



Contract No. IST 2005-034891

Hydra

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

D4.1 Technical Requirements Specification

**Integrated Project
SO 2.5.3 Embedded systems**

Project start date: 1st July 2006

Duration: 48 months

**Published by the Hydra Consortium
International Ltd.**

2007-05-24 - version 1.0 Coordinating Partner: C

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

Dissemination Level: Confidential

Document file: D4.1 Technical Requirements.doc
Work package: [WP4 – Embedded AmI Architecture]
Task: [T4.1 – Technical Requirements]
Document owner: [Peter Rosengren (CNET)]

Document history:

Version	Author(s)	Date	Changes made
0.1	Klaus Marius Hansen	2007-03	Outline and initial contents
0.2	Klaus Marius Hansen Peter Kostelnik	2007-04-17	Added contributions from UAAR and TUKE
0.3	Klaus Marius Hansen Marius Scholten	2007-04-18	Added contribution from FIT
0.4	Klaus Marius Hansen Giuseppe Menta	2007-04-23	Added contributions from TCON
0.5	Matts Ahlsen	2007-05-01	Reformat + update of sec 3
0.6	Matts Ahlsen	2007-05-03	Update
0.7	Matts Ahlsen Peter Kostelnik	2007-05-04	Update
0.8	Klaus Marius Hansen	2007-05-07	QAS done.
0.9	Matts Ahlsen Peter Rosengren	2007-05-08	Device constraints reworked. Hydra UPnP device descriptions added. Version for internal review
0.95	Klaus Marius Hansen Matts Ahlsen	2007-05-22	Follow up and update on internal review 1
1.0	Klaus Marius Hansen Matts Ahlsen Peter Rosengren	2007-05-24	Follow up and update on 2 nd internal review
			Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Comments
Tomas Sabol Peter Butka	2007-05-14	Approved with comments
David Riou	2007-05-23	Approved with comments

1. Introduction	1
1.1 Purpose and context of this deliverable.....	1
1.2 Scope of this deliverable.....	1
2. Executive summary	2
3. Approaches to Embedded AmI	3
3.1 Embedded Devices in AmI	3
3.1.1 HYDRA-Related Projects	3
3.2 Embedded semantics.....	5
3.2.1 Approaches to describing and modelling devices	5
3.2.2 Device capability profiles	9
3.3 Conclusions / Implications	12
4. Devices in end-user Domains	13
4.1 Scenario: "Walking the Dog"	13
4.2 Scenario: Beehive	14
4.3 Scenario Easy Does It.....	15
4.4 Scenario: Daredevils.....	16
5. Device Performance Considerations	18
5.1.1 Device Communication and Control	18
5.1.2 Device Networking	19
6. Devices in first Demonstrator	21
6.1 Device Description for HVAC System	21
6.1.1 Device Description of Pump	22
6.1.2 Device Description of Thermometer (temperature sensor)	23
6.2 Additional demonstrator devices.....	23
6.2.1 Device Description for Lamp	24
6.3 Device Description for Door Lock.....	24
6.4 Device Description for Mobile Phone.....	25
7. Quality Attribute Requirements	26
7.1 D2.5 Requirements.....	26
7.1.1 WP3.....	26
7.1.2 WP4.....	27
7.1.3 WP5.....	27
7.1.4 WP6.....	27
7.1.5 WP7.....	28
7.2 Quality Attribute Scenarios.....	28
7.3 Utility tree.....	42
7.4 Discussion	49
8. Conclusion	50
9. References.....	51
10. Appendix Device Examples	53
10.1 Building Automation.....	53
10.2 eHealth	58
10.3 Agriculture	62

List of Figures

Figure 1: WP4 vs. a middleware stack [Schmidt, 2002]	1
Figure 2: FIPA device ontology (taken from [FIPA02]).....	6
Figure 3: OWL device ontology (taken from [Bandara 04])	7
Figure 4: Amigo device ontology (taken from [Amigo 06])	9
Figure 5: EBI Control system	15
Figure 6: Limbo Resource Utilization.....	18
Figure 7: Limbo Time Efficiency.	19
Figure 8: Hydra WP4 Utility.....	43
Figure 9: Hydra WP4 Utility - Functionality.....	44
Figure 10: Hydra WP4 Utility – Reliability	45
Figure 11: Hydra WP4 Utility – Usability.....	45
Figure 12: Hydra WP4 Utility – Efficiency	46
Figure 13: Hydra WP4 Utility – Maintainability	47
Figure 14: Hydra WP4 Utility – Portability	48
Figure 15: Hydra WP4 Utility – Quality in Use	49

List of tables

Table 1: Devices in scenario "Walking the Dog".	13
Table 2: Devices in scenario "Beehive".	14
Table 3: Devices in scenario "Easy does it!".	16
Table 4: : Devices in scenario "Dareddevils".	17
Table 5: HYDRA-356	28
Table 6: HYDRA-348	29
Table 7: HYDRA-337	29
Table 8: HYDRA-329	29
Table 9: HYDRA-324	30
Table 10: HYDRA-292.....	30
Table 11: HYDRA-234.....	31
Table 12: HYDRA-233.....	32
Table 13: HYDRA-209.....	32
Table 14: HYDRA-204.....	33
Table 15: HYDRA-201.....	33
Table 16: HYDRA-177.....	33
Table 17: HYDRA-170.....	34
Table 18: HYDRA-151.....	34
Table 19: HYDRA-146, HYDRA-314, HYDRA-267, HYDRA-144	35
Table 20: HYDRA-136.....	35
Table 21: HYDRA-86, HYDRA-365	36
Table 22: HYDRA-19, HYDRA-272	36
Table 23: HYDRA-373.....	36
Table 24: HYDRA-366.....	37
Table 25: : HYDRA-334	37
Table 26: HYDRA-334, HYDRA-318	38
Table 27: HYDRA-317.....	38
Table 28: HYDRA-315, HYDRA-193, HYDRA-369	38
Table 29: HYDRA-312.....	39
Table 30: HYDRA-380.....	39
Table 31: HYDRA-326, HYDRA-284, HYDRA-261	40
Table 32: HYDRA-359.....	40
Table 33: HYDRA-322.....	40
Table 34: HYDRA-143.....	41
Table 35: HYDRA-129, HYDRA-114	41
Table 36: HYDRA-297.....	42
Table 37: HYDRA-53	42

