

Support of low-end devices

Fit criterion:

Middleware is able to be installed and run on low-end 32-bit devices with 512 KB RAM/FLASH in 90% of all cases. . Proxies can be created to support more limited devices in 40% of all cases.

Assessment procedure:

Deploy minimal part of the HYDRA middleware on required low-end devices.

Description of the assessment result:

Supported. The Network Manager, as the minimal required HYDRA component currently requires 128 MB RAM, a 1GHz Processor, a JAVA Runtime Environment to run the OSGi framework and an IP stack implementation. The smallest devices that have been used with LIMBO in the current version are devices that offer the Connected Device Configuration required by JAVA ME. Such devices are comparable with mobile phones running a 16- or 32-Bit-Processor at 16 to 32 MHz with 512 KB ROM (256 MB RAM). For connecting more limited devices, the HYDRA proxy approach is provided.

Req. ID: **21**

Description:

HYDRA should be a service-oriented architecture

Fit criterion:

HYDRA is compatible to the SOA-definition by OASIS.

Assessment procedure:

Evaluate HYDRA architecture against OASIS Reference Model for Service-Oriented Architecture

Description of the assessment result:

Supported. Hydra has been successfully evaluated as adhering to the principles of SOA. A detailed description of the assessment can be found in D3.15 Final System Architecture Report in Section 15 – Validation of the Hydra Software Architecture.

Req. ID: **23**

Description:

Configuration by open, human readable languages

Fit criterion:

95% of the middleware's functionality is configurable by open / human readable languages, for example by editing XML files.

Assessment procedure:

Assess configuration process of each HYDRA manager.

Description of the assessment result:

Supported. HYDRA provides easy-to-use configuration tools inside the HYDRA IDE for Java and .NET development. Further, HYDRA allows for web-based configuration of managers. For first use, each manager provides a default configuration, which works out of the box. Nonetheless, developers can conveniently configure each manager by accessing the related web interfaces, which offer key/value-based configuration of all important features. This is even possible during runtime, without having to restart the environment.

Req. ID: **241**

Description:

Middleware should be open source

Fit criterion:

The core components of the Software are open source.

Assessment procedure:

Core components are publicly available under an open source licence.

Description of the assessment result:

Supported. Hydra is published open source under the LGPL v3 licence. All middleware components comply with this licence; they do not contain any 3rd party code that does not comply with it. Furthermore, all components are completely documented in the Hydra Wiki and provide usage examples and how-to guides (see HYDRA-41).

Req. ID: **324**

Description:

Systems built using HYDRA should be scalable in terms of devices communicating

Fit criterion:

The HYDRA middleware supports applications in which more than 100,000 devices exist.

Assessment procedure:

Develop and run respective HYDRA applications that incorporate large numbers of devices.

Description of the assessment result:

Partly supported. Demonstrators presented at the trade fairs GSMA and CeBIT cover areas such as home automation, e-health, and energy efficiency, and operate more than 10 devices at once. However, formal scalability tests with lots of more devices will have to be performed in the future.

Req. ID: **535**

Description:

Reduce number of rule engines

Fit criterion:

Survey of used rule engines and an assessment if they can be conjoined.

Assessment procedure:

Identify all rule engines used by HYDRA managers and conduct an assessment if these can be conjoined

Description of the assessment result:

Supported. In HYDRA several managers base on the utilization of rules but the processing of rules is encapsulated by the Ontology Manager and Context Manager. The Ontology Manager provides reasoning capabilities to the device and service managing software components: Network Manager, Application Device Manager, Orchestration Manager, Self-* Manager, and QoS Manager. However, the Context Manager uses the DROOLs rule engine (http://www.jboss.org/drools) in order to provide software developers with a reasoning means for a context model that is based on an attribute/value representation. This lightweight representation allows the creation of context-aware applications that omit the burden of processing real-time data with ontologies of the Ontology Manager, as this is decidedly expensive in terms of processing time. Since the Policy Framework utilises the Sun XACML implementation (http://sunxacml.sourceforge.net), which is rather a policy processor than a rule engine, the number of rule processing components of the Hydra middleware has been reduced to two.

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 of and reaction to 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.

