Contract No. IST 2005-034891

Hydra

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

D2.7 Updated Systems Requirements Report

**Integrated Project
SO 2.5.3 Embedded systems**

Project start date: 1st July 2006 **Duration: 48 months**

**Published by the Hydra Consortium
Coordinating Partner: Fraunhofer FIT**

December, 31st 2009 - version 3.0

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

Dissemination Level: Confidential

Document file: D2.7 Updated Systems Requirements Report v3.0.doc

Work package: WP2 Iterative user requirements engineering

Task: T2.3 Evolutionary requirements refinement

Document owner: FIT

Document history:

Version	Author(s)	Date	Changes made
3.0	Andreas Zimmermann (FIT) Marco Jahn (FIT)	15-12-2009	Initial version based on D2.7 v2.0
3.1	Andreas Zimmermann (FIT) Marco Jahn (FIT)	31.12.2009	Changes based on internal reviews

Internal review history:

Reviewed by	Date	Comments
Jesper Thestrup	26-12-2009	Minor corrections and format changes
Pablo Antolin Rafael	22-12-2009	Minor changes

1. Introduction

This document presents the third update of Hydra's system requirements, and thus, represents the third version of the D2.7 Updated Requirements Report.

The deliverable gives an overview of the refined set of requirements that will be used within the remaining iterative steps to assure a user-centred approach and methodology in all phases of the project. It takes into account all requirements that have been created or updated since January 2009 and provides an analysis of the impact that these updates have on each work package. Thereby, it gives an outlook on the activities that have to be performed in the next iteration coming in 2010.

1.1 Structure of this document

The report is structured as follows: Chapter 2 gives an overview of the work described in this document and summarizes the effects on the architecture and work packages. Chapter 3 presents all new and updated requirements since January 2009. In chapter 4, the impact on each work package is described in detail. Finally, chapter 5 gives a brief summary, and chapter 6 provides the complete list of open requirements.

2. Executive Summary

During the last two iterations, requirements have been created and constantly updated. The main activity for verifying and refining requirements has been the reporting of the lessons learned resulting from cycle 3 (see D2.11 Change request and re-engineering report from cycle 3). According to the elicitation process of the requirements (see D2.5 Initial requirements report), the creation of a new or refinement of an existing requirement is tracked in Jira. Thus, the Jira system has been the main source of input for collecting new, updated and rejected requirements and for reviewing, which requirements have been implemented during the last iteration.

Many requirements defined from the lessons learned of iteration 3 have been implemented. Also older requirements have been implemented. The amount of new requirements is decreasing and updated and new requirements focus on refinement and redesign.

Work package 3 implemented 10 and created one new requirement. The requirements analysis shows that the overall system architecture meets the initial requirements. Work package 4 implemented two and created one new requirement. Besides, the descriptions of two requirements have been updated and one more has passed the quality check. In Work package 5 two requirements have passed quality checks and one has been rejected. 11 requirements have been implemented in WP6, one has been created and two have been approved to be part of specification. Five requirements need to be revised as they have been marked as not making sense. WP7 has also implemented 11 requirements and one has been added to the specification.

For the next iteration all work packages will focus on the final integration of all Hydra managers and components. Also IDE integration of Hydra and especially the DDK and SDK will be a great part of the last development cycle.

3. Updated requirements for Hydra

This section contains the condensed list of functional and non-functional developer-user requirements that have been updated, deleted, or created since January 2009.

Each requirement listed in the following tables has a unique ID that allows referencing. The description of a requirement is a synthetic but clear description of the requirement. The rationale gives a reason why this requirement is relevant for the HYDRA system and thus has been included into the table. The column “source” gives an indication of where this requirement has been created, i.e. scenario, interview, focus groups, or lessons learned. According to the Volere scheme the requirements are divided into non-functional and functional requirements.

3.1 Requirements of WP3 - Architecture Design Specification

3.1.1 Architecture

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
535	Non-Functional	Major	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.	Architecture	Open
528	Functional	Major	Specification of the information flow among Hydra Managers.	During software integration of the first year prototype some problems were attributed to the event management, which has been overly used. The application of JAX-WS and Axis for event-driven application worked fine, although some latency has been identified due to multiple concurrent function calls. In addition, the use of web applications as Event Manager in the role of both publisher and consumer works fine. However, the development of web applications for small devices such as PDAs limits the usage of HTML, JavaScript and CSS.	Complete specification that clearly defines how the information shall flow among Hydra Managers.	Architecture	Implemented
527	Functional	Major	Extend Context Manager for Semantic Data, in addition to raw data storage.	The context manager was proposed to store raw context data, but from our point of view, additionally some semantic data will have to be stored about these data.	Besides raw context data, the Context Manager must provide a mechanism to store semantic data.	Architecture Context	Implemented
526	Functional	Major	Delineation between middleware and application in terms of context provision.	The Context Manager is mainly connected to the application itself not to the middleware (as agreed in discussion with the partners); it was withdrawn from the scope of the ontology manager.	In terms of context provision middleware and application itself must be delineated.	Architecture Context	Implemented

525	Functional	Major	Delimitation between Application and Device Elements.	In the first two cycles we found that we need clarification on the delimitation between application and device elements. The delimitation between Application and Device Elements seems to blur.	No interdependencies between Application and Device Elements.	Architecture	Implemented
524	Functional	Major	Determination and Description of the dependencies among Hydra Managers.	Some core managers exhibit a type of predefined collaboration between them; others offer their functionality to all components of the entire Hydra software architecture. Managers of the first group actually demand direct inter-manager calls or a refactoring of the software architecture focussing on the fusion of functionality. Managers of this second group provide functionality to all managers of the other group. Therefore, the managers of the second group offer functionality that runs orthogonally with respect to the basis functionality. In addition, this orthogonal functionality cannot be separated from the existing components.	The dependencies of all Hydra Managers must be determined clearly and described in detail.	Architecture	Implemented
522	Functional	Major	All HYDRA entities must have a semantic model description	If interoperability and security is to be possible, an entity must have a semantic model description. Otherwise other devices are not able to discover if they can communicate with the device or if the device security can be resolved according to the security policy. Devices or applications that are unable to present a semantic model description cannot be expected to be able to pass a security resolution according to security policies.	A hydra-enabled entity must have a semantic model description	Architecture	Implemented
329	Non-Functional - maintainability	Major	Middleware provides domain-independent services	A lot of the services needed in the apartment scenario are also needed in other scenarios (persistence, logging, visualization, etc.). These should be abstracted and built and provided as part of HYDRA	Large parts of the building-automation scenario can be built by reusing configurable services from across other application domains.	Architecture	Implemented
327	Non-Functional - performance	Critical	The HYDRA middleware should be flexible as to	Not all parts of HYDRA will make sense in all situations (it will not always be beneficial to use higher layers of communication such as a service composition protocol or maybe a	HYDRA is able to support the exact subset of services required by a client (user or service) in	Architecture	Implemented

			allow for opt-in and opt-out on parts	device may be too resource-constrained to use parts). One should be able to take the parts of the HYDRA middleware that makes sense for a certain application. For example, it should be possible to for embedded devices with few resources (see other requirements) to take part in a HYDRA application without having to install or run all HYDRA components. Another example may be that one may want to use just point-to-point communication of HYDRA without having to use the context-awareness part. (Werner Vogels, CTO/Amazon at JAOO 2006: "Middleware is evil!", referring to that if one chooses a certain middleware such as CORBA one makes too many decisions (not only on communication in the CORBA case but also, e.g., on transactions) that may not be appropriate for the case at hand)	70 % of all cases. In 20 % of all cases the middleware is able to provide a service package that includes the required service. In 10% of all cases HYDRA is not able to provide service similar to the desired service.		
18	Non-Functional - usability	Major	Support for different software architectural patterns	The HYDRA architecture should not prescribe one way to structure applications. Thus several architectural patterns, for example MVC and PACshould be supported.	HYDRA allows at least two different architectural patterns for applications.	Architecture	Implemented

3.1.2 Devices

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
33	Functional	Critical	Enable manufacturers to develop devices and applications that can be connected to HYDRA	The hydra SDK should provide the manufacturers with an API to develop devices that can be connected to the hydra network.	APIs are available to develop devices that can be connected to the hydra network	Devices IDE SDK	Implemented

3.2 Requirements of WP4 - Embedded AmI Architecture

3.2.1 Architecture

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
317	Functional	Major	Support runtime reconfiguration	To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime To ensure a conceptual integrity of the system and ease developer understanding, the tools for initial configuration and re-configuration should rely on the same concepts/mechanisms.	Services and devices can be connected in new ways during runtime in HYDRA-based applications	Architecture	Reopened

3.2.2 Communication

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
479	Functional	Major	The EventManager should support event prioritisation	The EventManager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system	Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort	Communication	Quality Check passed
368	Functional	Minor	Support of UDP and TCP protocols	Depending on the situation, the device developer can choose whether the WS communication runs on top of TCP or of UDP. Tools will be provided	60% of Hydra proxies are implemented selecting TCP or UDP as transport mechanism	Communication	Implemented

3.2.3 Configurability

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
334	Functional	Major	There should be support for developing auto-	A number of use scenarios calls for the ability to bring a device home, turn it on, and have it function reasonably	The middleware supports defining auto-configuration properties and using these at runtime.	Configurability	Implemented

			configuration of certain devices		This is not in conflict with security		

3.2.4 Networking

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
533	Functional	Major	Protocols should be changeable at runtime	Different protocols have different functional and non-functional properties. A self-optimizing system needs to be able to realize an optimal configuration of protocols, as guided by e.g. QoS properties such as for energy awareness or security.	It should be possible to realize a scenario involving protocol change.	Networking	Open

3.2.5 IDE

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
532	Functional	Major	Support autogeneration of eclipse project files	We support using a range of protocols: UDP, TCP, and Bluetooth. Each can be used on JSE standard, JSE on OSGi, JME; For web-services they can use REST or SOAP; This gives a total number of combinations: 3x3x2 resulting in 18 projects if we maintain one for each combination. Thus the range of potential combinations quickly becomes too large to support manually.	Manual labour required to use specific combinations of platforms, protocols etc should remain constant rather than proportional to the number of combinations of these	IDE	Reopened

3.2.6 Interface

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
530	Functional	Trivial	Domain-based Properties available for Querying	For invoking the reasoning in Hydra ontology in order to retrieve QoS property data values.	For QoS Manager it is necessary to be able retrieve current QoS property data values from Ontology Manager in order to process semantic service selection.	Interface	Open

3.3 Requirements of WP5 - Wireless Networks and Devices

3.3.1 Architecture

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
503	Functional	Major	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.	Replicated and Striped devices can be built up on each other.	Architecture	Part of specification

3.3.2 Communication

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
505	Functional	Minor	It should be possible to access data in Storage Manager using a well defined protocol, e.g. WebDav	Using external Applications it should also be possible to access data without to much pain. Exporting storage using WebDav gives the User the ability to access it as network devices on most operating systems.	50% of the storage can be accessed by non hydra applications.	Communication	Part of specification

3.3.3 Networking

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
523	Functional	Critical	Addressing without linking	JXTA is based on addressing using a persistent identifier. This mean JXTA represent a problem in the network layer. Instead JX can be used in the virtual middleware layer using a HID-specific identifier or as an application layer addressing mechanism. HYDRA needs to reconsider addressing mechanisms across locations - probably including addressing mechanisms in the ontology support to make addressing schemes interoperable. A solution likely involve mechanisms	No use of persistent device identifiers in the network layer or virtual layer.	Networking	Rejected

				to move e.g. IPv6 addressing to the application layer.			