List of tables in Appendix

Table A 1.....	53
Table A 2.....	54
Table A 3.....	55
Table A 4.....	55
Table A 5.....	56
Table A 6.....	56
Table A 7.....	57
Table A 8.....	58
Table A 9.....	59
Table A 10.....	59
Table A 11.....	60
Table A 12.....	61
Table A 13.....	62
Table A 14.....	63

1. Introduction

1.1 Purpose and context of this deliverable

The objectives of this report are to specify and analyse the technical requirements that pertain to the Embedded AmI architecture of Hydra, developed by WP4.

The requirements are specified in the form of Quality Attributes scenarios and Utility Trees, following the Quality Attribute Framework introduced and applied in WP6 (Deliverable D6.1).

This deliverable is based on and extends the work in the following deliverables:

- D2.1 Scenarios for usage of Hydra in 3 different domains
- D2.2 Initial Technology Watch
- D2.5 Initial Requirements Report
- D3.1 Existing applications, services, devices and standards

1.2 Scope of this deliverable

The scope of this deliverable is the embedded AmI architecture under development in WP4. However, the technical requirements for this design are based on the total requirements set of Hydra reported in Deliverable D2.5 "Initial Requirements Report" and the initial usage scenarios of D2.1 "Scenarios for usage of Hydra in 3 different domains".

The report describes features and constraints of devices and technologies which can be used in the design and implementation of embedded AmI solutions in Hydra. The figure below shows how WP4 fits into a common characterization of middleware for embedded systems.

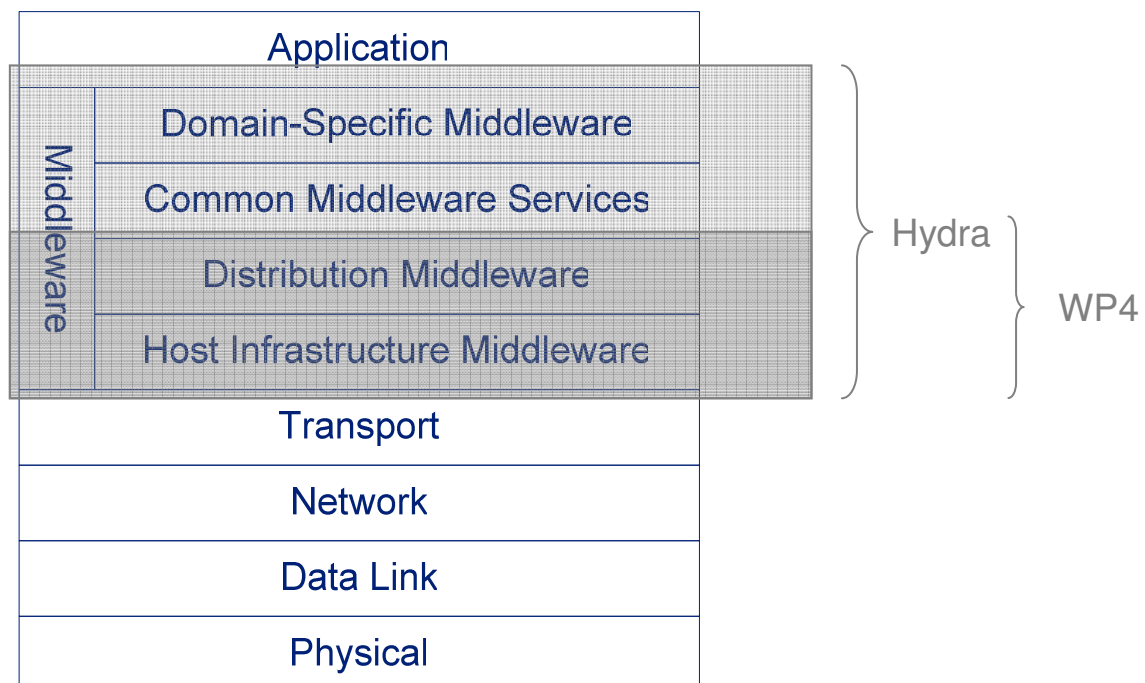


Figure 1: WP4 vs. a middleware stack [Schmidt, 2002]

2. Executive summary

The report describes features and constraints of devices and technologies which can be used in the design and implementation of embedded AmI solutions in Hydra.

We start by characterizing relevant approaches used to embed devices, services and semantics, as reported in other RTD projects and in standards initiatives. It extends deliverable D2.2 "Initial Technology Watch" and goes into specific details regarding embedded services and ambient intelligence. Locating a network service or a device on demand is a challenging task for enabling mobile and pervasive computing. A variety of semantic models or network protocols and architectures exist that enable an application to discover a network service with limited manual configuration.