No IDE support has been added for profiling devices in this cycle, as per the plan in the DoW, so the conformance with this requirement remains unchanged from the second cycle.

Req. ID: **317**

Description:

Support runtime reconfiguration. To supporting monitoring and leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime.

Fit criterion:

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

Assessment procedure:

The assessment in this cycle concerns IDE support for reconfiguration.

Description of the assessment result:

Supported, but implementation has only few features. The ASL interpreter component includes the capability of displaying a simple editor for asl scripts. It supports online execution of the script being edited, and auto-generation, e.g., of a script that yields the executing platform's current configuration, starting from a bare configuration containing only the core OSGi bundle and the ASL interpreter bundle.

Requirements for this cycle:

Req. ID: **544**

Description:

The tracing tool should allow developers to inspect interactions among Hydra OSGi bundles. When developers set up rules that guide self-management of the middleware, they will need to understand the high-level interactions taking place among Hydra components. A tracing tool that shows interactions among bundles as sequence diagrams can provide this understanding.

Assessment procedure:

This requirement is functional and just concerns the ability for bundle interactions to be displayed in a reasonably useful manner. Specific requirements for usability are outside the scope of this requirement.

Description of the assessment result:

Partly supported. The tool is capable of displaying live bundle interactions as they happen in a running system. However the tool still contains some bugs in the layout etc. before it is ready for user testing. Moreover, the middleware currently does not include a component for instrumentation of the OSGi platform to provide the event data required for visualization.

Req. ID: **543**

Description:

The Self-* Manager should exhibit good performance.

Fit criterion:

The Self-* Manager should have acceptable turnaround times for reconfiguration actions involving all components. The criteria for acceptable performance are inevitably application specific, but 5 seconds would be acceptable for interactive applications given the user does not interact directly with and is therefore not held up by self-management, but instead experience it as change in a system's qualitative properties, e.g. switching to a faster protocol for a given connection.

Assessment procedure:

Measure performance of individual subcomponents of the Self-* Manager, to locate potential bottlenecks and enable computation of their combined performance, i.e. that of the Self-* Manager as a whole.

An optimization cycle is adequate for validation as it involves all components in the Self-* Manager. The scenario starts when the reasoner detects a need for reconfiguration, and it involves the following components executing the following steps:

1. **Sensing.** AQL is used to retrieve the current system configuration. This is fed through the Event Manager to the Optimizer.
2. **Optimization.** The optimizer solves the optimization problem given by the state of the system and the current fitness function that reflects the current qualitative preferences for the system.
3. **Planning.** The result of optimization is a desired target configuration. The target configuration, along with the current system configuration from step 1 constitutes a planning problem.
4. **Actuation.** The result of planning is an ASL script transforming the system from its current configuration to the goal configuration. The result of planning is published by the planner through the Event Manager, and received by the ASL interpreter which executes it.

The combined execution time for an optimization cycle is thus:

$$T_{total} = T_{sense} + T_{optimize} + T_{plan} + T_{actuate} + 3T_{event}$$

Where the $3T_{event}$ is the time required to forward events among the components in between steps 1 to 4.

Description of the assessment result:

Supported. The following table lists the measured values for each of the components in the above formula:

Parameter	Value (ms)	Measurements	Devices
T_{sense}	66.9	100	4
T_{reason}	2058	55	-
$T_{optimize}$	1445	5	-
T_{plan}	7	20	-
$T_{actuate}$	100	50	-
T_{event}	20.6	100	4

Note that the value reported for T_{sense} includes two sequential event transmissions over the Event Manager, as part of the AQL protocol. The IPP planner was used in the experimental setup used to produce T_{plan}.

Adding the numbers in the table above yields a total time of 3739 ms; this is within the acceptable limit for interactive applications. It is clear from the results that improvement to this number can only be achieved by reducing the time needed for reasoning and optimization, as the time consumption of the other sub-components of the Self-* Manager are insignificant in comparison.

The numbers confirm the soundness of choosing the three layer architecture for the Self-* Manager. A key rationale in this architecture is that efficient algorithms should reside in lower layers. This is indeed the case as the components residing in the component control layer (sensing, actuation) performs significantly faster than the components in the above layer. The performance of the planning component appears good, however it should be noted that planning problems are inherently complex, so it is to be expected that when the planning problem instances increase in size, the performance of the planner will deteriorate significantly. Results of previous experiments show that the planning problems have to reach a size of more 70 architectural entities in the configurations before planning time exceeds 2000 ms. This confirms the allocation of the planning component to the top-layer, the goal management layer, along with the optimizer.