3.4 Requirements of WP6 - SOA and MDA Middleware

3.4.1 Architecture

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
534	Non-Functional - performance	Major	There should be one set of semantic APIs for all of Hydra	wp4:ll5, third cycle, as reported in D2.11	There should only be one in-memory copy of semantic information per device, and all managers relying on ontologies should use that.	Architecture	Open
112	Functional	Major	Dynamic Web Service Generation	Configuration tool that is able to generate the necessary interfaces to wrap the device functionality as a web service.	7 of 10 device functionalities are automatically represented as web services	Architecture IDE	Implemented

3.4.2 Configurability

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
470	Functional	Major	Device Ontology	Knowing a priori if a device could support a data format could help the application developer to better exploit the resources of the device	The device information could host a field where is pointed out the proper data format	Configurability	Requirement does not make sense
393	Functional	Major	Deployment scenario configurable by developer user	A developer user should be able to specify how an application should be deployed over a set of devices, e.g, choosing a host device for a Device Application Catalogue.	Developer can specify deployment for specific devices by means of a tool or configuration file.	Configurability SDK	Requirement does not make sense

3.4.3 Device Discovery / Devices

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
501	Functional	Critical	A Hydra enabled device must support UPnP discovery	UPnP has been proven as a well-functioning network discovery mechanism i HYDRA.	All HYDRA enables devices support UPnP	Device Discovery Middleware Layer	Implemented

						Networking	
500	Functional	Major	Semantic annotations of devices using SAWSDL	Device developers should via the DDK be able to produce (SAWSDL) annotations for devices, in order to facilitate device discovery and ontology update.	For a given UPnP discoverable device, it is possible to create an SAWSDL annotation which can be accessed from the UPnP discovery information.	Device Discovery Devices	Implemented
392	Functional	Major	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 fails and another device exists which can provide a replacement service, or, when different levels of quality of service are available.	In the SDK, constructs are available that allow the developer to specify rules for when and how devices and services can be interchanged and combined.	Device Discovery SDK	Part of specification
110	Functional	Major	Device Categorisation in runtime	Middleware should after discovery of device be able to categorise a device based on device ontology information.	7 of 10 devices are correctly categorised and described.	Device Discovery Middleware Layer	Implemented
91	Functional	Major	Any HYDRA device should have an associated description	For management, search and discovery purposes, all HYDRA enabled devices should be described (classified) according to the HYDRA device ontology.	Any device associated to a HYDRA application is also included in the HYDRA device ontology, and its description can be retrieved.	Device Discovery Devices	Implemented
492	Non-Functional - performance	Minor	Semantic grouping for Non Hydra Enabled devices	Hydra could manage not only the single device services at once but a group of them if they are in the same semantic context in order to reach higher performances and low energy consumption (even if the hydra middleware doesn't address this topic) limiting the communication time per single device.	90% of non-hydra enabled devices with the same functionality in the same context has to implement similar functions (e.g. user detection in a room will turn on all lights)	Devices	Reopened

3.4.4 IDE

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
212	Non-Functional	Major	Support for a declarative application development paradigm	A declarative approach can hide complexity of underlying structure and can increase productivity of embedded software development.	More than 50% of the module functionality should be programmable using a declarative approach.	IDE	Requirement does not make sense

121	Functional	Major	Optimised device ontology	It should be possible to optimise the device ontology for instance by deploying a subpart of it to be used in device discovery process.	Possible to select and extract subparts of the device ontology	IDE	Requirement does not make sense
113	Functional	Major	Composition (of services and devices)	In order to enhance or replace application level functions it will be useful to be able to compose services and devices from different providers and/or manufacturers into high level services/devices	Service composition during design-time is possible.	IDE Middleware Layer Service Discovery	Implemented
102	Functional	Major	Device Ontology with user interface	Tool that allows browsing, searching, navigating device classes and their capabilities.	Tool for browsing device ontology exists	IDE	Part of specification

3.4.5 Middleware Layer

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
376	Functional	Major	Security requirements must be part of the Hydra MDA	Security must be defined to be resolved semantically	Security model can be defined semantically	Middleware Layer	Implemented
120	Functional	Major	Multiple Device Virtualisations	It should be possible to have several different views/virtualisations of a device depending on context and applications.	At least 2 different virtualisations are provided	Middleware Layer	Implemented
115	Non-Functional - operational	Major	Decomposable middleware	Middleware must consist of decomposable components to allow different deployments depending on available performance restrictions.	It is possible to deploy middleware on at least 3 different platforms.	Middleware Layer	Implemented
114	Functional	Major	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.	Middleware Layer	Implemented

3.4.6 Security

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
477	Functional	Major	Device proxies should make use	If non-Hydra-enabled devices are communicate to the Hydra network by a proxy, security features of the protocol supported	Device proxies must support WEP and WPA for Wi-Fi-	Security	Implemented

			of available security features for "last mile" communication	by the device should be used.	connections as well as Bluetooth authentication and encryption		

3.4.7 Service Discovery

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
499	Non-Functional - usability	Minor	Services common naming convention	Using a common naming convention, for the same services, to give uniformity in interaction procedures. This convention should be extended to input/output services data format. This should provide a more efficient group creation, a simple internationalisation and localisation processes.	When at least the 95% of services with the same purpose, realised by means of different solutions (sensor for a ZigBee node, a Web Service connection for the weather monitoring), respect the same naming convention.	Service Discovery	Reopened

3.5 Requirements of WP7 – Trust, Privacy and Security

3.5.1 Architecture

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
510	Functional	Major	Enforcement of obligation policies	Security obligation policies dictate certain actions that have to taken upon occurrence of an event trigger. Components that are part of a policy domain must negotiate on the action they can enforce and must provide the respective enforcement mechanism	Hydra components negotiate their capability to enforce different actions with the policy decision point and provide an enforcement mechanism for at least one action type.	Architecture Security	Implemented
509	Functional	Major	Enforcement of Access-control policies	Access control decisions must be enforced.	Policy enforcement points can be attached to Hydra web services so that access control decisions can be enforced.	Architecture Security	Implemented

508	Functional	Major	Storage for security policies	Access-control policies and obligation policies need to be kept in a repository that is available to Policy Decision Points and administrative tools. Access to the repository should be regulated so that no authorised changes to the policies are possible.	A repository for storing access-control and obligation policies exist and access to that repository can be controlled.	Architecture Security	Implemented
498	Functional	Major	Mechanisms used for communication security should be replaceable by configuration	Cryptographic algorithms, protocols and authentication mechanism might become insecure after a Hydra-based application has been deployed. In that case, it should be possible to exchange security modules without having to recompile/deploy the middleware	For at least two of the communication protection mechanisms (Core / Inside / Outside Hydra) it should be possible to replace security modules without recompiling the middleware.	Architecture Communicatio n Security	Implemented
496	Functional	Major	Conflict resolution between policy domains	Devices may be subject to different policy domains. If communication between these domains is to be established, conflicts between the different policy domains may occur. Hydra needs to provide mechanisms and protocols to resolve these conflicts.	Conflicts between policy domains are recognized and can be handled.	Architecture Security	Part of specification
493	Functional	Major	Obligation Policies for Security	Obligation policies trigger certain actions upon certain events. Hydra needs to provide such obligation policies for security reasons, e.g. to force devices to update their security modules at runtime or to change configurations. These policies would be triggered by user interactions, context changes or any other event.	A mechanism for obligation policies exists that allows to specify security-related actions depending on situations and events.	Architecture Configurability Security	Implemented
50	Functional	Major	An identity management must be provided	HYDRA middleware has to provide highly sophisticated mechanisms for identity management in order to ensure that in systems featuring HYDRA only authorised access to data, applications and devices is possible.	Identity management mechanisms are provided at all levels and to all stakeholders. Furthermore, the identification process of the managers must be uniform and standardised.	Architecture Security	Reopened

3.5.2 Communication

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
364	Functional	Major	Hydra's Access-Control policies support credential based authentication	Instead of identifying the user or device, a session may be authenticated through credentials recognised by the application such as blinded certificates, direct anonymous attestation, previously agreed tickets, reuse of previous accepted keys (e.g., PGP keys). That means the network can operate with authentication schemes using credentials without having to identify the device and/or user. The point is that identification of people or devices MUST NOT be MANDATORY. Alternative mechanisms such as credential based authentication MUST be ALLOWED. Example: In Smart Home when a Service Agent of a Service Provider needs access to the home - instead of a door identifying the person or the device from the service agent, the Home Owner/Home System provide the Service Provider with a one-time-only token that the Service Provider is accountable for. The Service Provider can then forward this to the Service Agent who presents the token to the Home Access Control System. The Home Access Control System can accept the token as is or in real time contact the Home Owner and/or Service Provider System when the Service Agent is at the door. The System doesn't need to create the risk of identity theft by identifying the Service Agent person or device. He can use a device that create a random handle and communicate without further security requirements even though the system only has a credential proving traceability to the Service Provider.	Access-control can be based on credentials	Communication Security	Implemented
49	Functional	Major	Mechanisms for verifying the authenticity of a communication partner	For critical communication the authenticity of the communication partners has to be ensured.	Mechanisms enabling mutual authentication have to be provided. Especially HYDRA enabled devices have to support authentication mechanisms.	Communication Security	Implemented

3.5.3 IDE

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
494	Functional	Major	Static analysis of semantic policies	In order to support developers in designing security policies, the Hydra IDE needs to provide analysis techniques revealing the possible impact of the policy.	An analysis mechanism exists that tests "semantic" policies for correctness, for possible impacts and conflicts	IDE Modelling Security	Implemented

3.5.4 Security

ID	Type	Priority	Description	Rationale	Fit Criteria	Component/s	Status
512	Non-Functional - security	Major	Policy decision and enforcement on embedded / mobile devices	In many cases, security policies (access-control as well as obligation) have to be enforced on resource-restricted platforms. Further, as those platforms might represent a "policy domain" by themselves, policy decisions also have to be made on that platform. It has to be evaluated whether it is possible to port existing decision mechanisms to mobile devices. If not, a dedicated solution for resource-restricted platforms has to be found.	Security policies can be decided and enforced on resource-restricted platforms, e.g. a smartphone.	Security	Reopened
471	Functional	Major	End-to-end message protection at application level must be supported	Confidentiality and non-repudiation can only be guaranteed if messages are end-to-end protected and can't be altered during communication	The middleware provides mechanism that can be used by application developers to implement end-to-end message protection of application-specific data.	Security	Implemented
79	Functional	Major	Secure cryptographic key management	As a large variety of keys will be used, for authentication, encryption, access control etc., a secure key management is needed.	A secure cryptographic key management is provided.	Security	Implemented
45	Functional	Major	Stored data must be protectable	Any stored data must be protectable from unauthorised access. This can be done by access-control, encryption, context isolation or a combination.	HYDRA provides developers at least one mechanism to protect any stored data from unauthorised access.	Security	Implemented

4. Impact on the Work Packages

This section summarizes the impact of newly created and changed requirements on future work conducted in the technical work packages.

4.1 Impact on WP3

4.1.1 Architecture

During the third project iteration, 10 requirements regarding the Hydra architecture have been implemented.

Four of them deal with conceptual decisions regarding dependencies and workflow among managers, and specifications of certain managers. Dependencies turned out to be too strong and needed a restructuring. Also the delineation between application and device elements had to be reconsidered. These issues have been solved in the last iteration and are covered in D3.9 Updated System Architecture Report (528, 527, 526, 525, 524).

Further progress has been made with regard to the DDK. Requirement 522 states that every entity has to provide a semantic model description, in order to facilitate interoperability and security. The DDK now supports semantic model descriptions.

Three requirements dealing with architectural concepts have been validated in D10.2 Validation Report for DDK Demonstrator, which covers issues like loose coupling and domain independent services. (329, 327, 18). The positive validation of these requirements shows that Hydra is still following a modular, service-oriented approach.

What still remains to be done is reducing the number of rule engines (535). Currently, several managers employ different rule engines, which will be confusing for Hydra developers. During the last iteration, WP3 will assess these rule engines and investigate whether they can be conjoined into on single and common rule engine.

4.1.2 Devices

The DDK provides an API that allows developers to develop Hydra devices (33).

4.2 Impact on WP4

4.2.1 Architecture

The service-oriented architecture should allow services, components and devices to connect at runtime. The updated version of this issue states that also the tools for initial configuration and re-configuration should rely on these concepts. This aims towards conceptual integrity and an easy to use application (317).

4.2.2 Communication

Depending on the concrete application, Hydra supports TCP as well as UDP for Web Service Communication. In order to equip the device developer with the right flexibility, it would be much more convenient to leave this decision to him. This requirement has been implemented during the third iteration; the validation has been successful through tests that have been performed on different protocols for Web Services (368).

It has become clear, that the Event Manager has to handle events according to their priority. Since some events are critical to the functioning of the system they have to be prioritized when the load is high. Stress tests are recommended for testing this mechanism (479).

4.2.3 Configurability

Auto-configuration of devices is now supported (334).

4.2.4 Networking

In the networking area, WP4 will develop methods to make protocols changeable at runtime. For self-optimizing systems, it is necessary to realize an optimal configuration of protocols as guided by e.g. quality of service properties (533).

4.2.5 IDE

Hydra supports a range of communication protocols (TCP, UDP, Bluetooth; SOAP, REST), which can be used in different combinations (532). WP4 will develop concepts and tool-based support, to find a meaningful solution to the increasing amount of possible combinations of protocols.

4.2.6 Interface

WP4 will also take care of extending the Ontology Manager so that it provides domain-based properties for querying, which is needed e.g. by the QoS Manager (530).