From the initial state-of-the-art analysis we move into considering the end user domains of Hydra and the specific devices that we expect to be faced with. We here build on the work in deliverable D3.1 "Existing applications, services, devices and standards" which have surveyed and analysed a number of existing devices in the three end-user domains of Hydra – *Building Automation, eHealth* and *Agriculture*. In this deliverable we move one step forward and analyse the specific end-user scenarios described in deliverable D2.1 "Scenarios for usage of Hydra in 3 different domains", and for these scenarios we analyse which devices are needed and what kind of features we expect them to provide. This is done in chapter 4. In chapter 5 we also discuss and analyse performance issues regarding devices both in terms of resources and speed.

In chapter 6 we analyse the needs (in terms of devices) of the demonstrator to build for month 12 and the devices we have selected to integrate into the demonstrator. For each device we exactly define the services it will provide and the interface it will offer. This is done using the UPnP metadata descriptions that have been analysed in chapter 3.

With all this in mind we then in chapter 7 finally perform a quality attribute analysis mapping the relevant Hydra requirements to a set of quality attributes and quality attribute scenarios for the embedded AmI architecture in WP4. Chapter 8 concludes the deliverable by discussing implications of the requirements and initial decisions. Among the decisions are to use WSDL-based web services on a device level, to build upon UPnP (Universal Plug and Play), and to use a statemachine-based approach for self-* properties.

In the appendix we describe a number of typical devices for the Building Automation sector, which provides a more detailed description than in deliverable D3.1 "Existing applications, services, devices and standards".

3. Approaches to Embedded AmI

The main elements of the Hydra AmI architecture are based on approaches and technologies used to manage devices and services, to manage context, and to build and maintain the semantic descriptions of devices and services.

3.1 Embedded Devices in AmI

Ambient intelligent systems are built with manifold types of embedded systems. Special devices serve as user interfaces or sensors. They are equipped with some sort of interface that allows for communication with other devices. They can be provided with a certain intelligence that allows high-level communication with other devices and acting on certain events. Another type of devices has a general purpose and thus is usable in many ways. PDAs and cell phones (and of course notebooks and Personal Computers, which are not considered to be embedded devices) combine a variety of communication interfaces. They are easily programmable by wide-spread, well-known programming languages. A third group of embedded devices include facility management-, household- and consumer electronics devices. These are designed for a well-defined purpose that they can fulfil stand-alone. Additionally, they are equipped with some sort of programmable processing unit and communication interfaces that allow for integration into complex ambient intelligent systems. Embedded devices, that are in the focus of the HYDRA application domains, are already covered in deliverable 3.1. In the reminder of this section, we introduce different HYDRA related projects that aim at the creation of ambient intelligent systems by networking large numbers of varieties of embedded devices. All these projects have in common that they do not basically concentrate on the development of embedded devices for ambient intelligent applications. Instead, the central aspect is creating networks of embedded devices and building of intelligent applications by combining the devices' functionalities using well-known standards and frameworks.

3.1.1 HYDRA-Related Projects

3.1.1.1 AMIGO

The IST project AMIGO (IST-2004-004182) aims at the creation of ambient intelligent home systems by networking different types of embedded devices. The intelligence of AMIGO applications is realised by a middleware that exposes the functionality of connected devices as services. These are either.NET or OSGi services. Four classes of devices are considered: Personal Computers, mobile devices like cell phones, PDAs and notebooks, Home Automation devices and Consumer Electronics. A prototype of the AMIGO middleware specified in April 2007 includes low-level drivers for X10-, EIB- and BDF-based devices for building domotic services. More information on the above mentioned protocols and other protocols and standards in the building automation domain can be found in deliverable D3.1.

<http://www.hitech-projects.com/euprojects/amigo>

3.1.1.2 BETSY

BETSY (IST-2004-004042), an IST project that ended in March 2007, had the goal to enable mobile users to downstream multimedia based content to their portable devices as long as they have access to a wireless network. The quality of the media stream automatically adapts to restrictions on the provider- as well as on the client-side, to maximise availability of multimedia contents. The lower the quality of a stream, the more computational power, memory and battery power are saved on client side and network bandwidth is saved on the provider side. Some principles for quality may be applicable to Hydra.

Devices in a BETSY environment must have a wireless network interface and must be able to process MPEG video.

<http://www.hitech-projects.com/euprojects/betsy>

3.1.1.3 SIRENA

The ITEA SIRENA project (ITEA-02014), ended in November 2005, developed a framework to establish a Service-oriented Architecture on the link level. Devices that are connected to a network can expose their capabilities via service description interface and call other services. The framework resides directly on top of the Internet Protocol (IP) and is based on the DPWS (Device Profiles for Web Services) framework.

DPWS (<http://schemas.xmlsoap.org/ws/2006/02/devprof/>) is a further development of the UPnP device network protocol (see below). In SIRENA, the service advertisement and discovery facilities of UPnP are exploited and enhanced, while also adding security.

The application domains "Industrial Manufacturing", "Automotive", "Home" and "Telecom" are considered. A reference implementation of the SIRENA framework is available in JAVA.

In the project's context, "Schneider Electric", SIRENA's lead contractor, started a program called "Transparent Ready", that allows all their products to communicate and access each other. Schneider Electric builds products for the residential, building, industry, energy and infrastructure markets.

<http://www.sirena-itea.org/Sirena>

3.1.1.4 WASP

WASP (IST-2006-034963) has started in September 2006. This project aims at overcoming the mismatches between academic research and industrial needs in the field of wireless sensor networks. To achieve this goal, sensors, the communication and organisation of sensor nodes, the application of sensor networks and many more aspects are considered in three application domains: road transport, elderly care, and herd control. The project's (public summary) deliverable "D1.2 State of the Art", released in March 2007, gives an incomplete overview of the latest developments in the area of wireless sensors.