4.3 WP5 validation results

Requirements from the previous cycle:

Req. ID: **276**

Description:

New communication technologies

Fit criterion:

80% of new technologies are supported.

Assessment procedure:

Implement discovery and access modules for several communication technologies.

Description of the assessment result:

Supported. The discovery framework currently support Bluetooth, Zigbee, Xbee, Z-wave, IRDA, NEXA, UPnP and WiFi among others

Req. ID: **407**

Description:

Storage Manager - Gateways information stored synchronization

Fit criterion:

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:

Supported. The Storage Manager is able to synchronize the information stored in real time in the devices and the information stored (as cached information) in the gateways. This needs the support of timing information that is attached to the data itself.

Req. ID: **465**

Description:

Networks overlapping.

Fit criterion:

Device is not to be added to an existing Hydra network if it is unauthorised or if it belongs to another Hydra network, which is temporarily near other Hydra networks.

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.

Req. ID: **506**

Description:

It should be possible to lock files (Storage Manager)

Fit criterion:

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 sequentialize multiple accesses.

Description of the assessment result:

Supported. The Lock Manager is able to lock a file so no other client can access it until it is released by the entity that first blocked it. This prevents inconsistencies in the data stored in the files.

Req. ID: **504**

Description:

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

Fit criterion:

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

Assessment procedure:

Adding 10 devices to a striped and a mirrored storage and removing them.

Description of the assessment result:

Supported. The Storage Manager is in charge of enlarging striped devices in order to add new physical storage to them, by using distributed storage space in the cloud.

Req. ID: **503**

Description:

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

Fit criterion:

10% of the storage is striped or mirrored.

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:

Supported. Data managed by the Storage Manager is automatically mirrored or striped so that the information can be accessed and retrieved at any time. Synchronization of the date is also supported.

Req. ID: **502**

Description:

It should be possible to store simple key/value pairs

Fit criterion:

Storing and receiving cookies to a given Manager does not need more than 3 requests.

Assessment procedure:

Building a test application not sending more than 3 requests per access.

Description of the assessment result:

Supported. Communication between Network Managers proves that this requirement is met.

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 criterion:

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.

Requirements for this cycle:

Req. ID: **488**

Description:

Modular and standard device integration

Fit criterion:

30% of proxy modules rely on common kernels.

Assessment procedure:

Use of limbo tool in combination with OSGi framework as common kernel.

Description of the assessment result:

Supported. All Java proxies (70% of total proxies implemented) are being developed using Limbo tools, which now integrates the generated code for the service skeletons in the OSGi framework, using the common services deployed on them.

Req. ID: **487**

Description:

Improve handshake protocol between Network Managers for exchanging certificates

Fit criterion:

In 95% of cases simple protocol would work.

Assessment procedure:

Implementation of improved handshake protocol to be included in the Network Manager distribution.

Description of the assessment result:

Supported. Every communication between Network Managers do use certificates exchanged through the Hydra handshake protocol.

Req. ID: **486**

Description:

Hydra proprietary supernodes are needed to support D2D communication between networks

Fit criterion:

80% of the time, own supernodes are up and running.

Assessment procedure:

Deployment of private Hydra supernodes at CNET premises (Sweden)

Description of the assessment result:

Supported. CNET runs a Hydra supernode that can be used by all the Network Managers in the Hydra network.

Req. ID: **446**

Description:

Security parameters negotiation

Fit criterion:

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

Assessment procedure:

Configurable Security Manager

Description of the assessment result:

Supported. The Security Manager is configurable in real time by OSGi configurations that can be changed easily using web based configuration tools (part of the SDK)

4.4 WP6 validation results

Requirements from the previous cycle:

Req. ID: **104**

Description:

Automatic Discovery of Services.

Fit criterion:

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 HYDRA discovery model is based on the discovery of physical devices. The device services can be discovered in applications by using various search criterion, e.g., based on Quality of Services (QoS) properties recorded in the Device Ontology.