4.3 Impact on WP5

4.3.1 Architecture

For the next iteration, the aim of having at least 10% of the data mirrored or striped using some kind of RAID technology still remains open (503).

4.3.2 Communication

Requirement 505, dealing with data access mechanism in the Storage Manager has been revised in the last iteration and has now passed quality checks. Stored data has to be accessible by non-Hydra applications.

4.3.3 Networking

The requirement 523 has been rejected for two reasons: First, it is not a requirement by itself but a rather a criticism to the JXTA solution, thus it does not follow the requirement structure that is required. Second, this requirement if reformulated as one, contradicts other network and security requirements where the required persistent identifiers (CryptoHID) are pointed out.

4.4 Impact on WP6

4.4.1 Architecture

From the lessons learned it became clear that there should only be one in-memory copy of semantic information per device, and all managers relying on ontologies should use that (534).

4.4.2 Device Discovery

Four requirements have been implemented. The middleware now supports sophisticated UPnP-based device discovery mechanisms (501, 500, 110, 91).

For the next iteration WP6 will develop tools for allowing developers to specify rules of functional quality of service (e.g. replacing failed devices or services) (392). This also relates to requirement 492, which states the need for semantic grouping of devices in order to reach high performance and low energy consumption.

4.4.3 IDE

Hydra now supports service composition during design-time (113).

4.4.4 Middleware Layer

In the middleware component, four requirements have been implemented. Security can be defined semantically (376). Depending on context and application, Hydra provides different views on a device (120). Furthermore, the middleware components can be deployed in a distributed way on different platforms (115) and semantic descriptions can be attached to device web services (114).

4.4.5 Security

The security protocols of Hydra device proxies can be used when integrating devices via proxies (477).

4.4.6 Service Discovery

A common naming convention for services needs to be proposed (499). This requirement has been reopened after being marked as not making sense.

4.5 Impact on WP7

4.5.1 Architecture

Regarding security issues on the architectural level, five requirements mainly dealing with security policies have been created from the revision phase after the second review (510, 509, 508, 498, 493). These requirements have been implemented during the third iteration.

Requirement 496, also belonging to this set, will be implemented in the next development cycle.

In the last version of this report, requirement 50 has been marked as being imprecise. It has been updated during the lessons learned process of cycle three and will be considered in the next iteration.

4.5.2 Communication

Two requirements have been implemented during the last iteration. Hydra access control policies now support credential-based authentication (364). Requirement 49 (authentication) has been partially implemented by the Trust Manager in the last iteration. What was still missing was a protocol for exchanging and verifying the keys. This issue has been solved.

4.5.3 IDE

The IDE now provides techniques for analysing the possible impact of semantic policies for correctness, possible impacts, and conflicts (494).

4.5.4 Security

Requirement 471 dealing with mechanisms that can be used by application developers to implement end-to-end message protection has been implemented.

Discussions about two security-related requirements have been finalised and led to the implementation of secure cryptographic key management (79) and protection mechanisms for stored data (45).

Requirement 512 states that regarding policy decision and enforcement on embedded and mobile devices, it has to be evaluated whether it is possible to port existing decision mechanisms to mobile devices. This requirement has been marked as duplicate.

5. Conclusion

In comparison to the last Requirements Report (D2.7 v2), more requirements have been implemented and less new requirements have been created. This progress is in line with the Hydra development and requirements engineering process and plans. The lessons learned from iteration 3 should be the last possibility to create new requirements. From a requirements engineering point of view, the middleware has evolved very well, because less new requirements have been created and the amount of implemented requirements is much higher.

6. Open requirements

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-535	Non-Functional	Architecture	Reduce number of rule engines	Major	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 assesment if they can be conjoined.	Open
HYDRA-534	Non-Functional - performance	Architecture	There should be one set of semantic APIs for all of Hydra	Major	wp4:ll5, third cycle, as reported in D2.11	There should only be one in-memory copy of semantic information per device, and all managers relying on ontologies should use that.	Open
HYDRA-533	Functional	Networking	Protocols should be changeable at runtime	Major	Different protocols have different functional and non-functional properties. A self-optimizing system needs to be able to realize an optimal configuration of protocols, as guided by e.g. QoS properties such as for energy awareness or security.	It should be possible to realize a scenario involving protocol change.	Open

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-532	Functional	IDE	Support autogeneration of eclipse project files	Major	We support using a range of protocols: UDP, TCP, and Bluetooth. Each can be used on JSE standard, JSE on OSGi, JME; For web-services they can use REST or SOAP; This gives a total number of combinations: 3x3x2 resulting in 18 projects if we maintain one for each combination. Thus the range of potential combinations quickly becomes too large to support manually.	Manual labour required to use specific combinations of platforms, protocols etc should remain constant rather than proportional to the number of combinations of these	Reopened
HYDRA-530	Functional	Interface	Domain-based Properties available for Querying	Trivial	For invoking the reasoning in Hydra ontology in order to retrieve QoS property data values.	For QoS Manager it is necessary to be able retrieve current QoS property data values from Ontology Manager in order to process semantic service selection.	Open
HYDRA-521	Functional	Security	Linking Securlty Policy Language and Policy Manager to the Security Ontology	Major	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 expresse 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.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-520	Functional	Security	Semantic Interoperability of Security only using external elements	Major	To enable true Inclusive interoperability of security you need to be able to introduce new security solutions without changing a device or an application. This require the full support of ontologies, external assertion providers and possible load modules to add new cryptographic capabilities.	It should be possible the phase-out broken security component or phase-in a new security component without changing a device or application. Phase-out can happen merely by changing the assertion of the broken which is referenced in security policy. Phase-in by mapping the new capabilities or credentials with the required Assertiona evaluation linked to the Security Objectives	Part of specification
HYDRA-519	Non-Functional	Architecture	It should be possible to implement managers in either programming model.	Major	The architecture should be fairly independent of any specific programming model. It should be possible to implement managers in either programming model.	It is possible to implement managers in either programming model or not.	Open
HYDRA-518	Functional	Networking	No external standards should dictate the virtual layer.	Major	Hydras manage internal standards in the virtual layer.These cannot be dictated by external standards.	External standards do not create limitations for HYDRA internal. All access to the virtual layer is done through HYDRA middleware	Open
HYDRA-516	Project Issue - open issue	Modelling	Context management need to be layer-specific	Major	Context management need to be layer-specific.Collection of real-world sensor and other application data should be classified as an application. Otherwise context management will conflict with security requirements.	Context management is layer-specific or not.	Open

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-515	Functional	Configurability	Support of domain-specific ontologies	Major	To establish knowledge or application domain interoperability, HYDRA should be able to support domain-specific ontologies on a structural level. Interoeprability can only be established to the degree external ontology support exists.	HYDRA is able to support domain-specific ontologies or not.	Open
HYDRA-514	Functional	Security	Authorisation based on blinded certificates in the Virtual layer	Major	Authorisation for devices and users based on blinded certificates must be supported in the virtual layer in order to support logical groups without linking virtual devices or violating the principle of only revokable information in the virutal layer.	A protocol based on blinded certificates exists that can be used to authorise devices or users towards a software component.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-513	Non-Functional - maintainability	_unassigned	Requirements must be validatable	Critical	The requirements stated in this list must be formulated in a way that makes it possible to validate them. They have to define a goal that can be reached at some point in the project. The following types of requirements are NOT validatable: - Open formulations: "Hydra should support..." / "It should be possible with Hydra ..." / "Hydra should not limit ..." Instead of describing what should not be subject to Hydra, describe what Hydra WILL provide. - Fuzzy statements: "Semantic interoperability and semantic-cooperative standards need to be established ..." Instead make clear what the goal is: "Ontologies and reasoning mechanisms have to be used for negotiating X between Y and Z" - Statements that can't be quantified: "Communication should be secure" Instead describe what that means: "Communication has to be encrypted and signed, anti-replay mechanisms have to be included and trustworthy non-repudiatable timestamps must exist"	For every requirement, it can be clearly evaluated whether it is fulfilled or not.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-512	Non-Functional - security	Security	Policy decision and enforcement on embedded / mobile devices	Major	In many cases, security policies (access-control as well as obligation) have to be enforced on resource-restricted platforms. Further, as those platforms might represent a "policy domain" by themselves, policy decisions also have to be made on that platform. It has to be evaluated whether it is possible to port existing decision mechanisms to mobile devices. If not, a dedicated solution for resource-restricted platforms has to be found.	Security policies can be decided and enforced on resource-restricted platforms, e.g. a smartphone.	Reopened
HYDRA-511	Functional	Modelling Security	Semantics for obligation policies	Major	When specifying obligation policies, developers should be able to use impllicit knowledge represented in ontologies. For example. semantic information could be used to describe the conditions that trigger the policy and the actions that have to be executed. The decision mechanism for obligation policies must apply reasoning techniques to make use of this implicit knowledge.	Obligation policies can be based on ontologies and reasoning can be used for deciding those policies.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-507	Functional	IDE Security	Policy Editor for Access-Control Policies	Major	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.	Part of specification
HYDRA-506	Functional	Communication	It should be possible to lock files.	Major	For many reasons it can be important to know that an application updates some data, so that other applications should wait using it until update is done. There should be a read/write locking.	All write access is aborted if a file is locked.	Part of specification
HYDRA-505	Functional	Communication	It should be possible to access data in Storage Manager using a well defined protocoll, e.g. WebDav	Minor	Using external Applications it should also be possible to access data without to much pain. Exporting storage using WebDav gives the User the ability to access it as network devices on most operating systems.	50% of the storage can be accessed by non hydra applications.	Part of specification
HYDRA-504	Functional	Configurability	It should be possible to add and remove physical storage from a Mirror/Striping-Set.	Major	If there is some striped storage and it is no more 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.	Part of specification
HYDRA-503	Functional	Architecture	It should be possible to combine different storage for mirroring or striping.	Major	To get better storage we need to implement some RAID-Technologies inside Hydra to mirror data over different Storage Manager or to stripe data.	Replicated and Striped devices can be build up on each other.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-502	Functional	Interface	It should be possible to store simple key/value pairs.	Major	Not every Application storing some data like sensor data want to use the full overhead of a filesystem and files. The idea behind this issue is to store somethink like cookies in a browser.	Storing and recieving cookies to a given Manager does not need more then 3 requests.	Part of specification
HYDRA-499	Non-Functional - usability	Service Discovery	Services common naming convention	Minor	Using a common naming convention, for the same services, to give uniformity in interaction procedures. This convention should be extended to input/output services data format. This should provide a more efficient group creation, a simple internationalisation and localisation processes.	When at least the 95% of services with the same purpose, realised by means of different solutions (sensor for a ZigBee node, a Web Service connection for the weather monitoring), respect the same naming convention.	Reopened
HYDRA-497	Functional	IDE Security	Analysis of conflicts between policy domains (dynamic analysis)	Major	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.	Part of specification
HYDRA-496	Functional	Architecture Security	Conflict resolution between policy domains	Major	Devices may be subject to different policy domains. If communication beween these domains is to be established, conflicts between the different policy domains may occur. Hydra needs to provide mechanisms and protocols to resolve these conflicts.	Conflicts between policy domains are recognized and can be handled.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-492	Non-Functional - performance	Devices	Semantic grouping for Non Hydra Enabled devices	Minor	Hydra could manage not only the single device services at once but a group of them if they are in the same semantic context in order to reach higher performances and low energy consumption (even if the hydra middleware doesn't address this topic) limiting the communication time per single device.	90% of non-hydra enabled devices with the same functionality in the same context has to implement similar functions (e.g. user detection in a room will turn on all lights)	Reopened
HYDRA-491	Functional	Security	Authorisation based on semantic information	Major	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.	Part of specification
HYDRA-488	Functional	Devices	Modular and standard device integration	Major	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.	Part of specification
HYDRA-487	Non-Functional - security	Security	Improve handshake protocol between Network Managers for exchanging certificates	Major	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	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-486	Functional	Networking	Hydra propietary supernodes are needed to support D2D communication between networks	Minor	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	Part of specification
HYDRA-482	Functional	SDK	Support fuzzy or probability concepts in self* reasoning'	Major	The swrl/owl based flamenco tool should be complemented with ability to handle fuzzy concepts/ probabilistic reasoning.	fuzzy concepts should be supported through e.g. probabilistic models.	Open
HYDRA-481	Project Issue - task	Configurability	support self-* experimentation	Major	We should develop tools that makes it easy to do experimentation with self-managing techniques	'-	Open
HYDRA-480	Functional	Architecture	Only a Jira test issue	Trivial	The only rationale is to test Jira.	always fits	Quality Check passed
HYDRA-479	Functional	Communication	The EventManager should support event prioritisation	Major	The EventManager should handle events according to their priorities. Some events are critical to the health of the system and should be prioritized over others when there are a high number of events being routed through the system	Stress test of the event notification system. If the volume of events exceeds the capacity, events with high priority should be delivered first, and only be discarded as a last resort	Quality Check passed