<http://www.hitech-projects.com/euprojects/wasp>

3.1.1.5 Smart Health

This project seeks to develop a network infrastructure referred to as the *Semantic Medical Device Space* described as a pervasive computing environment that exploits semantic web services and technology. The over-all objective is to facilitate medical device interoperability as well as integration with legacy health-care systems, which could promote solutions for home-care, monitoring and care chains.

The infrastructure connects *Ambient Intelligent (AmI) Medical Devices*, devices that are able to adapt to changes in their environment and capable of dynamical configuration for communication with other devices and systems, and, which are context-aware.

The functionality provided in the infrastructure include,

- device capabilities are exposed as web services
- service advertisement and discovery based on UPnP
- subscription and eventing
- explicit management of context

Smart Health is an IP (IST-2004-016817) which started in December 2005 with duration of 4 years.

<http://www.smarthealthip.com>

3.2 Embedded semantics

Services are generally viewed as functional components, which may be introduced within an open service environment, and subsequently located, invoked or composed. This type of functionality is usually implemented as software components; some services are also supported or offered by hardware devices. In this case, the characteristics and capabilities of the device play an important role in the description and behaviour of the offered services. The semantic device description enables it to present its properties and capabilities to other devices and can be used for inter-device communication, collaboration, content exchange, etc.

This section covers approaches to semantic modelling of devices using ontology-based knowledge models. The use of ontologies facilitates and supports service discovery, semantic reasoning and creates common standards for description of device capabilities.

3.2.1 Approaches to describing and modelling devices

3.2.1.1 FIPA Device Ontology

The Foundation for Intelligent Physical Agents (FIPA, www.fipa.org) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. To support agents when communicating about devices and exchanging device profile, FIPA specified a device ontology which contains properties of devices [FIPA 02]. These profiles can be then validated against the ontology. When two devices have a connection, they can exchange they profiles (directly or using some broker) in order to acquire the list of services provided by the devices. The profiles need to support identification of services for several device capabilities.

The frames provided by the specification represent the classes of objects in the domain of discourse within the framework of the FIPA device ontology. Example of the frame-based device model is shown in Figure 2. A device has the following properties:

- Device general description - basic information like device name, vendor details, device type, etc.;
- Software description - software resources like operating system information;
- Hardware description – CPU information, description of memory and connection units, description of the provided user interfaces. Each user interface specifies the audio I/O capabilities and the screen information (width, height, resolution, colour support, etc).

The values of some properties can change at any time (for example, if extra memory is inserted into device or another version of operating system is installed, the values of the corresponding parameters are changed).

The values of parameters can be further divided into static and dynamic, depending on the ability to change them in runtime. For example, agent compliancy and memory type description describing the memory available can change without booting the device, hence they are dynamic. On the other hand, screen description or CPU is static information, since it cannot change while the machine is running.