Req. ID: **114**

Description:

Semantic enabling of web services.

Fit criterion:

7 of 10 devices are semantically enabled.

Assessment procedure:

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

Description of the assessment result:

In HYDRA it is possible to associate semantic descriptions with device (web) services in the device ontology. These descriptions are associated with the device taxonomy that relates the device classes known to HYDRA. Devices are discovered in a three stage process, first physically by identifying it by protocol, e.g., ZigBee, and secondly, using UPnP to announce it in the local network, and then thirdly 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.

Req. ID: **117**

Description:

HYDRA component ontology.

Fit criterion:

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 based on device discovery data and automatically included in a device proxy.

Description of the assessment result:

The discovery process currently works with a software component model. This model represents the device managers and service managers that can be selected when generating a proxy for a newly discovered device. The device and service manager objects are available to developers 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: **122**

Description:

Configurable and easy to install middleware.

Fit criterion:

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 meets the installation time constraint.

Req. ID: **126**

Description:

Automatic Device ontology updates.

Fit criterion:

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 with devices. When a newly discovered device has been resolved against the device ontology (third discovery step) the Ontology Manager will create a corresponding runtime instance for the device.

Requirements for this cycle:

Req. ID: **92**

Description:

Rule-based configuration of devices

Fit criterion:

The functionality (services) of a device is accessible (by user or application) through a rule-based interface.

Assessment procedure:

Use the SDK/DDK to write rule-based expressions over devices and their services.

Description of the assessment result:

This requirement is supported by several facets of the Hydra IDE, including,

- the definition of various forms of policy e.g., relating to energy efficiency, and for security by means of access control policies. Rule languages used are XML/XSL-T and XACML.
- the Semantic Device construct, which can be used to define device aggregates based on rule-oriented expressions over the constituent Hydra devices. Rule languages used are SPARQL, XSLt.

Req. ID: **94**

Description:

Simulation environment

Fit criterion:

Simulation environment is available.

Assessment procedure:

Use SDK to write stubs for devices based on the HYDRA class libraries and device descriptions in the device ontology.

Description of the assessment result:

Several HYDRA applications have been developed combining device stubs with discovered HYDRA devices. This is supported by the use of stubs and abstract device classes from the SDK and DDK class libraries.

Req. ID: **102**

Description:

Device Ontology with user interface

Fit criterion:

Tool for browsing device ontology exists

Assessment procedure:

Browse device ontology using the administrators tool UI (Eclipse version)

Description of the assessment result:

The device ontology (user) interface provides graphical views of the device taxonomy, device instances and of the device properties. There is also a web service based interface which is used by Hydra Managers.

Req. ID: **103**

Description:

Automatic device ontology construction

Fit criterion:

5 of 10 device descriptions can be successfully processed

Assessment procedure:
Test the discovery process with a range of different UPnP based devices. Also use SAWSDL to annotate a web service device.
Description of the assessment result: This requirement is only partly supported in that there is no automatic translation of arbitrary device description formats. UPnP Devices with associated SCPD descriptions can be parsed.

Req. ID: **106**

Description:

Persistent storage

Fit criterion:

Data can be persistently stored

Assessment procedure:

Write an application that uses the DAC (Device Application Catalogue) and a set of Hydra devices. Retrieve device and application specific data between sessions. Use the Application Bindings configurations file to define device specific persistent identifiers.

Description of the assessment result:

Several applications have been designed that attach persistent device identifiers to HYDRA Devices via the DAC. An application, or a Hydra Device, can also use the Hydra Storage Manager for persistent storage.

Req. ID: **119**

Description:

Domain modelling support

Fit criterion:

The HYDRA IDE supports as a minimum 2 defined domain modelling frameworks.

Assessment procedure:

Use the Ontology Manager interface (or tool) to define an OWL model of a selected domain, e.g., for home device control.

Description of the assessment result:

This requirement can be viewed as supported in principle since the Ontology Manager can host any (domain) specific model. However, these models must be designed and expressed in RDF/OWL.

Req. ID: **124**

Description:

Automatic downloadable updates over the Internet

Fit criterion:

Automatic updates works without disruption.