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-474	Non-Functional - performance	Security	Core Hydra security mechanisms should run on embedded devices	Major	Core Hydra security is essential for protecting communication between managers of a virtual device. Thus, it should be scalable down to resource-restricted platforms	Core Hydra security handlers perform sufficiently fast on resource-restricted platforms	Part of specification
HYDRA-470	Functional	Configurability	Device Ontology	Major	Knowing a priori if a device could support a data format could help the application developer to better exploit the resources of the device	The device information could host a field where is pointed out the proper data format	Requirement does not make sense

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-468	Functional	Communication Networking Security	Different levels of security must be supported	Major	In the healthcare scenario there are 2 communication types: - the inter-BAN communication - the internet communication Each of them could implement a different security criteria. The middleware could support different security levels during communications with wireless devices. For example, a simple accounting procedure for devices near to the user (a BAN in the healthcare scenario) and an harder codification for long distance communications where identity data are transmitted are supported.	It must always be possible to implement at least two different security levels for an application.	Ambiguous Requirement
HYDRA-467	Functional	Configurability	Device reliability and fault tolerance awareness	Major	A device could always communicate information but how to assure that this information is correct?	The middleware should provide procedures to identify the device reliability and the state of the device (Self-Diagnosis)	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-466	Functional	Communication	QoS selection	Major	In the healthcare scenario there will be the eventuality to send information with a particular QoS due to the data nature (e.g. some high quality images or emergency data)	The middleware should provide a QoS selection criteria for devices which have to work with critical information	Part of specification
HYDRA-464	Functional	Communication	Connection availability monitoring	Major	It could be useful inform the user or application about network status, giving also to him information about the status of network operations	The middleware should provide methods to monitor connection availability, providing also the connection type and the status of the pending network operations	Part of specification
HYDRA-461	Functional	Modelling	Device development kit will be used mainly for creation of semantic device descriptions	Major	devices are developed independently, so it is usually described without knowing exactly the environment in which device will be used.	In 90% of all cases the device description could be done within device development kit	Part of specification
HYDRA-460	Non-Functional - maintainability	SDK	use production system instead of, or together with fixed workflow	Major	to let system workflow be easily modifiable via production rules, instead of modifying workflow every time the system is modified (devices added, removed, ...)	Modification via production rules is possible in more than 33% of all cases	Ambiguous Requirement
HYDRA-459	Functional	Networking	Load balancing	Major	A sort of load balancing could be provided if information like number of devices involved in the communication are available, e.g. only 8 Bluetooth devices could realize a piconet, so the user couldn't add more than 8 devices	New devices have to be easily added to the system	Ambiguous Requirement

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-458	Non-Functional - performance	Communication	Bandwidth and central frequency	Major	knowing the busy channel bandwidth and central frequency, may help to rule out a set of frequencies by the available frequencies for a certain technology	This information could improve the HYDRA system performances in terms of communication delay	Ambiguous Requirement
HYDRA-457	Non-Functional - performance	Communication	Wireless devices busy channels	Major	Knowing all busy communication channels may help to minimize RF collisions	This information could improve the HYDRA system performances in terms of communication delay	Ambiguous Requirement
HYDRA-453	Functional	Configurability Context Devices	Data Acquisition must be adjustable to national laws and regulations.	Major	In different countries exist different regulations and laws regarding processing of data, especially in case of health related data. The Data Acquisition needs to be flexible so that it can be adjusted to these regulations.	Data aquisition must be adaptable to EU-country laws.	Part of specification
HYDRA-448	Functional	Context Security	The data acquisition must be in user control, i.e. the end user needs to be informed about leakage of information.	Major	To ensure privacy it is essential that the end user stays in control of what data is passed on by/to any acquisition component. This can either happen by notifcation (e.g. via logs) or by dedicated user interfaces (provided by the developer user or the middleware itself) where the user can adjust the settings.	Data acquisition components of the middleware should not hide data and provide access to all data.	Part of specification
HYDRA-447	Constraint	Context	To enable data acquisition, the required devices need to be in the network	Trivial	If the required components are not available in the network, data acquisisiton, and in parts context awareness, cannot take place	In 100% of all cases where data acquisition will take place the components must be available in the network	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-446	Functional	Communication Security	Security parameters negotiation	Major	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	Part of specification
HYDRA-444	Functional	Communication Security	Pseudonymous communication should be supported	Major	If the user/device is not required to identify himself to participate in a communication, it should be possible to use anonymous communication or to use pseudonyms as in e.g. eBay.	In 100% of cases where pseudonyity/anonymity is required the communication does not rely on idenitifcation.	Ambiguous Requirement
HYDRA-443	Non-Functional - performance	Devices	Storage Manager - Gateways must allow efficient access to store data from associated devices	Major	Users and authorized external systems can access the data (received from associated devices) stored on the gateways in an efficient way	90% of data access requests are answered within seconds	Reopened
HYDRA-442	Functional	Devices	Proxy - Gateways can filter and react to data received from associated non-hydra devices	Minor	Part of the proxy functionality may include support for filtering of the received data and possibly a reaction to high or low values. Non-hydra devices can not be expected to analyze the data themselves, so the gateways could take care of this	50 % of Gateways supports filtering and reaction to received data	Part of specification
HYDRA-433	Functional	Communication Networking	Session Management - Persistent sessions	Major	It must be possible to make sessions persistent.	90% of the sessions inside Hydra can be persistent.	Requirement does not make sense

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-427	Functional	Networking	D2D communication - Group management	Major	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 this groups share the same credentials and can communicate isolated from non-group-members.	90% of the devices involved in the the D2D communication system can create, join and leave groups	Part of specification
HYDRA-409	Functional	Devices	Storage Manager - Device information metadata	Critical	Devices' data must be stored with essential metadata, like time and device localization	90% of device information is stored with metadata	Reopened
HYDRA-407	Non-Functional - usability	Devices	Storage Manager - Gateways information stored synchronization	Major	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	Requirement does not make sense
HYDRA-406	Non-Functional - operational	Devices	Storage Manager - Gateways information gathered storage	Major	The Hydra middleware will need mechanisms that allow the storage on the Gateways of information gathered by devices with accessibility limitations	90% of Gateways support the information gathered storage	Requirement does not make sense
HYDRA-396	Non-Functional - usability	Devices	Hydra-enabled devices - May be mobile or fixed equipment	Major	A subset of the Hydra middleware (mainly Network Manager) can be deployed in mobile (PDA, Smartphone) and in resource constraint devices (Home Gateway)	30% of state of the art PDAs, Smartphones and Home Gateways can host part of the Hydra middleware	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-394	Functional	Middleware Layer Modelling	Stateful service orchestration	Minor	In order to specify service workflows we need to be able to keep state between the execution of the stateless services.	Service orchestration can be done by creating service workflow definition.	Part of specification
HYDRA-393	Functional	Configurability SDK	Deployment scenario configurable by developer user	Major	A developer user should be able to specify how an application should be deployed over a set of devices, e.g, choosing a host device for a Device Application Catalogue.	Developer can specify deployment for specific devices by means of a tool or configuration file.	Requirement does not make sense
HYDRA-392	Functional	Device Discovery SDK	Rules for selection of alternative devices	Major	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, contructs are available that allow the developer to specify rules for when and how devices and sevices can be interchanged and combined.	Part of specification
HYDRA-391	Functional	Devices SDK Service Discovery	Device and service exception handling	Major	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 application logic.	SDK provides exception handling constructs that the developer can use to specify exception responses with a succes rate of 9/10.	Quality Check passed

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-390	Functional	IDE Modelling	Different views on the device ontology	Major	It should be possible to present a developer user with different perspectives on the device ontology, depending on that users functional needs (e.g., a services perspective, device category perspective. etc.)	At least two different views are available in the ontology browser	Part of specification
HYDRA-379	Functional	Context	Intelligent data fusion on real-time data	Major	In order to derive information from sensor data a semantic interpretation on incoming data needs to be performed in a semantic way on real-time data.	Data fusion on real-time data is performed in 90% of the time without dropping real-time data	Part of specification
HYDRA-378	Non-Functional - security	Modelling Security	Application model must provide the security requirements	Blocker	Application must provide the security requirements on a semantic level in order to resolve if devices are allowed to interact with the application or to allow the middleware to resolve the security in the process. If the application model contains security requirements all requests will be resolved correctly	For applications that allow devices to interact with them, the application model should contain at least one security requirement on a sematic level	Part of specification
HYDRA-375	Functional	Configurability Context Modelling	Dynamic Semantic discovery of application objectives	Major	Ask Jesper!	Ask Jesper!	Requirement does not make sense

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-373	Functional	Modelling Service Discovery	Semantically locate devices and information providers	Major	Because devices and service providers enter and leave the network dynamically the developer need to be able to specify what kind of device/information provider is needed and the middleware will search for a match and make it available.	lookup of devices/information providers can be semantically specified and matching devices/information providers will be reported by the middleware in less than 10 seconds after entering the network	Open
HYDRA-372	Functional	Middleware Layer	Interfacing wiht external systems	Major	Searching and using external services in decision support and application intelligence must be supported	Access to external systems using web service protocols (WS-I Basic Profile) is supported	Part of specification
HYDRA-370	Functional	Middleware Layer	Support for interfacing with external workflow systems	Minor	Applications must include workflow management possibilities	Supports at least one workflow system, for instance OpenWorkFlow	Quality Check passed
HYDRA-369	Functional	Modelling	Devices must have semantic description of its user interface	Major	Devices must be remotely accessible through a multitude of heterogenous networks using multiple, multimodal user interfaces.	Supports at least two different types of user interfaces for a device. If the device has a user inteface a developer should be able to control 80% of the user interface using semantic expressions.	Reopened

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-366	Non-Functional - performance	Devices	Services should run on embedded devices	Major	Service-orientation is a good match for many embedded devices. Web services will provide a gateway to many applications and it would be beneficial to be able to structure all of the communication in a system using the same primitives. Depending on the resources (energy, processing capacity) available such a service may run on the device or on a proxy	HYDRA supports services on embedded devices (Initial target should be Develco's DevCom 02 ZigBee module)	Part of specification
HYDRA-365	Functional	Middleware Layer	Ability to self-adaptation	Major	A knowledge model enables the middleware to contain a representation of itself and manipulate its state during its execution. This feature should serve as the basis for self-adaptation of the middleware (e.g. reconfiguration of resource usage, triggering the component-based services).	Middleware is able to adapt its configuratiton in 60% of identified cases requiring reconfiguration.	Part of specification
HYDRA-361	Functional	Architecture	Protection of System Integrity	Major	In order to prevent an inexperienced user to cause malfunctions by changing system configurations, the middleware should monitor, analyse and, if necessary, prevent or give notifications about faulty changes.	HYDRA middleware provides mechanisms to monitor system integrity and to react in the case of failures.	Quality Check passed
HYDRA-359	Functional	Device Discovery	Handling of different device versions in device	Major	The device ontology should be able to handle different versions of a device.	The device ontology can maintain at minimum 2 versions of any single device	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
			ontology				
HYDRA-358	Functional	Architecture Modelling Security	Developer must be able to semantically define security requirements	Major	If developers are to make devices that can co-operate through other protocols and security mechanisms, they have to be able to describe the inherent security requirements in a semantic interoperable language. It is not enough just to use a specific protocol's security as this does NOT tell WHY he uses it and WHAT he really needs for the application to proceed.	On the one hand HYDRA supports the semantic description of security requirements and provides mechanisms to translate those requirements into device specific protocols automatically. On the other hand HYDRA provides means in order to analyse (prospectively) existing device specific proprietary security protocols. HYDRA can detect incompatibilities of different protocols' security mechanisms.	Ambiguous Requirement