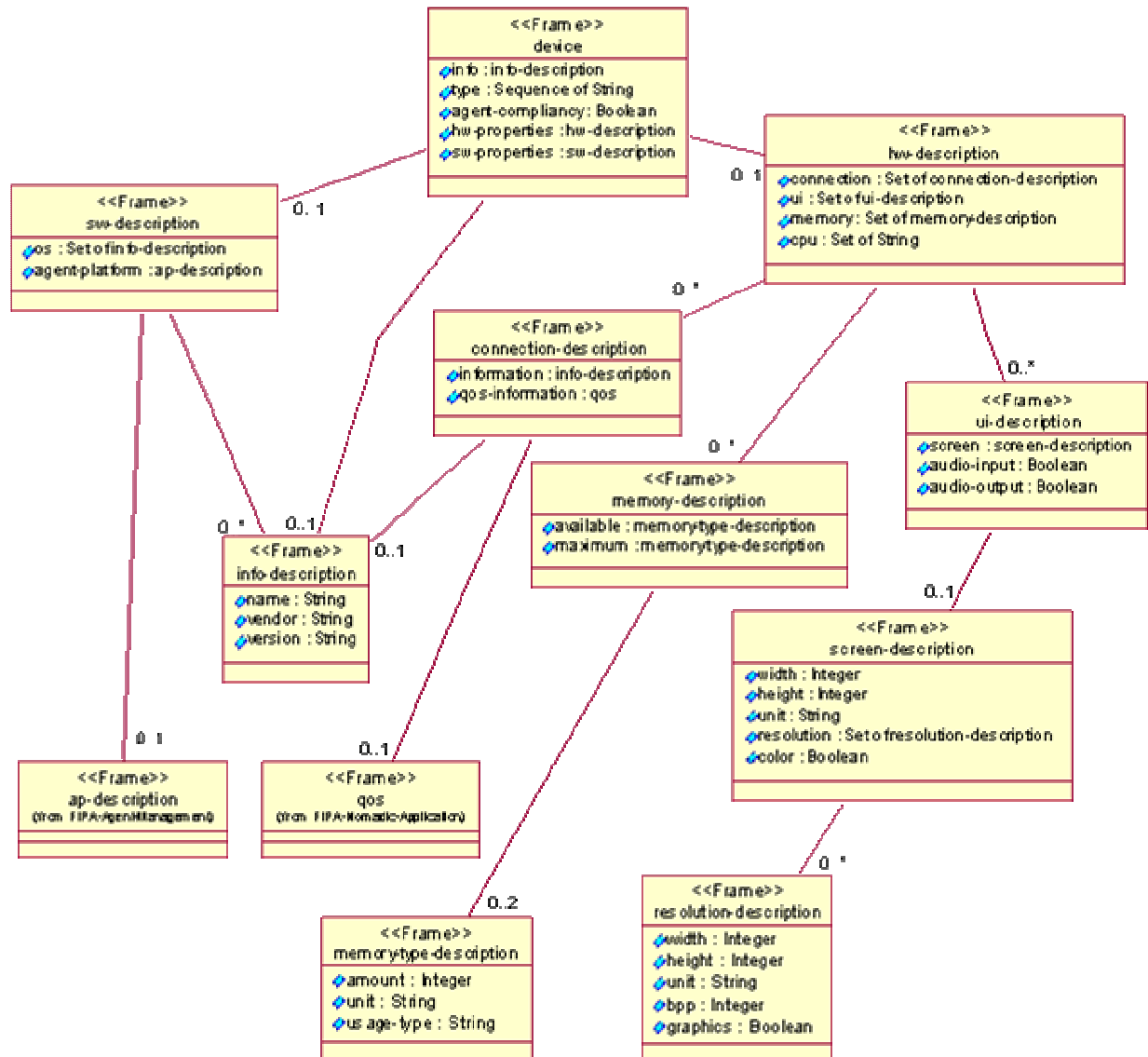


Figure 2: FIPA device ontology (taken from [FIPA02])

3.2.1.2 Ontological Framework for Semantic Description of Devices

Bandara et. al. claim in [Bandara 04] that several service discovery protocols exist for discovering devices in open, dynamic environments, but the characterization of these services is limited to static data structures, and implicitly defined keyword-based categorizations.

They assume that when a service involves device (for example printing service, scanning service), some level of detail about the hosted device will be required for service selection purposes. Such information related to the device could be included along with the service description itself in the service ontology, but having separate ontologies to describe devices and services should promote ease of use, readability and reusability and is therefore a better for system design. In certain cases of service composition, where devices are involved, it will be necessary to reason about the capabilities of available devices in order to determine a broker platform, where the execution and the coordination of the services take place. The broker platform may need to be selected based on the factors such as resource capability, proximity of the device to individual services, etc. In such cases the device ontology is useful for describing the capabilities of the devices available on the network.

Due to lack of declarative semantics, they propose an OWL-based ontology with the aim of providing a formal framework to describe devices and their services to support effective service discovery. Their proposal is shown in Figure 3.

Information related to a device is divided into five classes according to type of the provided information:

- Device Description contains basic information related to a device such as the device name, vendor details and the model of the device.
- Hardware Description contains the details about the hardware resources of the device, the details of its CPU, the connection to the network and memory.
- Software Description contains the details of the operating system of the device where relevant.
- The Device Status contains the details of its location, CPU usage and the power (method of power supply, whether its battery or mains and the remaining power level). The details of power supply and power level becomes important when it is necessary to determine the resource capability of a device. Location details will be required when service selection needs to consider the location of the device in choosing of the right service.
- The Service class provides information about the service(s) hosted on the device concerned. OWL-S could be potentially used to describe these individual services. There is a 1:n relationship between the Device class and the Service class. For example, if a particular device has a printer service, scanner service and a photocopy service, there will be three Service classes in the device ontology for this particular device.