Assessment procedure:

Deploy the HYDRA middleware and provide a new version of a specific HYDRA manager and verify that the update is effected in the installation.

Description of the assessment result:

This requirement is yet to be supported by the coming Open Source release of the HYDRA middleware.

Req. ID: **248**

Description:

Definition of Virtual Devices

Fit criterion:

Separation of physical and logical device definition. A virtual device can fully replace a physical device

Assessment procedure:

Use the DDK to HYDRA enable a specific physical device.

Description of the assessment result:

A large number physical devices have been HYDRA enabled demonstrating the "virtualization" capabilities of the SOA-based HYDRA Device architecture. Various developer constructs such as Semantic devices provide additional possibilities for device abstractions.

Req. ID: **391**

Description:

Device and service exception handling

Fit criterion:

SDK provides exception handling constructs that the developer can use to specify exception conditions.

Assessment procedure:

Use the SDK to write an application that subscribes to events corresponding to exceptions.

Description of the assessment result:

Several applications have been developed that combine the exception handling capabilities of the selected IDE platform programming language (e.g., C# in the HYDRA .Net IDE), with the event management provided for HYDRA applications. Code stubs for event management are made available through Hydra Application Project Templates in the HYDRA IDE.

Req. ID: **392**

Description:

Rules for selection of alternative devices

Fit criterion:

In the SDK, there are constructs available allowing the developer to specify rules for when and how devices and services/devices can be interchanged and combined.

Assessment procedure:

Use the SDK to write an application that selects alternative device services if selected device is not available.

Description of the assessment result:

Rules for alternative device/service selection can be expressed in the application program code. In addition to this, the Semantic Device construct was designed as a way to specify aggregations of HYDRA Devices, where the constituent devices can be specified explicitly or be inferred from rules.

4.5 WP7 validation results

Requirements for this cycle:

Req. ID: **521**

Description:

Linking Security Policy Language and Policy Manager to the Security Ontology

Fit criterion:

It must be possible to express and resolve security policies linked to assertion providers' evaluation of security capabilities linked to the meta-model security objectives as defined in the security ontology.

Assessment procedure:

Test of SemanticPIP.

Description of the assessment result:

Supported. The Policy Framework contains a SemanticPIP (Semantic Policy Information Point) which can be used to retrieve information about security capabilities from the security ontology. It is also possible to integrate other external assertion providers as PIPs.

Req. ID: **507**

Description:

Policy Editor for Access-Control Policies

Fit criterion:

An editor for the creation of Access-Control-policies exists in the Hydra IDE.

Assessment procedure:

Test of Policy Editor.

Description of the assessment result:

Supported. Policy Editor is integrated into IDE and it can be used to edit XACML policies.

Req. ID: **497**

Description:

Analysis of conflicts between policy domains

Fit criterion:

A tool exists that reveals potential cross-domain conflicts. A protocol and further mechanisms exist that resolve these conflicts if they occur.

Assessment procedure:

Analysis of a prototypical implementation.

Description of the assessment result:

Partly supported. Mechanisms for policy conflict resolution have been described and prototypically tested for a WWRF Paper. The tool is not yet fully integrated into Hydra middleware.

Req. ID: **491**

Description:

Authorisation based on semantic information

Fit criterion:

Semantic information and inferred knowledge is used for policy decisions.

Assessment procedure:

Test of SemanticPIP.

Description of the assessment result:

Supported. The Policy Framework contains a SemanticPIP (Semantic Policy Information Point) that can be used to retrieve information about security capabilities from the security ontology.

Req. ID: **308**

Description:

The Security Level of an existing network should be determinable

Fit criterion:

HYDRA middleware provides at least one mechanism enabling devices to determine the security level of an existing network.

Assessment procedure:

Analysis of security ontology, Policy Framework and Secure Session Protocol.

Description of the assessment result:

Supported. The security ontology can be used to describe the security level of a network. This information can be used before entering a network, e.g. in a policy or with a small extension to the Secure Session Protocol.

Req. ID: **107**

Description:

Tool for managing access rights of services

Fit criterion:

Access rights can be configured and managed.

Assessment procedure:

Test of Policy IDE.

Description of the assessment result:

Supported. A XACML policy editor has been implemented and tested as part of the Policy IDE.