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-357	Constraint - requirement constraint	Devices Modelling Security	HYDRA must support device authentication based on context and semantics	Critical	In strong security implementation, virtualisation and context isolation depend on isolation. As such HYDRA has to be able to support devices that authenticate indirectly through recognition of pre-shared keys or using credentials (such as Direct Anonymous Attestation plus additional credentials) instead of through assumed identification of the physical device (such as MAC). The Security & Communication meta-model must not assume mandatory identification.	Device authentication is supported without device identification.	Ambiguous Requirement
HYDRA-356	Functional	Architecture	support for both a pull and push model	Major	By default, HYDRA components should exchange messages according to the push-model. However, in some cases, a pull model should also be available.	In 90% of all cases, the system can handle push and pull commands.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-354	Functional	Architecture	Support for virtual devices	Major	In order to make each user have his own view on a device, there has to be some kind of support for virtual devices. This means, that a single device may show up as multiple devices, which respectively provide a fraction of the original physical device's functionallity, depending on actual user needs.	95% of all access to the HYDRAmiddleware should be able to set up virtual devices.	Part of specification
HYDRA-351	Non-Functional - security	Architecture SDK	device should support virtualization of devices	Critical	if two users has access to the same physical device, the access to the device, by id, could be virtualized so each user only uses his id and the other user uses his own. We want the id to be large (min. 256 bit long.)	The sdk must provide a unique id in 99,99999%	Part of specification
HYDRA-350	Functional	Architecture	Data type transparency	Critical	Different devices in sensor networks use different bit sizes. HYDRA must provide tranparency between data types. HYDRA must provide some sort of datatype wrappers for the different arch and cpu types.	100% of all basic datatypes, 90% of less common datatypes can be transferred between devices with different bit sizes.	Part of specification
HYDRA-348	Functional	Devices	Detect errors in devices	Major	there should be specification language which allows the middleware to detect errors in a device	In nine out of ten cases the Middleware is able to detect errors in devices.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-346	Non-Functional - operational	Architecture	It should be possible to have closed subsystems	Minor	HYDRA should not prescribe that a system should be completely open (and service-based) in order to be part of a HYDRA application (e.g., Siemens may want Siemens heating systems to not be usable (in parts) by Phillips home control systems)	A manufacturer or an application developer should be able to design HYDRA components with propietary interfaces in 100% of all cases.	Part of specification
HYDRA-345	Functional	Middleware Layer	Support conflict resolution	Critical	Configurations/rules of devices may be in conflict (e.g., one rule wanting to open a window, another wanting to close it). There should be components available that help in conflict resolution by 1) proposing basic conflict resolution, 2) allowing for automatic reaction to proposed conflict resolution, and 3) allow for overriding basic/general resolution policies with application-specific ones	Check whether the SDK contains a component that supports such conflict resolution	Reopened

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-343	Non-Functional - usability	Configurability	Users should be able to understand and modify automatically created user profiles	Major	If the system learns automatically based on the behavior of its devices and users, profiles for users may get created. These may (will) be wrong and it should be possible to understand these (e.g., by visualization) and change them according to personal preference	Create scenarios and have users do a usability test [may be too application-oriented]	Quality Check passed
HYDRA-342	Functional	Configurability	Learning support in middleware	Minor	There should be support for developing components that support learning, e.g., from previous use patterns of tenants and react upon this. This may imply machine learning techniques	Are there components available for machine learning?	Quality Check passed
HYDRA-339	Non-Functional - usability	Configurability	User orders should generally take precedence over device orders	Major	When an authorized user issues a command to a HYDRA-based system, this order should take precedence over preprogrammed rules in general. It should be possible to determine when such order are dangerous/unsafe however	Conflict resolution system should have mechanism for the user to take precedence	Reopened

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-337	Non-Functional - operational	Architecture	There should be a procedure/strategy for interfacing with non-HYDRA devices	Critical	Not all devices will be HYDRA-enabled neither in the near nor the far future. The architecture should support communication with and use of such devices to enable developers of HYDRA-based applications to create rich applications	75% of non-Hydra devices can be integrated into HYDRA Middleware	Part of specification
HYDRA-335	Functional	Context	Location awareness/positioning support	Critical	HYDRA should enable developers to write applications that depend on context, especially spatial context.	A component for acquiring spatial context exist. At any tme, min. 75% of all devices attached to a HYDRA system can be spatially located. Also, there is a programming model for using spatial context.	Part of specification
HYDRA-331	Functional	Communication	THere should be a binary, efficient protocol for communication as default	Major	XML-based communication is good for interoperability but bad for performance. There should be provisions for using both types of protocols to communicate with services and since we are building middleware for embedded systems that are resource-constrained, an efficient protocol should be default	Is a binary protocol default? Is it efficient in terms of bandwidth used and processing time? (Should be evaluated against a reasonable definition of minimum hardware requirements for HYDRA-devices)	Quality Check passed

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-325	Functional	Devices	Support aggregation and separation of devices and services	Minor	Devices and services may exist in a separate application where they should not be influenced by nearby (wireless) devices such as in the case of an apartment. Thus it should be possible to view a set of services/devices as an aggregate that is separated and isolated from other sets of services/devices	Check support for aggregation and separation of devices/services	Part of specification
HYDRA-324	Non-Functional - performance	Architecture	Systems built using HYDRA should be scalable in terms of devices communicating	Major	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.	Part of specification
HYDRA-323	Constraint - scope of the product	Architecture	Distributed Intelligence should not lead to resource-heavy systems	Major	We have a need for "intelligence" (Semantics, reflection etc.). We have a need for supporting embedded systems. This should not conflict	Minimum hardware requirements (which must be supported by all target hardware) are defined and all hardware that meets the specifications is guaranteed to work with hydra.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-322	Functional	Configurability	UUAR: Support lightweight service composition	Major	A number of tasks in the domains will require collaboration from multiple services. This coordination and collaboration among services should be expressed via a service composition mechanism (could be ala BPEL for web services, but it should be useful for embedded systems)	Existence of service composition mechanism (specification and implementation of support) that is able to compose a potentially unbounded number of services. Evaluate that it is able to compose services from and run on our selected lowest range of embedded devices	Reopened
HYDRA-320	Non-Functional - maintainability	Architecture	Separate domain-oriented services and user interface services architecturally	Minor	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.	Part of specification
HYDRA-318	Functional	Device Discovery	Devices should be able to be added to the system at runtime	Critical	It should not be necessary, e.g., to shut a building complex down to add a new device to a room :-)	Devices can be installed, discovered, and used while the HYDRA runtime is running	Part of specification
HYDRA-317	Functional	Architecture	Support runtime reconfiguration	Major	To supporting monitoring leading to adaptation, the architecture should be dynamic in the sense that components/services should be connectable in new ways at runtime To ensure a conceptual integrity of the system and ease developer understanding, the tools for initial configuration and re-configuration should rely on the same concepts/mechanisms.	Services and devices can be connected in new ways during runtime in HYDRA-based applications	Reopened

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-314	Functional	Communication	Faults should be intercepted by middleware, notified to interested services	Major	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	Part of specification
HYDRA-312	Non-Functional - operational	Devices	Support profiling of devices' performance	Major	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	Part of specification
HYDRA-311	Functional	Devices	Special watchdog devices for monitoring availability	Trivial	The middleware should provide easy implementation of special watchdog devices (or services) for availability monitoring	Existing services that implement watchdog functionality available	Part of specification
HYDRA-310	Non-Functional - operational	Middleware Layer	Interoperability with external systems	Major	HYDRA should facilitate ease of interoperability with existing, non-HYDRA systems	Compare support in HYDRA to state-of-the-art (such as semantic web services). HYDRA should support interoperation on at least that level	Reopened