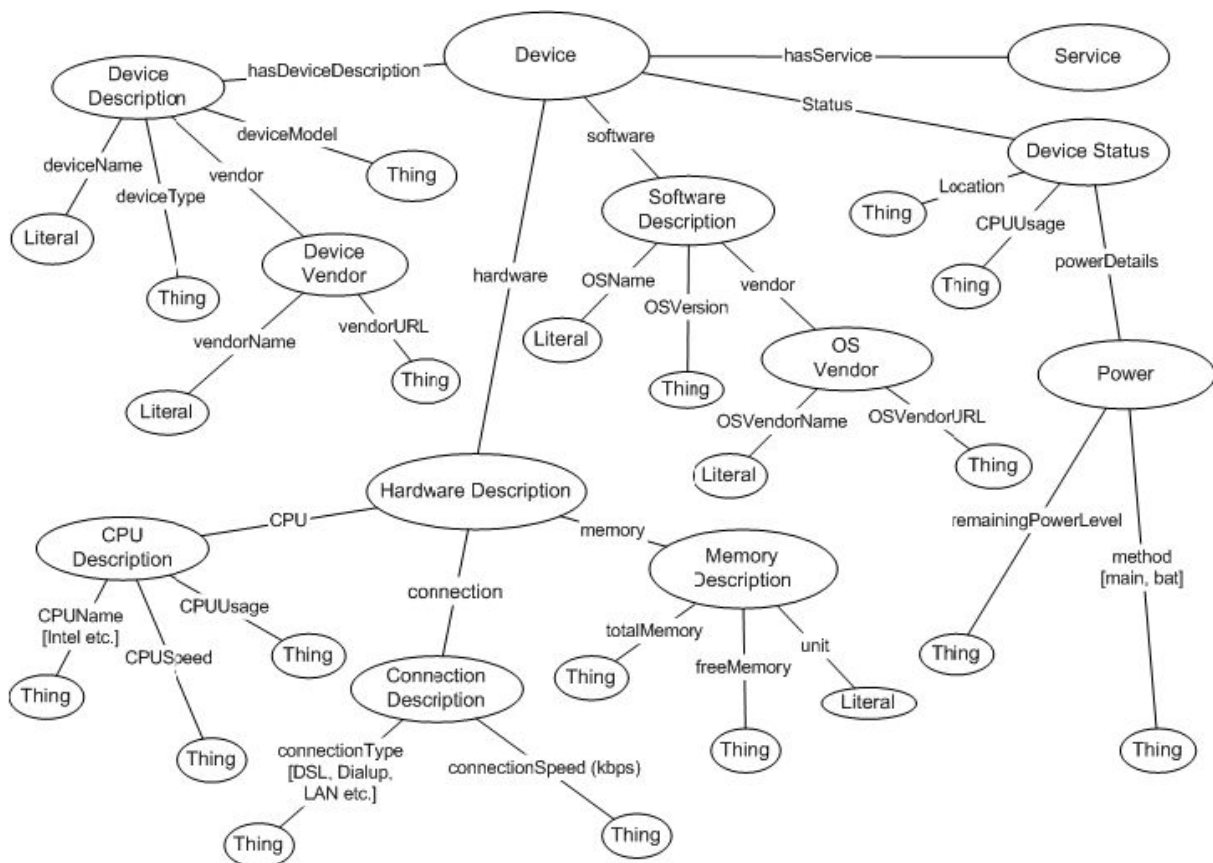


Figure 3: OWL device ontology (taken from [Bandara 04])

The Device Ontology is intended to provide a general framework to describe any type of device. But to describe specific types of devices more precisely, the concept of class hierarchies can be used. A hierarchy of sub-classes can be constructed using inheritance from the Device class to provide an effective device categorization. For example there can be a Printer sub-class which inherits from the device class and builds on additional properties (such as printer resolution etc.) necessary to

effectively describe printers. In the case, when a device does not fit into any of the available categories, or when it is not clear to which category the device belongs to, it could be specified as an instance of the Device class itself and thereby avoiding the use of the hierarchical classification.

3.2.1.3 Amigo Device Ontology

In the Amigo project [Amigo 06], the semantic modelling approach was adapted for modelling the services to enable efficient automated execution of the several tasks such a discovery, integration and composition of services in a ubiquitous and seamless manner.

In order to enable a rich representation of services and thus facilitate efficient service discovery and composition, functional capabilities, inputs, outputs and non-functional attributes of the services can be further semantically annotated using external vocabulary ontologies. The use of ontologies enables computational entities and services to have a common set of concepts and vocabularies for representing knowledge about a domain of interest, while being able to interact with each other. By Using of such ontologies, where relationships between entities can be more clearly expressed, leads to better reasoning on their properties. Ontologies are also beneficial for the re-use of knowledge, as several ontologies from various sources can be integrated to describe the specific domain.

The modularisation of service description vocabularies is based mainly on the specificity of the concepts defined by the vocabulary and can be classified into three levels:

1. The concepts defined by a generic ontology are considered to be generic across many fields (top-level ontology).
2. Core ontologies define concepts which are generic across a set of domains (descriptions of capabilities, devices, context, multimedia content, etc.).
3. Domain ontologies express conceptualizations that are specific for a particular universe of discourse. The concepts in domain ontologies are often defined as specializations of concepts in the generic and core ontologies (for example descriptions of the requirements of devices in home automation, consumer electronics, mobile, personal computing domains, etc.).

The main classification of generic platforms and devices are combined into the platform and device vocabularies (as the part of core ontologies). The device vocabularies provide a classification on platforms hosted by devices and generic classification of device types and their states. Device vocabulary is shown on Figure 4.

Although these concepts can be used to characterize the device context, these vocabularies are still very generic. Basic device ontology can be specifically extended with concepts of specific domains, such as consumer home automation, electronics, mobile or pc domain. The structure of concepts can be integrated with other standards such as FIPA device ontology.

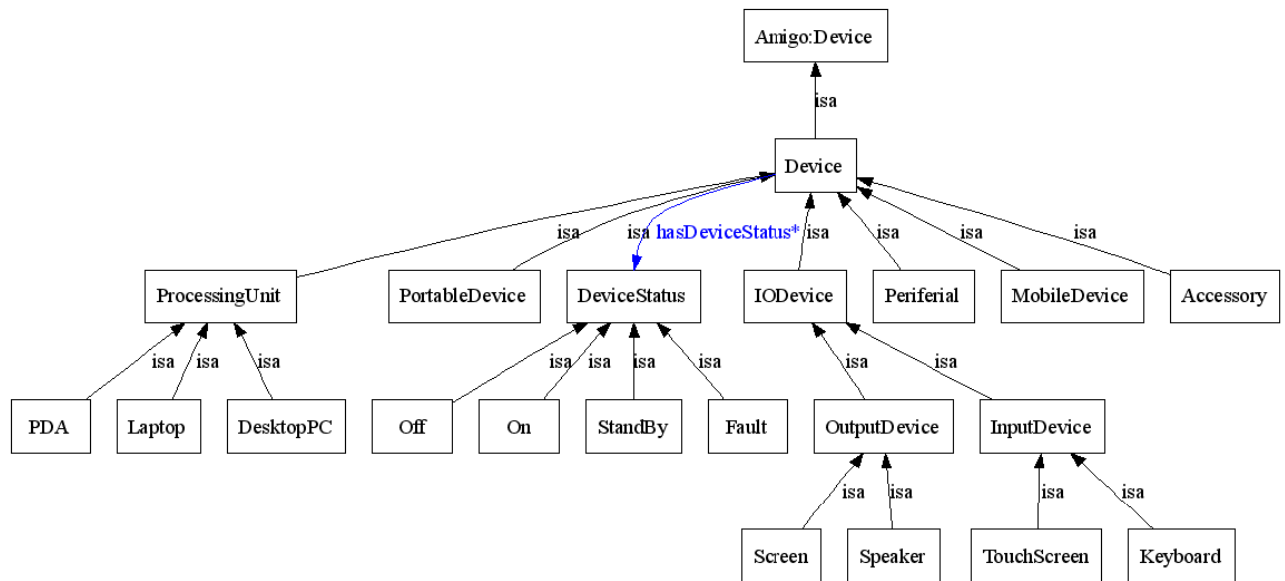


Figure 4: Amigo device ontology (taken from [Amigo 06])