Req. ID: **66**

Description:

Access control for context data

Fit criterion:

It is possible to control user access to context data either during runtime or when setting up the middleware.

Assessment procedure:

Test of Policy Framework.

Description of the assessment result:

Supported. Implemented and tested using a Policy Enforcement Point for the Context Manager.

4.6 Summary of the evaluated requirements

In the following table we summarise the results obtained for the validation of the selected requirements throughout the three validation cycles.

WP3		I	II	III
17	When applicable, middleware interfaces are exposed by WSA-compatible services			Supported
18	Support for different software architectural patterns	Supported		
19	Support of low-end devices			Supported
21	HYDRA should be a service-oriented architecture			Supported
23	Configuration by open, human readable languages			Supported
31	An easy-to-use programming framework should be provided	Not yet supported	Partly supported	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	Supported
136	Dynamic architecture	Not yet supported	Partly supported	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	Supported
207	Service selection by context	Partly supported	Partly supported	Supported
217	The middleware should ensure high robustness of services	Partly supported	Partly supported	Supported
234	The middleware should be self descriptive	Not yet supported	Not yet supported	Supported
241	Middleware should open source			Supported
320	Separate domain-oriented services and user interface services architecturally	Not yet supported	Not yet supported	Supported
324	Systems built using HYDRA should be scalable in terms of devices communicating			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	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	
535	Reduce number of rule engines			Supported

WP4				
312	Support profiling of devices' performance	Partly supported	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	Partly supported[1]
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	
543	The Self-* Manager should exhibit good performance			Supported
544	The tracing tool should allow developers to inspect interactions among Hydra OSGi bundles			Partly supported
WP5				
264	Common message protocol	Supported		
276	New communication technologies	Supported	Supported	Supported
336	Discovery protocol should support multiple networks	Supported		
396	Hydra-enabled devices – May be mobile or fixed equipment		Supported	
407	Storage Manager – Gateways information stored synchronization	Not yet supported	Not yet supported	Supported
419	Device services and resources provision through its Gateway	Supported		
425	D2D communication Overlay Hydra network	Supported		
427	D2D communication – Group management		Not yet supported	Not yet supported
442	Proxy – Gateways can filter and react to data received from associated non-hydra devices		Supported	
445	The level of protection should be independent from the currently used low-layer protocol	Supported		
446	Security parameters negotiation		Supported	
446	Security parameters negotiation			Supported
455	Identity - Update of the correspondences between identifier and physical addresses	Supported		
465	Networks overlapping	Not yet supported	Not yet supported	Not yet supported
475	Multimedia streaming in the Hydra network	Supported		
476	Network Manager Configuration and Testing	Supported		
486	Hydra proprietary supernodes are needed to support D2D communication between networks		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	
487	Improve handshake protocol between Network Managers for exchanging certificates			Supported
488	Modular and standard device integration		Supported	
488	Modular and standard device integration			Supported
502	It should be possible to store simple key/value pairs		Not yet supported	Supported
503	It should be possible to combine different storage for mirroring or striping		Partly supported	Supported
504	It should be possible to add and remove physical storage from a Mirror/Striping-Set		Not yet supported	Supported