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-309	Functional	Devices	Map device (e.g., name of it) to presentation of, e.g., a room	Minor	There should be support for mapping the presence of devices and services to their location (e.g., where in a room inside a building) and description when building user interfaces on top of Hydra	The SDK supports this	Part of specification
HYDRA-308	Functional	Networking Security	The Security Level of an existing network should be determinable	Minor	For a device entering an existing network it can be useful to determine the security level of that network. Depending on the provided security level the device can decide to enter the network or not.	HYDRA middleware provides at least one mechanism enabling devices to determin the security level of an existing network.	Part of specification
HYDRA-300	Non-Functional - usability	_unassigned	This is a requirement for testing purposes	Blocker	Testing is grreat	blah	Requirement does not make sense
HYDRA-296	Functional	Security	Adaptability of Security Model with regard to existing security system(s)	Major	In the case of already existing security systems, HYDRA Security Model should be able to interoperate with them.	The HYDRA Security Model can operate with already existing security systems in 9 of 10 cases.	Ambiguous Requirement
HYDRA-294	Functional	Middleware Layer	Central service registry	Major	Services announce their availability and announce a description of their functionality in a central service registry in a unified format. Clients (users or other services) query that registry to find an appropriate service for their needs.	A central service registry exists. Services announce their availability and describe their functionality in a unified from. Clients can query the registry to find an appropriate service.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-293	Non-Functional - maintainability	Middleware Layer	Documentation of API and basic services	Major	To enhance the developers' productivity, the API and the basic services provided by the middleware must be documented.	Documentation is available for API and basic services.	Part of specification
HYDRA-292	Functional	Middleware Layer	Self-diagnosis of devices	Major	To enhance the robustness of a HYDRA system, devices should be able to check its own diagnostic state and report errors to an appropriate component	HYDRA Devices can conduct self-diagnosis and detect / report failures in operation in 98% of all cases.	Part of specification
HYDRA-291	Non-Functional - usability	Communication Configurability Middleware Layer Modelling Service Discovery	Quality of Service as search criteria for service selection	Major	The selection of appropriate services for a given task requires the reflection of QoS-related search criteria such as cost, performance, etc.	QoS-criteria can be used in the selection of services in 95% of all cases	Part of specification
HYDRA-290	Functional	Configurability	Share service orchestration between users	Minor	Service orchestration definition should be shared between developer users, in order to allow a distribution of useful service orchestration to other developers	Service orchestration definitions can be shared between users	Part of specification
HYDRA-288	Functional	Architecture Communication Device Discovery Devices Modelling	Query devices for their functionality	Critical	Enable developers to get information about the offered functions of a certain device in an ad-hoc manner	All HYDRA enabled devices can be queried for their supported functionality	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-258	Functional	Middleware Layer	Automatic software updates	Major	HYDRA middleware should prevent the need to manually update software	Support for automatic software updates	Part of specification
HYDRA-253	Functional	Architecture Configurability Security	Hydra provides mechanisms to users for managing identities	Major	In order to build trust & security, HYDRA middleware has to be open to many identity mangement principles, of which user empowerment is core to overcome basic trust challanges. Identity management must be provided by the Hydra middleware and must be available to users (not only service providers or other central parties)	A mechanism exists that allows end users and devices to have at least two non-linkable identities.	Part of specification
HYDRA-248	Functional	Configurability	Defintion of Virtual Devices	Critical	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 logcial device definition. A virtual device can fully replace a physical device	Reopened
HYDRA-247	Functional	Configurability Devices	Integrate non-HYDRA devices with an existing HYDRA environment	Critical	For HYDRA to be inclusive and able to provide value beyond what developers has inentionally enabled, third parties have to be able to integrate their devices.	90% of Non-HYDRA devices can be integrated in a Hydra environment	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-244	Constraint - scope of the product	Security	Security Metamodel has to be self-contained	Major	The Security metamodel has to be able to describe a full security model without the use of external knowled/databases. Making a consistent trust model across not pre-known devices require the ability to resolve the security properties in context.	Can a Business Model define a set of security requirements which can be resolved at runtime against actual configuration	Ambiguous Requirement
HYDRA-241	Constraint - requirement constraint	Architecture	MIddleware should be open source.	Blocker	We have stated in the DoW that we will produce open source software.	The core components of the Software are open source.	Part of specification
HYDRA-239	Functional	Context Security	Automatic service diagnostic for security relevant services	Major	Security relevant services should provide a self-diagnostic services that provides an overview of all security-relevant features	Self-diagnostics in all security relevant services implemented	Ambiguous Requirement
HYDRA-237	Non-Functional - maintainability	Architecture	The guaranteed future should be ensured	Major	The HYDRA middleware should be kept adaptable and future proven.	After 10 years in the market, the middleware can still be used.	Part of specification
HYDRA-236	Functional	Architecture	Middleware is extendable with additional functionality by plug-ins	Critical	The middleware provides basic services that could be enhanced and adapted by additional integration of specific plug-ins.	The middleware provides well-defined interfaces for additional plugins	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-235	Functional	Modelling	Modelling components should be available as plug-ins	Critical	The middleware should provide basic services. Advanced features like modelling components should be available as additional plug-ins. Approved components could be integrated in a later stage	At least 2 Modelling plug-ins available.	Part of specification
HYDRA-234	Non-Functional - usability	Configurabilit y	The middleware should be selfdescriptive	Major	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 developer have a clear understanding of the HYDRA middleware after one week of experience	Part of specification
HYDRA-233	Functional	Middleware Layer	Self-healing function of middleware	Major	To ensure robustness and reliability, the middleware should dispose of self-healing and self-reconfiguration abilities.	A breakdown of service components should be automatically intercepted in 65% of the cases	Part of specification
HYDRA-229	Functional	Security	Services are responsible for authentication	Critical	The single service should be responsible for authentication request in order to ensure a robust and secure system	All security critical services trigger authentication requests	Part of specification
HYDRA-226	Functional	Configurabilit y Devices	Device ontology should be available	Major	In order to be able to integrate devices in an ad-hoc manner a device ontology must exist allowing to exchange basic information of services	In 90% of all cases, devices can be integrated in an ad-hoc manner.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-225	Non-Functional - maintainability	SDK	Interactions and consequences of changes to services on other services should be highlighted	Minor	The developer should have a tool that helps him understand the complex interactions of services and the possible consequences of changes on one middleware service to other middleware services	A service monitor that is able to show interactions with other services is implemented	Part of specification
HYDRA-222	Functional	Security	Role-management should be handled by the middleware	Major	Conflict resolution referring to access rights should be based on a role management	Role management is implemented	Part of specification
HYDRA-221	Functional	Middleware Layer	Policy should handle the possible actions	Major	Automatic system actions should be based on well defined policies to avoid conflicts.	All automatic actions are policy based.	Part of specification
HYDRA-219	Non-Functional - performance	Architecture	Redundant core components	Major	To ensure high robustness, core components should be redundant.	No core component should be unique.	Part of specification
HYDRA-218	Functional	Device Discovery Devices Interface	Support interaction devices	Major	Interaction devices provide users with different forms of output (display) capabilities. This could include simple displays, tablets or more advanced units.	Interaction devices (displays) are included in the HYDRA device ontology and can be mapped to the end-user inteface of an application.	Part of specification
HYDRA-217	Non-Functional - performance	Architecture	The middleware should ensure high robustness of services	Major	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.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-216	Functional	Configurability Middleware Layer	The middleware should have a graceful degradation service	Major	Services should be organised in a cascade of services in order to allow an orchestration of services providing best possible services down to basic services automatically, according to their availability	Service orchestration is possible n a hierarchical way. An automatic selection of the best service is possible within max. 500 msec.	Part of specification
HYDRA-215	Functional	Middleware Layer	Middleware only handles communication.	Critical	The middleware should implement only the most basic service, i.e. the communication. All high level functionalities will be realized by additional services.	The middleware only handles communication. All other functionality is realised by external components.	Part of specification
HYDRA-214	Functional	Modelling	A decision component/service should exist	Major	There should be a decision component that is able to take actions according to specified rules or reasoning components.	At least one decision component in the middleware	Part of specification
HYDRA-212	Non-Functional	IDE	Support for a declarative application development paradigm	Major	A declarative approach can hide complexity of underlying structure and can increase productivity of embedded software development.	More than 50% of the module functionality should be programmable using a declarative approach.	Requirement does not make sense
HYDRA-211	Functional	Architecture	There are components/services in the middleware that integrate subsystems	Major	The integration of basic systems to subsystems should ease the configuration of higher level services. Higher level services could then consist of a combination of basic systems	It should be possible to combine basic services to higher level services. At least one higher level service relying on a combination of basic services exists.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-209	Functional	Service Discovery	Middleware has a service for providing information about the technical environment/infrastructure	Major	In order for the services to query the available infrastructure the middleware should provide such a service	A services at the middleware provides information about more than 95% of the technical environment/infrastructure	Part of specification
HYDRA-207	Functional	Middleware Layer	Service selection by context	Critical	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 are returned that are relevant in the current user´s location and context.	In search requests for a specific service, contextual information like a spatial position is allowed.	Part of specification
HYDRA-206	Non-Functional - operational	Service Discovery	Middleware supports service discovery	Blocker	The developer needs to query the available services during runtime	Services discovery during runtime in the Middleware results in at least 95% available services	Reopened
HYDRA-204	Non-Functional - performance	Devices	Devices have automatic error diagnostics	Major	The devices should perform their own diagnostic test to provide their status upon request of the middleware in order to save performance and increase robustness and scalbility	In nine out of ten cases a request of the middleware should result in a valid status	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-203	Non-Functional - usability	SDK	Easily usable interface for developers/manufacturers	Major	SKD tools, APIs and documentations should be intuitive, well aranged and easy usable for developers.	Developers don't have to contact technical support nor to use help in 90% of problem cases.	Part of specification
HYDRA-202	Functional	Architecture	Functionality handled in Grid	Major	Grid technologies should be used for device networks e.g. to share resource usage, data storage over network, to access the data mining services, to use grid-enabled services provisioning, ontology services, etc.	Grid technology (share resource usage or data storage and data access over network) is used in 50% of cases where other than Grid technology does not fullfill needs.	Part of specification
HYDRA-201	Functional	Configurability	Self configurable devices	Major	Devices should be able to join (an leave) the network without any need for manual management or configuration handled by user. This feature requires the ability of devices to configure its connection and communication properties automatically.	Devices are able to join (and leave) the network without any manual user action in 80% of all cases.	Part of specification
HYDRA-199	Functional	Architecture	Modules should be extendable	Critical	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 replaceble by 3rd-party modules with equivalent functionality.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-198	Functional	Service Discovery	A service broker is responsible to provide services according to specific keywords	Critical	Service discovery should be enhanced by a service broker module/service as basic service of the middleware that enables the search for services according to specific keywords	Requests according to specific keywords will be provided a corresponding service in 8 out of 10 cases.	Part of specification
HYDRA-197	Functional	Communication Modelling Service Discovery	Services define their communication needs in terms of needed QoS parameters	Major	The services define their communication needs in terms of needed QoS parameters (needed bandwidth, needed quality...) without specifying the technical details. The middleware is free to choose the appropriate networking matching the specified needs	Every service specifies its QoS parameters	Part of specification
HYDRA-196	Functional	Service Discovery	Basic Service Registry	Critical	Services should register at a basic service/module of the middleware in order to provide a base for service orchestartion	All services should be itemised at the Basic service registry	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-195	Functional	Middleware Layer	Dynamic prioritisation	Major	Ability of active application components (e.g. devices, services) to dynamically assign the priorities to active components available for cooperation. Using the assigned priorities, active components should be able to dynamically select e.g. whom to communicate, what to communicate, when to communicate. Dynamic prioritization process is essential for system to be able to atapt its behaviour to current context (e.g. user's location, his interests, enviroment's characteristics, etc.).	Ability to change priorities based on current context provided.	Part of specification
HYDRA-194	Functional	Middleware Layer	Conflict resolution mechanism	Critical	Information obtained from different sources can be conflicting or contradictory. In this situation a conflict resolution mechanism should determine on relevance, reliability, and risk related to these sources.	The HYDRA middleware is able to proceed in its operation in 98% of all cases, where contradicting information or conflicting commands are received.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-192	Functional	Context Modelling	Context modelling	Major	Use knowledge models in order to specify the interrelations among context entities, to ensure common, unambiguous representation of these entities, to provide an explicit semantic representation of context, and to represent current context supports reasoning about context.	Current context represented as an instance of a knowledge model.	Part of specification
HYDRA-191	Functional	Context	Intelligent location determination	Major	Incorporating a wide range of location sensing techniques to obtain location information from different providers enables a reasoning engine to determine location with a certain probability.	Always select location determination mechanism with the highest accuracy.	Part of specification
HYDRA-190	Functional	Context	Learning situational context	Major	Knowing situational context (based on e.g. learnt knowledge on people's actions, behaviour patterns, movement patterns, intonation, registering specific events, etc.) is essential for classification of possible situations and related actions. Necessary for guessing intent of the user.	Recognition of 50 % of all situations.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-189	Functional	Devices Middleware Layer	Plug and play support for adding devices	Critical	Devices should be accessible as soon as they are discoverable and with the need for the developer to implement this functionality. This should be something like Plug´n´Play in operating systems.	Plug and play mechnism for inclusion of newly detected devices is done by the middleware	Part of specification
HYDRA-188	Non-Functional - operational	Middleware Layer	Conflict prevention service	Major	Certain combinations of multiple services' functionality can lead to contradicting instructions. A conflict prevention component should exist, that checks for agreeable combination of services.	Service combinations, that lead to contradicting instructions, are prohibited by a conflict prevention service.	Part of specification
HYDRA-187	Non-Functional - maintainability	SDK	standardized API for device classes	Major	All devices of a device class should have a set of methods that will be supported by each device. This makes it easier to implement functionality. To get a complete list of supported methods of a device the device should support querying it and responding back. This query for a complete list of methods is an example of one standardized method.	A set of methods is standardized for each device class.	Part of specification
HYDRA-186	Non-Functional - operational	SDK	GUI for configuring middleware parameters	Minor	To make the configuration of the parameters of the middleware easier for the developer	A GUI exists for configuring the middleware	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-185	Non-Functional - operational	Middleware Layer	Middleware provides basic services	Critical	In order to program AmI applications, the middleware must provide basic services. This makes life easier for application developers .Baisc services provide e.g. methods to query available devices and services or to pass messages between components	Middleware provides a set of basic services that at least contain basic functionality, that is needed by all services, like communication and a service / device registry.	Part of specification
HYDRA-184	Non-Functional - operational	Configurabilit y	Configuration with text files	Major	In order to configure the middleware, a configuration file in text format, e.g. XML, should be used. This makes the developers' lifes easier, since such a configuration allows for fast changes of the behaviour of the middleware.	80% of all middleware components are configurable with a text file.	Part of specification
HYDRA-182	Non-Functional - operational	Communicati on	Middleware realises communication	Major	The developer doesn´t need to care about how to communicate between devices. The communication between the devices is handled by the middleware	Middleware handles all communication without the need of the developer to implement communication code	Part of specification
HYDRA-180	Non-Functional - performance	Middleware Layer	Service mediating network connections according to different qualities	Major	There should be a service which lists different network connections depending on specified properties (connection speed, encryption). Devices can then negotiate such connections with remote devices, without the need to take care about the networking details	In 9 out of 10 cases devices should be able to automatically negotiate their networking condition.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-179	Non-Functional - performance	Service Discovery	Dynamic resource handling	Major	Resources (computational as well as devices) should be able to join or leave the environment whenever they choose. Could e-g. be enabled by short-lived transient services	Resources are able to join/leave the runtime middleware within less than 8 sec.	Part of specification
HYDRA-178	Functional	Security	Single-sign-on, run anywhere	Major	Single-sign-on, run-anywhere authentication service, with support for delegation of credentials to sub-computations, and mapping from global to local user identities can be beneficial in distributed environments.	One authorisation is sufficient to use resources in a distributed environment.	Ambiguous Requirement
HYDRA-177	Functional	Configurability	Dynamic scheduling of resource usage	Major	Dynamic scheduling of resource utilisation enables for applications to tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services, increases fault tolerance and cope with unexpected situations.	Application is able to re-schedule resource utilisation in 80% of single resource failure cases, if there exists the suitable resource(s) for substitution.	Part of specification
HYDRA-176	Functional	Architecture	Aggregation of resources	Minor	Aggregation of resources (e.g. computational) enables to outperform the limitations of a single system and to leverage available resource distributed across devices. This aggregation should be based on automatic coordination of multiple resources.	Device can distribute computation efforts among several other devices	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-175	Constraint	Middleware Layer	Support for resource sharing standards	Major	Middleware should support actual standards to describe what should be shared, who is allowed to share, the conditions under which sharing occurs, and protocols to negotiate access to individual resources.	Sharing resources is possible using main producers standards in 80% of cases.	Ambiguous Requirement
HYDRA-174	Functional	Architecture	Coordinated resource sharing	Major	Resource sharing enables the exploitation of distributed collections of available resources both computational as well as other services. Scheduling computation and access to resources is essential for running several applications on the middleware concurrently.	Resources can be shared by at least two entities.	Part of specification
HYDRA-172	Functional	Architecture	Learning resource usage	Minor	Learning usage patterns of utilizing devices and computational resources, and collaboration among application components is essential for self-configuration in order to optimise usage of available resources and overall application performance.	Model of a resource usage can be learnt.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-171	Functional	Architecture	Learning user behaviour patterns	Major	Learning of basic user behaviour patterns on device level (device configuration, sensor activation) in relation to specific users and specific security and situation contexts. Adaptation of devices enables applications to offer added value (e.g. detection of unusual situations, customized default configuration).	Device knowledge model of user behaviour can be expanded with new information.	Part of specification
HYDRA-170	Functional	Architecture	Statefull and stateless communication	Minor	Application developers should have the possibility to use statefull as well as stateless communication between components.	HYDRA provides an API that allows the implementation of stateful and stateless communication protocols.	Part of specification
HYDRA-167	Functional	Architecture	Distributed response composition	Major	Service orchestration should also enable the distribution of responsibility of response composition (e.g. Multi agent collaboration).	Response composition is distributed among two entities at least.	Part of specification
HYDRA-164	Constraint	Interface	Support for Service standards	Major	Middleware should support widely used standards for service description, discovery, orchestration and execution.	Standards defined by W3C and OASIS implemented.	Part of specification
HYDRA-163	Functional	Middleware Layer	Policy and Context are not part of the middleware	Major	Context awareness as well as making decisions based on policy strategies can be resource intensive computing processes. Modules providing these functionality must not be part of the middleware.	The middleware does not implement context awareness and policy strategies.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-162	Functional	Middleware Layer	Middleware allows implementation of fault detection service.	Major	Although fault detection as part of the middleware is not mandatory, the middleware must lay the foundation (e.g. an API) for building such services.	The middleware includes an API to implement fault detection.	Part of specification
HYDRA-161	Functional	Middleware Layer	Middleware must implement a role concept	Critical	A role concept can significantly simplify the resolution process of contradicting instructions.	A role concept implementaion is part of the middleware that can resolve contradicting instructions in 90% of all cases.	Part of specification
HYDRA-160	Functional	Device Discovery	Search masks for device/service discovery	Critical	When the developer needs a service he wants to be able to define search criteria for discovery of services	Search criteria can be specified and are respected by search services	Part of specification
HYDRA-159	Non-Functional - operational	Architecture	Service brokers must be organized in a hierarchical way	Minor	With hierarchical brokers the system becomes more robust and scalable. Users do not want that everything acts up in case of a fire and a broker goes down. Additionally hierarchical brokers allow for having certain rules/services only within a sub-domain.	Brokers are organized hierarchically	Part of specification
HYDRA-158	Functional	Communication Service Discovery	There should be a hook-up-service	Critical	When the developer creates a new application/device he wants to have a broker that can supply him with all available services that match certain criteria.	A request for a specific service according to specific keywords results in the provision of the corresponding service in 8 out of 10 cases	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-157	Functional	Service Discovery	Availability of combined services	Minor	A developer wants to easily access a higher level service which is in fact a combination of multiple services	High level services, consisting of at least two basic services, can be composed manually by the developer but this will not be done automatically.	Part of specification
HYDRA-155	Non-Functional - maintainability	Communication	All communication occurs through a central communication unit	Critical	Application developers need total control over a HYDRA system. Decentralized communication is considered as not feasible.	Communication and coordination happens through centralized unit.	Part of specification
HYDRA-154	Non-Functional - usability	Communication	Physical details of communication are invisible to the developer	Major	Developer is only intererested in getting messages to other devices and (very often) not in how they get there	Developer can build up basic communication links between two devices without having to know what the physical transport layer looks like.	Part of specification
HYDRA-153	Functional	Configurability Devices IDE Interface	Automatic generation of user interface	Minor	Manufacturers describe their devices in a special description language which can be used to automatically generate user interfaces for each device.	a user interface generator for all devices with standard capabilities exists	Part of specification
HYDRA-152	Functional	Interface	User must be able to overwrite automatism	Critical	Users dislike the idea of losing control and want to have the means to change system decisions	User can overwrite system automatisms in 90% of all cases	Part of specification
HYDRA-151	Functional	Devices	Devices send events when their status changes	Critical	This alleviates the problem of always having to poll for a device's status, when another device is interested in that status.	10 status changes at device level result in 10 events sent	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-148	Functional	Middleware Layer	Access to basic and extended functionality	Major	The middleware should provide a device's basic functionality via standardized, common methods. The way, extended / extra functionality can be accessed, should be also standardized.	HYDRA provides standardized access methods for at least 90% of all HYDRA-enabled devices. Some devices can have proprietary interfaces.	Part of specification
HYDRA-147	Functional	SDK	Simple interface for exploring / testing devices	Major	There should be an unintelligent/simple user interface which allows one to explore / test the functionality of a device out of the box. This interface is not part of the device, but can connect to all different kinds of devices.	A user interface for testing / exploring the funtionality of a device exists in the SDK.	Part of specification
HYDRA-146	Functional	Devices	Report errors in devices	Major	Devices should be able to report errors	The API provides at least one interface for reporting all kinds of possible errors to the middleware	Part of specification
HYDRA-144	Non-Functional - usability	Networking	Detect defective connection	Minor	The middleware should be able to detect if a device that has recently been integrated into the hydra network was not connected appropriately for whatever reason. May be that device simply does not fit into the network of other devices.	Middleware is able to detect defective connections of devices in 8 out of 10 cases.	Ambiguous Requirement