3.2.2 Device capability profiles

This section reviews how several existing standards try to address specific use cases for inter-device interaction. These standards all rely on devices being able to describe their capabilities to other devices. In order to achieve this, each standard defines its own profile structure and profile vocabulary. Furthermore applications using this information often need to perform standard tasks such as selection, generation or adaptation based on this information.

3.2.2.1 CC/PP

The W3C has specified a data structure and sample vocabulary for profiles, which can convey delivery of context information. The current Composite Capabilities/Preferences Profile (CC/PP) [CCPP] is based on the original XML serialized Resource Description Framework (RDF).

The CC/PP structure is vocabulary independent and allows the use of one or more vocabularies, optionally described using RDF Schema.

As CC/PP is vocabulary neutral, it allows different vocabularies to be developed and implemented by communities involved in developing applications, devices and browsers. It also allows for the dynamic composition of a delivery context profile from fragments of capability information that may be distributed among multiple repositories on the web.

CC/PP is the preferred approach to communicating delivery context information between clients, intermediaries and origin servers. It is the basis for UAProf [UAProf 07], which is currently used to express the capabilities of many mobile devices. There are a number of implementations available which consume CC/PP profiles, and there is also a Java Community Process interface definition for profile consumers.

The current CC/PP Recommendation provides a structure and a sample vocabulary based on the version of RDF current during its development. It is expected to be brought up to date with the latest revisions of RDF and RDF Schema, and to be extended to include support for capabilities asserted by intermediate proxies and gateways. Further work is also required to specify a protocol for exchanging CC/PP profiles, and to specify how a profile should be processed and used within any mechanism for content adaptation.

3.2.2.2 UAPProf

The Open Mobile Alliance (OMA) has defined a User Agent Profile as an implementation of CC/PP for WAP-enabled mobile terminals, providing convergence of mobile web technologies with those of the Web.

WAP 1.2.1 recommends transporting UAPProf information over the Internet using the HTTP Extension Framework which was originally suggested for CC/PP. WAP defined the WSP protocol, which includes a compressed encoding, for use between the phone and the gateway onto the Internet. Due to the lack of implementations of HTTPex, WAP 2.0 instead defined an extension of HTTP 1.1 as an Internet protocol (W-HTTP) that uses custom headers.

The WAP Forum also defined a UAPProf vocabulary, based on CC/PP that includes hardware and software characteristics as well as WAP specific features of the mobile terminal and associated transport network. Subsequent updating to UAPProf V2.0 [UAPProf 07] by OMA has based the definition on the latest version of RDF and RDF Schema.

3.2.2.3 WURFL

WURFL [WURFL], is a free, open source project that provides an alternative source of information to UAPProf (<http://wurfl.sourceforge.net/>). It tries to provide a comprehensive resource of device information, and contains device information for 4500 variants of devices. Because WURFL is open source, anyone can correct device information, not just device manufacturers. WURFL provides its own XML format for device characteristics description.

3.2.2.4 Universal Plug and Play (UPnP)

UPnP [UPnP] is architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to provide easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. UPnP is a distributed, open networking architecture that leverages TCP/IP and the Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices in the home, office, and public spaces.

UPnP is more than just a simple extension of the plug and play peripheral model. It is designed to support zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices. DHCP and DNS servers are optional and are used only if available on the network. Finally, a device can leave a network smoothly and automatically without leaving any unwanted state behind.

UPnP leverages Internet components, including IP, TCP, UDP, HTTP, and XML. Like the Internet, contracts are based on wire protocols that are declarative, expressed in XML, and communicated via HTTP. IP internetworking is a strong choice for UPnP because of its proven ability to span different physical media, to enable real world multiple-vendor interoperability, and to achieve synergy with the Internet and many home and office intranets. UPnP has been explicitly designed to accommodate these environments. Further, via bridging, UPnP accommodates media running non-IP protocols when cost, technology, or legacy prevents the media or devices attached to it from running IP.

UPnP does not require device drivers, common protocols are used instead. UPnP networking is media independent. UPnP devices can be implemented using any programming language, and on any operating system. UPnP does not specify or constrain the design of an API for applications running on control points; OS vendors may create APIs that suit their customer's needs. UPnP enables vendor control over device UI and interaction using the browser as well as conventional application programmatic control.

UPnP uses XML in order to provide a structure for describing device capabilities and service descriptions. Currently UPnP does not specify specific device vocabularies; instead it is expected that device manufacturers will devise these vocabularies themselves.

3.2.2.5 Media Queries

In W3C recommendations, such as CSS and HTML, style mark-up can be made conditional on delivery context by using Media Queries [MQ 02]. Media Queries introduce another vocabulary for accessing device characteristics.

Media Queries build upon the use of 'media types' as defined in CSS2, which allow styles to be conditional on a number of named categories of device: aural, braille, embossed, handheld, print, projection, screen, tty and TV. In Media Queries, device characteristics ('media features') may be queried and combined using restricted expression syntax. The style used to present an element of HTML, XHTML or XML can therefore be made conditional on the characteristics of the delivery device. By making use of the CSS 'display' property, it is also possible to conditionally include or exclude complete elements from the presentation.