[1] The assessment in this cycle concerns IDE only, and it resulted to be partly supported while in the DDK is still 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	Supported
WP6				
91	Any HYDRA device should have an associated description	Not yet supported	Supported	
92	Rule-based configuration of devices			Supported
94	Simulation environment			Supported
101	Manual device ontology definition	Supported		
102	Device Ontology with user interface			Supported
103	Automatic device ontology construction			Partly supported
104	Automatic Discovery of Services		Partly supported	Partly supported
106	Persistent storage			Supported
108	Device discovery	Supported		
110	Device Categorisation in runtime	Partly supported	Supported	
111	Dynamic Web Service Binding	Supported		
112	Dynamic Web Service Generation		Supported	
113	Composition (of services and devices)		Supported	
114	Semantic enabling of device web services	Not yet supported	Partly supported	
114	Semantic enabling of device web services		Partly supported	Supported
117	HYDRA component ontology		Partly supported	Supported
119	Domain modelling support			Supported
120	Multiple Device Virtualisations		Supported	
122	Configurable and easy to install middleware	Not yet supported	Not yet supported	
122	Configurable and easy to install middleware		Not yet supported	Supported
124	Automatic downloadable updates over the Internet			Supported
126	Automatic Device ontology updates		Partly supported	Supported
129	Support for Semantic Web Standards for Device Communication	Supported		
210	Middleware should support different architectural styles	Supported		
248	Definition of Virtual Devices			Supported
376	Security requirements must be part of the Hydra MDA	Not yet supported	Supported	
389	Service browsing in device ontology	Supported		
391	Device and service exception handling			Supported
392	Rules for selection of alternative devices			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				
66	Access control for context data			Supported
107	Tool for managing access rights of services			Supported
308	The Security Level of an existing network should be determinable	Not yet supported	Supported	
308	The Security Level of an existing network should be determinable			Supported
364	Hydra's Access-Control policies support credential based authentication		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		
491	Authorisation based on semantic information			Supported
497	Analysis of conflicts between policy domains (dynamic analysis.			Partly supported
498	Mechanisms used for communication security should be replaceable by configuration		Supported	
507	Policy Editor for Access-Control Policies			Supported
509	Enforcement of Access-control policies		Supported	
510	Enforcement of obligation policies		Supported	
521	Linking Security Policy Language and Policy Manager to the Security Ontology			Supported

Table 11 - Summary of evaluation results

From the table it is possible to sketch the graphics of the success rate for the actual validation, in terms of requirement percentages reaching the threshold.

On average, 97% of the tested requirements have been partly or completely covered, as it appears in Figure 2 below.

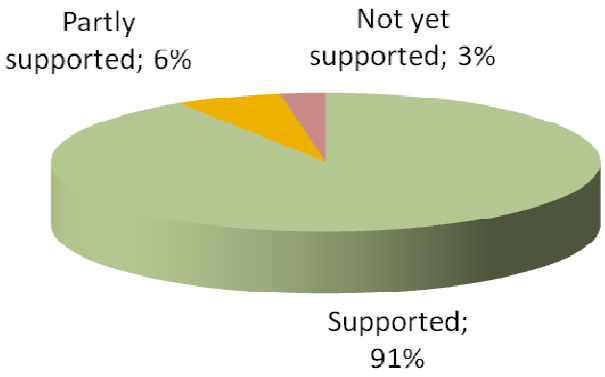

Figure 2 - Overall success percentages after 3rd validation cycle

In the next summarising graphs we present the results obtained per WP. The indication is not relevant in terms of quantitative aspects, but it emphasises the achievement of excellent results in all WPs.

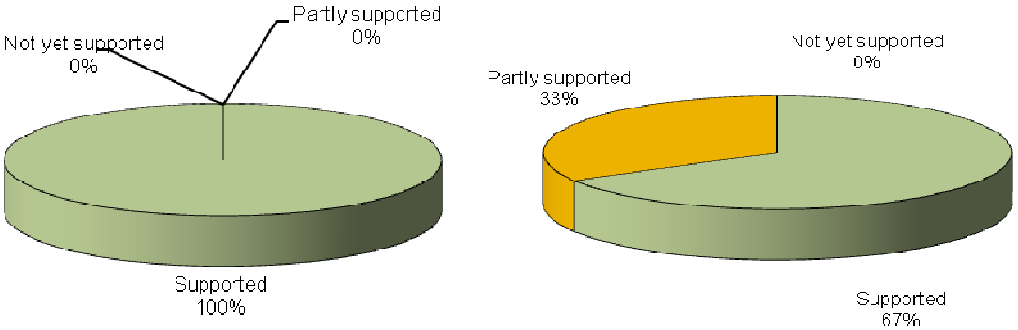

Figure 3 - Requirements fulfilment for WP3 (left) and WP4 (right)