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-141	Functional	_unassigned	Download and harmonisation of third party device ontologies	Major	Device ontological models describing devices, which will be provided by manufacturers or third parties, should be automatically downloaded (updated) and harmonised to ensure the same ontological view. Formal definition of ontologies should be realised using the world wide accepted formats, recommended by W3C, such as RDF, OWL, OWL-S.	Ontologies from different manufacturers can be used if they are in RDF, OWL or OWL-S	Reopened
HYDRA-139	Functional	Architecture	Knowledge model of hydra middleware	Major	Knowledge model of the whole middleware providing developers with knowledge on all middleware components offers a guidance how ho compose a hydra-based application.	Support for knowledge model based rapid development is available	Part of specification
HYDRA-138	Functional	Modelling	Reasoner module	Major	A reasoner is fundamental to use ontological knowledge models and to infer new knowledge based on information from the models.	Reasoner module exists.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-137	Functional	Communication	Knowledge model of communication	Major	A knowledge model of possible connections and communication protocols of available devices, and cooperation/negotiation patterns enables to control communication within an application under normal circumstances as well as in unpredictable situations (fault tolerance and graceful degradation)	Devices are able to select device and suitable protocol for communication in dependence of available devices and their communication capabilities in 80% of irregular situations (if there exists available connection and communication protocol).	Reopened
HYDRA-136	Non-Functional - performance	Architecture	Dynamic architecture	Major	An architecture of a running HYDRA system can be easily modified by increasing or decreasing the degree of centralisation in order to balance utilisation of available resources.	In 95% of all cases, HYDRA supprots dyncmic migration of components to realise centralised and decentralised systems.	Part of specification
HYDRA-135	Functional	SDK	Migration to other platforms	Major	The IDE should support easy migration of HYDRA applications between different platforms. The IDE should contain tools for the identification of platform dependent code. Tools supporting the identification and writing of platform specific code should make the developement process more easy and efective.	The IDE supports application migration at least between two different platforms.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-133	Non-Functional - usability	SDK	Platform independent (meta) codebase	Critical	Using only one (meta) codebase for an application to be deployed on several platforms reduces development cost, time to deployment, and makes maintenance more easy since the developer is not bothered by writing platform specific code.	A unique codebase can be used at least on two different platforms.	Part of specification
HYDRA-132	Functional	Middleware Layer	Hot swap of platform components	Major	Deployed Hydra application should enable replacement of a platform component (utilised by some middleware module(s)) without interrupting operation. It enables to reduce down time of the application.	Hot swapping a component at run time is possible in 50% of all cases.	Part of specification
HYDRA-131	Functional	IDE	Model-based rapid development environment	Major	Development process can be speeded up by utilising formal models (structural as well as behavioural) of applications. Using the formal models, applications could be analysed, simulated, visualised, validated against requirements and documented on various levels of abstraction.	IDE enables to use abstract models.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-130	Non-Functional - usability	Modelling	Support for abstract modelling	Major	Enabling developers to work on more abstract level makes their task easier due to cutting out the necessity to cope with complexity imposed by target platform specific issues.	No need to cope with target platform specific issues in 9 out of 10 cases.	Part of specification
HYDRA-128	Functional	Architecture Communication	Comply with industrial standards	Major	The IDE and middleware should embrace existing industrial device integration and communication standards (initially at least the EIB/KNX for building automation).	Claimed support for any specific standard in HYDRA can be verified using the conformance rules / procedures available from the issuing standards body.	Reopened
HYDRA-127	Functional	Context IDE Middleware Layer	Spatial information management	Major	In order to be able to deal with the location of devices and other actors Hydra needs to manage spatial information.	The system can refer to in 90% of all cases "where" something is with a accuracy of 80%.	Part of specification
HYDRA-126	Functional	IDE	Automatic Device ontology updates	Major	The device ontology should automatically update its device descriptions.	The device ontologty can detect device updates and handle that in 7 of 10 cases.	Part of specification
HYDRA-125	Non-Functional - usability	Middleware Layer	Transactional updates	Major	It should be possible to rollback and recover from an unsuccesful update.	Rollback works in 7 out of 10 scenarios.	Part of specification
HYDRA-124	Non-Functional	IDE Middleware Layer	Automatic downloadable updates over the Internet	Major	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.	Quality Check passed

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-123	Non-Functional - usability	Architecture Configurabilit y	Support updates at run-time	Major	The middleware should be dynamically updatable at run-time due to critical systems updates (security updates, component upgrades, etc.).	Deployed middleware should execute 70% of the dynamic updates without failure and restart	Part of specification
HYDRA-122	Non-Functional - usability	Middleware Layer	Configurable and easy to install middleware	Major	The middleware should be configurable and easy to install/deploy.	The average installation time is less than 1 hour.	Part of specification
HYDRA-121	Functional	IDE	Optimised device ontology	Major	It should be possible to optimise the device ontology for instance by deploying a subpart of it to be used in device discovery process.	Possible to select and extract subparts of the device ontology	Requirement does not make sense
HYDRA-119	Functional	Architecture Context IDE	Domain modelling support	Major	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.	Part of specification
HYDRA-118	Non-Functional - operational	Context Device Discovery Middleware Layer Service Discovery	Considering interaction device capabilities	Major	The device should be able to collect data about the environment regarding other hydra devices in its proximity. Additionally, the system should be able to use this knowledge in adapting information sent to the interaction devices.	Interaction devices receive information that is tailored to its capabilities	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-117	Functional	Architecture IDE Middleware Layer	HYDRA component ontology	Major	In order to support automatic device proxy creation, a HYDRA middleware managers 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 devicce and service managers can be identified and selected through a software component ontology	Part of specification
HYDRA-116	Functional	Middleware Layer	Prioritisation of services	Major	Middleware should provide different methods/policies to prioritise available services.	Supports at least 2 different methods.	Quality Check passed
HYDRA-109	Functional	Configurability Device Discovery IDE	Device Virtualization	Major	The complexity of devices may be hidden, or simplified, by means of virtual device interfaces, these would correspond to "views" on device descriptions as provided by the HYDRA device models (ontologies).	An existing virtualization can be used to find exactly one proper HYDRA device.	Part of specification
HYDRA-107	Functional	IDE	Tool for managing access rights of services	Major	Tool that allows setting and managing access rights of services and resources.	Access rights can be configured and managed.	Quality Check passed
HYDRA-106	Functional	Architecture Middleware Layer SDK	Persistent storage	Major	Settings, configuration and other data should be persistently stored in the system.	Data can be peristently stored.	Reopened
HYDRA-105	Functional	Middleware Layer	Controlling access rights to services	Major	The middleware should provide methods to control access rights to services and resoruces	All un-authorised accesses are blocked.	Quality Check passed