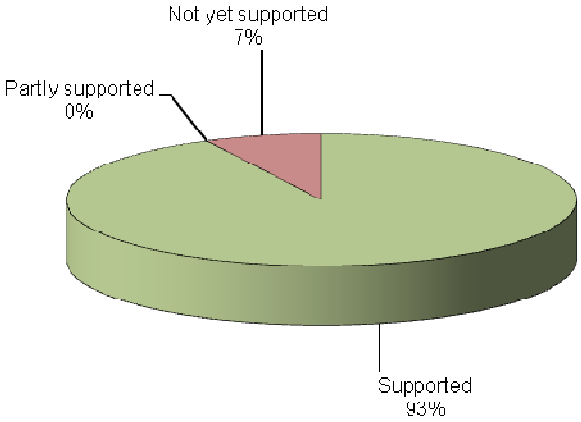

Figure 4 - Requirements fulfilment for WP5

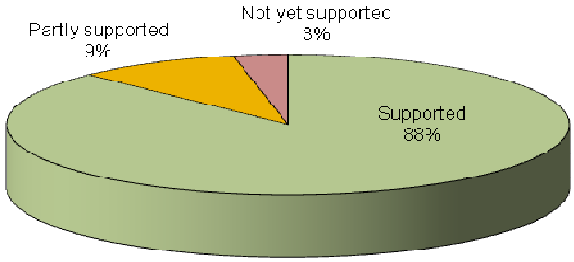

Figure 5 - Requirements fulfilment for WP6

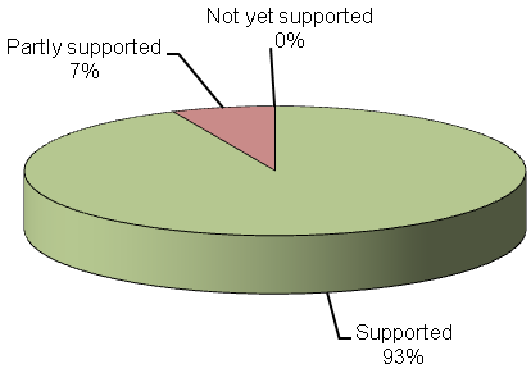

Figure 6 - Requirements fulfilment for WP7

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 by 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 included. The environment selected for the validation was the software laboratory of the Hydra partners, where potential developer users, not previously working with the Hydra implementation, were selected to carry out the assessment.

In more detail, the validation methodology consisted in the verification that each selected requirement fit criterion has reached the threshold level, or whether the requirement has been partly 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 three validation cycles, 112 requirements have been assessed. The overall results are summarised in the following table.

Assessment threshold level	No. of requirements fulfilling the threshold
Supported	102 (91%)
Partly supported	7 (6%)
Not yet supported	3 (3%)

Table 12: Overall success rate.

Note that the number of supported requirements is increased compared to previous cycles (it was 52% at the end of the first iteration and 70% at the end of the second iteration). Both the number of requirements "not yet supported" and "partly supported" have drastically decreased:

- "not yet supported" were 31% and 12% at the end of the previous cycles(first and second, respectively).
- "partly supported" were 17% and 16% at the end of the previous cycles(first and second, respectively).

Specifically, in the last validation cycle, in total 54 requirements have been assessed:

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

Focusing on the re-assessed requirements only, we can state that 19 requirements out of 24 (79%) moved from not yet supported to supported, or moved from partly supported to supported; i.e. we had a substantial improvement in the development of the software tools (SDK and DDK) and middleware in the last year, towards an almost final release of the Hydra technologies.

Focusing on the newly assessed requirements only (IDE and middleware), the results are summarised in the following table.

Assessment threshold level	No. of requirements fulfilling the threshold
Supported	27 (90%)
Partly supported	3 (10%)
Not yet supported	0 (0%)

Table 13: New requirements success rate.

Note that the success rate for the new requirements of this third validation cycle has improved compared to the success rate of the second validation cycle (it was 69%).

These validation outcomes clearly show that the Hydra platform implementation has achieved almost all of the target objectives. Following the positive results of the previous validation cycle, this validation cycle confirmed and actually improved the obtained results. Indeed, the acquired know-how of researchers and developers about the involved technologies and features of the platform helped the achievement of these improved results.

The data emerged in the present analysis and, thus, the user feedback, will be distributed to the Hydra consortium (starting from each technical Work Package, but also looped back to WP2) for deploying the final software releases of the project and detailing the project lessons learnt.

6. References

[Shi 2009] Lei Shi: Semantic Web Service Selection: Supporting Quality of Service and Context Awareness, Master Thesis, Bonn, 2009