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-104	Functional	Middleware Layer	Automatic Discovery of Services	Major	It should be possible to configure the middleware to discover available services that meets defined criteria.	8 of 10 services are automatically discovered.	Part of specification
HYDRA-103	Functional	IDE	Automatic device ontology construction	Major	The construction of a device ontology should be facilitated through finding and parsing product or device descriptions to annotate and produce ontology entries. The component should handle different input formats like Word, PDF, HTML, databases.	5 of 10 device descriptions can be succesfully processed	Part of specification
HYDRA-102	Functional	IDE	Device Ontology with user interface	Major	Tool that allows browsing, searching, navigating device classes and their capabilities.	Tool for browsing device ontology exists	Part of specification
HYDRA-100	Non-Functional - operational	Architecture	Graceful degradation	Major	The system should be able to continue working even when devices and/or communication fails.	The system continues working	Reopened
HYDRA-99	Functional	IDE Middleware Layer	Device reliability level	Major	It must be possible to assign a reliability level to a certain device that for instance can be used to resolve contradictory device events.	Reliability levels exist.	Part of specification
HYDRA-98	Functional	Middleware Layer	Detection of device failures	Major	The system should be able to detect malfunctioning devices in order to be robust.	Malfunctioning devices are detected in 8 out fo 10 cases.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-97	Functional	Middleware Layer	Detect livelocks	Major	The middleware must be able to detect livelocks between two or more devices, i.e. devices that are constantly changing each others state back and forth.	Detects livelocks in 7 out of 10 cases	Part of specification
HYDRA-96	Functional	Middleware Layer	Detect deadlocks	Major	The middleware must have functionalities for detecting deadlocks between devices, for instance two devices that are waiting for each other to take an action.	Detects deadlocks in 7 out 10 cases	Part of specification
HYDRA-95	Functional	IDE	Rule Editor	Major	A tool that allows editing, visualising and structuring of device and application rules.	The rule editor works and allows expression of 80 percent of rules in an application	Quality Check passed
HYDRA-94	Functional	IDE	Simulation environment	Major	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	Part of specification
HYDRA-93	Functional	Configurability IDE Middleware Layer	Re-playable event logging	Major	The HYDRA system should maintain a re-playable event log of all events and tasks relevant for a specific application and its set of related devices. It should be possible to parameterize the logging funcitonality regarding event types and time.	History list and event logging is automatically available after the application is deployed.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-92	Functional	Middleware Layer	Rule-based configuration of devices	Major	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. Possibility to hide device specific details.	The functionality (services) of a device is accessible (by user or application) thru a rule-based interface.	Part of specification
HYDRA-87	Functional	Context	AmI module	Minor	This module shall support discovery, security, devices ontology, and domain ontology among others to collect all ambient intelligence related issues in one module.	A module exist that is able to collect all ambient intelligence related issues supporting discovery, security, devices ontology, and domain ontology.	Part of specification
HYDRA-86	Functional	Modelling	Self adaptability	Critical	The middleware should automatically adapt to new situations, e.g. if another hardware component has been made available. This functionality needs to consider the influence of the work of already existing hardware	Automatic adaptation to new added /removed devices in 9 out of 10 cases	Part of specification
HYDRA-84	Functional	Configurability	Configuration abilities for developers	Major	This is important for a flexible development	At least 50 functionalities out of 100 documented functionalities of the device can be configured by the developers	Part of specification
HYDRA-83	Functional	Context	Adapt presentation to device capabilities	Major	Adjust and adapt content so that it is suitable for the devices capabilities or objects which are planned for displaying them	Every content adapts automatically to the resolution, screen size, and bandwith of the displaying device in 99,9% of all cases	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-82	Functional	Middleware Layer	Data Logging	Major	For system maintenance and debugging, logging functionality is mandatory.	HYDRA provides a logging component that can log system actions. Also, an API is available that enables developers to include logging in their applications.	Part of specification
HYDRA-80	Functional	Context	Proactivity	Major	The system should react proactively anticipating new operational circumstances taking into account information from different sources	????????Does the system can handle a limited set of expected situations	Ambiguous Requirement
HYDRA-76	Functional	Interface Security	Interfaces for user configuration	Major	To enable the user to configure his security settings, the system should provide appropriate interfaces.	HYDRA supports particular mechanisms in order to improve security usability in user interfaces	Requirement does not make sense
HYDRA-75	Functional	Configurability Devices Security	Auto configuration/ re-configuration	Major	To ensure, that the system is scalable, an auto-configuration of the security system must be provided. In this case auto-configuration means for example the adoption of the security policies by the entering device.	HYDRA allows to adopt security policies of entering devices to the GRID	Ambiguous Requirement
HYDRA-72	Functional	Modelling Security	Role-based access control	Major	An entity does not trust another entity in all cases. For example it may have trust in the abilities of another as a technician, but not as a doctor. Therefore, policies should regulate the entity's permissions depending on the current context.	HYDRA allows role-based access control modeling in order to distinguish specific levels of trust depending on the given context.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-70	Functional	Modelling SDK Security	The security system/model must be highly scalable.	Major	As nodes will be easily added to networks or trusted domains, the network itself will expand very easily. Therefore the security system must be scalable.	The security system continuously scales and allows support for both small and specific as well as large and general ambient lifestyle systems.	Requirement does not make sense
HYDRA-67	Functional	_unassigned	Support Wireless Security	Major	Since the majority of the communication in Hydra will be wireless, the hydra security model should enable the wireless security by supporting wireless security protocols.	Is the wireless communication secure in hydra?	Ambiguous Requirement
HYDRA-66	Functional	Security	Access control for context data	Major	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	Reopened
HYDRA-63	Functional	Communication Interface Networking Security	Remote access through distrusted networks	Major	As users are moving freely around in their environment, a secure remote access to their private data/devices at home has to be ensured. This may be realized through Virtual Private Networks (VPN).	HYDRA allows the integration of mechanisms that allow secure and confidential remote access to private information.	Ambiguous Requirement

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-60	Functional	Configurability Security	Delegation of access rights	Major	In case of eBilling and others, third party authorisation (i.e. delegation of access rights from a legitimate user to a different user of his choice) is necessary. These may happen through certificates and digital signatures.	Simulate a Billing Scenario and check whether a third party can do the transaction on behalf of users.	Part of specification
HYDRA-57	Functional	Context Security Service Discovery	Enable profiling	Major	To enable context-aware access, the security model/system must provide user profiling. This does not mean, that all the profile is open to everybody. The profile may be stored within the intimate domain, which is only accessible by the owner, and only information necessary will be passed out. For access control, only the credentials may be used for the profiling, which are not too private, as they are exchanged anyway.	HYDRA allows and supports profiling in order to enable service providers to serve personalised services to third parties (e.g. end-users).	Ambiguous Requirement
HYDRA-52	Functional	Security	Mechanisms for non-repudiation	Major	Especially for accounting information it is necessary to proof that a transaction took place.	Set up a scenario in which a communication partner can't repudiate a message he has sent.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-51	Functional	Communication Configurability Context Security	Private communication must be particularly secured	Major	Any private communication must not be monitored by any unauthorised third party.	HYDRA middleware has to provide particular mechanisms to protect communication indicated as 'private'.	Requirement does not make sense
HYDRA-50	Functional	Architecture Security	An identity management must be provided	Major	HYDRA middleware has to provide highly sophisticated mechanisms for identity management in order to ensure that in systems featuring HYDRA only authorised access to data, applications and devices is possible.	Identity management mechanisms are provided at all levels and to all stakeholders. Furthermore, the identification process of the managers must be uniform and standardised.	Reopened
HYDRA-48	Functional	Communication Middleware Layer Security	Support for multilateral communication involving several security protocols.	Major	The HYDRA security system should support multilateral communication involving several security protocols.	The HYDRA security framework supports mechanisms (e.g. as plug-in extension) to support multilateral communication between today's and future security protocols.	Ambiguous Requirement
HYDRA-46	Functional	Configurability Interface Security	End-User Configurability	Major	According to the so-called Beehive-Scenario end-users must be able to do some minor configuration by themselves (i.e for example user can introduce/ add a trusted device to his network).	HYDRA allows and supports the developer-user to integrate mechanisms in order to let the end-user do some minor configuration.	Ambiguous Requirement

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-44	Functional	IDE	IDE must provide support for Model Driven Architecture	Minor	The developer must be able to choose the appropriate software model for his/her project and hence the IDE must provide support for model driven architecture. The user must be able to select various models while starting his project. For example MVC architecture, Client-Server Model etc.	The user is able to select MVC Architecture for his new project.	Part of specification
HYDRA-43	Functional	IDE	Undo / Redo Feauture	Trivial	Just in the case of any other popular IDE, hydra IDE must also have a Undo/redo functionality so that the developer can go back to the previous state in case of an error.	The HYDRA IDE provides undo / redo functions.	Part of specification
HYDRA-42	Functional	IDE	Maintaining a History	Trivial	The IDE must maintain a History cache for the previous projects. It will make it easier for the developer to access the project which he/she was programming before and resume from where he/she left.	The user is able to view the history of his actions.	Part of specification
HYDRA-41	Functional	SDK	Hydra Developer's Companion	Major	Complete and comprehensable documentation is very important to the hydra software developer.	Complete documentation is available. It is at least considered "very helpful" by at least 8 out of 10 developers.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-40	Functional	IDE	IDE must be capable of deploying software to real devices.	Major	The IDE must support multiple interfaces with different devices, so that the developer can not only test his code on the simulation tool but also deploy it on the actual devices through the IDE. This might require the IDE to have device specific interfaces/ drivers.	Developers can deploy their application code on real devices via the IDE.	Part of specification
HYDRA-39	Functional	SDK	Cross compiling on different architectures	Minor	Hydra SDK must support cross compiling on different architectures	Cross compiling features are available in the IDE.	Part of specification
HYDRA-38	Functional	IDE SDK	Compiling & debugging feature	Major	Just like any other popular IDE, the Hydra IDE must be able to compile and debug the code.	Compiling & debugging functionality is available in the IDE.	Part of specification
HYDRA-37	Functional	IDE	Online Help / documentation with IDE	Minor	IDE must provide a help/ documentation so that the users can directly access the help pages to know more about the working of IDE or about deploying IDE and its various features.	Users are able to open & view help pages related to creating a new project and the corresponding steps from within the IDE.	Part of specification
HYDRA-36	Non-Functional - look and feel	IDE	Drag & Drop components	Minor	Drag & Drop functionality makes the programming easy for the developer	User is able to drag & drop components into the project.	Part of specification
HYDRA-34	Non-Functional - usability	IDE	The IDE must be easy to use .	Major	If the IDE is cluttered and complex, It will refrain the users from using Hydra Middleware	40 out of 50 users should find that the IDE is easy to use	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-31	Non-Functional - look and feel	SDK	An easy-to-use programming framework should be provided	Critical	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 IDE as intuitive.	Part of specification
HYDRA-30	Functional	IDE	Security Modelling to choose services and devices	Major	The developer should be able to choose predefined security modules he wants to use in his application. This could be done in a "Drag&Drop" way.	Tthe developer can include predefined software modules for security in his application.	Part of specification
HYDRA-29	Functional	IDE	IDE provides real-time hot plugging of software modules	Minor	The developer must be able to add modules/plug-ins and remove them from the IDE in real time.	The developer can add/delete software modules in real time.	Part of specification
HYDRA-28	Functional	IDE	Emulation / simulation tool is needed	Major	Developers need to test applications under reality-like conditions. IDE integrated software modules for real time evaluation of software components should be available.	Eemulation / simulation tools exist.	Part of specification
HYDRA-27	Non-Functional - usability	Configurability	Enable configuration for end-users	Major	Users want to configure the system and perform changes to the application with ease	90% of the end-users are able to change the behaviour of their application	Part of specification
HYDRA-26	Non-Functional - usability	Configurability	Central configuration	Major	In order to enhence the system's usability, all HYDRA components sould be managable over a single component.	The configuration and administration of a HYDRA system occurs via a central component.	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-25	Functional	Architecture Configurability Interface Middleware Layer Security	Overwriting system decisions	Critical	Possibly dangerous outcomes of system decisions must be overwritible by end-users	End-users can overwrite 90% of the application decisions	Part of specification
HYDRA-21	Constraint - assumption	Architecture	HYDRA should be a Service-Oriented Architecture (SOA)	Blocker	HYDRA should be a SOA per the Description of Work of the project	HYDRA is compatible to the SOA-definition by OASIS: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf	Part of specification
HYDRA-19	Constraint - scope of the product	Middleware Layer	Support of low-end devices	Major	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.	Part of specification
HYDRA-17	Constraint - requirement constraint	Architecture	When applicable, middleware interfaces are exposed by WSA-compatible services	Major	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.	Part of specification
HYDRA-14	Constraint - assumption	Device Discovery	Automatic device discovery	Critical	In order to be able to ad-hoc enter a device into an environment	From 100 devices brought into a new environment, at least 90 should be automatically discovered	Part of specification

Key	Requirement Type	Component/s	Summary	Priority	Rationale	Fit Criteria	Status
HYDRA-13	Functional	_unassigned	Communiction module - negotiate and establish suitable communication channel	Major			Open
HYDRA-10	Functional	_unassigned	Communiction module - Identify communication partners in the proximity like sensors etc.	Major			Open
HYDRA-9	Functional	_unassigned	Management of Message Queues	Minor			Open
HYDRA-8	Functional	Security	The middleware must support mechanisms for user authentication	Blocker	Different user groups with different access rights and responsibilities interact with complex distributed systems. HYDRA systems must be able to identify users and determine their access rights and their role in the system.	If necessary, HYDRA systems can identify system users in 100% of all cases.	Ambiguous Requirement