In the future, it should be therefore possible to add device-dependent styling to common device-independent content, at least as far as the CSS media types and media features will allow. Like CSS, Media Queries are typically expected to be processed directly in user agents, based on local delivery context information. However, they could also be fully or partially processed at servers or intermediaries in the response path, based on delivery context information passed as part of a request for content. This highlights the need for the vocabulary used for the device capabilities passed in the delivery context to correspond to the vocabulary used within Media Queries.

3.2.2.6 SMIL

A further W3C standard, the Synchronized Multimedia Integration Language (SMIL) [SMIL 01] introduces yet another vocabulary for accessing a limited number of device characteristics. SMIL 2.0 is defined as a set of mark-up modules that can be integrated into specific language profiles. In particular, it defines a BasicContentControl Module that defines certain system characteristics that may be used to control a SMIL presentation. These characteristics may be given dynamic values according to the runtime environment. Like Media Queries, they therefore allow a user agent, that supports dynamic SMIL characteristics, to access local delivery context information.

System test characteristics, included as part of the SMIL BasicContentControl Module, cover presentation-related capabilities such as screen size, network bandwidth, and text and audio captions, as well as system-related characteristics such as CPU and operating system identity.

3.2.2.7 SyncML

The SyncML Initiative [SyncML] aims to develop a common synchronization protocol for data between mobile devices and servers, i.e. it addresses the seamless device synchronization use case. Devices such as phones only support a limited number of applications: for example most have an address book and some have diaries. SyncML can be used to synchronize entries in these applications, but it is envisioned that it could be used for potentially any file type. Crucially when two devices undergo synchronization, they have to exchange a description of their capabilities. This is done using the SyncML Device Information (DevInf) standard, which is based on XML but uses a vocabulary created specifically for DevInf.

3.2.2.8 Media Feature Sets

Media Feature Sets (MFS) [MFS 99] were proposed by the Internet Engineering Task Force (IETF) to allow devices to describe their capabilities to servers when retrieving web content, exchanging fax messages or emails, i.e. seamless device independence. Like the other standards, MFS define a syntax allowing a complex description of capabilities and requirements, but allows vendors to define vocabularies via a tag registration procedure. Unlike the standards previously mentioned, MFS is not based on XML. This is a disadvantage as it is easier to process files based on XML due to easy availability of XML parsers. One of underlying design decisions of MFS is that device capabilities can be regarded as constraints. As a result it allows these constraints to be explicitly joined using Boolean operators such as AND, OR and NOT. This is very useful as it can be used for description also when device has different capabilities for the same vocabulary property depending on the

modality or the particular mode of operation. For example, it might be useful to describe that a PDA has different capabilities depending on whether it is being used in landscape or portrait mode.

3.2.2.9 TCN

Transparent Content Negotiation was first proposed as an experimental protocol in RFC 2295 [TCN 98].

Transparent negotiation uses both HTTP server-driven and agent-driven negotiation mechanisms, together with a caching proxy that supports content negotiation. The proxy requests a list of all available representations from the origin server using agent-driven negotiation, then selects the most appropriate and sends it to the client using server-driven negotiation. However, this technique has not been widely implemented.

3.2.2.10 MPEG-21

ISO/IEC is defining the MPEG-21 [MPEG 02] framework which is intended to support transparent use of multimedia resources across a wide range of networks and devices. The fundamental unit of distribution is the 'digital item', which is an abstraction for some multimedia content with associated data.

One aspect of the requirements for MPEG-21 is Digital Item Adaptation which is based on a Usage Environment Description. It proposes the description of capabilities for at least the terminal, network, delivery, user, and natural environment, and notes the desirability of remaining compatible with other recommendations such as CC/PP and UAProf.

3.2.2.11 Wireless Village Initiative

The Wireless Village Initiative [WVI] seeks to define and promote a set of universal specifications for mobile instant messaging and presence services aimed at mobile devices, mobile services and Internet-based instant messaging services. Like all the other standards described here, Wireless Village also defines its own device capability description format and vocabulary. Here the device capability format is based on XML and it is noticeable that there is some overlap in the vocabulary and the UAProf and SyncML vocabularies: for example all three define the MIME types that a device supports.

3.3 Conclusions / Implications

In this section we have looked at how current efforts and technologies can be used to support the different features of embedded AmI architectures, i.e.,

- Embedded devices
- Context Awareness
- Embedded Services
- Embedded Semantics

Locating a network service or a device on demand is a challenging task for enabling mobile and pervasive computing. A variety of semantic models or network protocols and architectures exist that enable an application to discover a network service with limited manual configuration.

Several approaches build on and extend the UPnP connectivity architecture, which also is our initial choice for the Hydra middleware. Adopting UPnP implies that the project is able to apply existing design toolkits (e.g., from Intel) which can readily be used to design first demonstrator versions of Hydra device discovery and control protocols. Further, there are example device descriptions available from manufacturers and the various UPnP initiatives

The prevailing approaches to the modelling of devices and their capabilities employ ontologies using OWL/OWL-S, as well as the common semantic web (WS-*) standards.

4. Devices in end-user Domains

There are a number of considerations regarding devices that needs to be taken into account. Firstly, a major challenge will be to master the multitude and heterogeneity of devices that we will have to face. A second challenge will be to manage the performance constraints when dealing with small devices and web service technology.

The work described in this section builds on the scenario description completed for D2.1 "Usage scenarios for usage of HYDRA in 3 different domains" and on the technology overview presented in D3.1 "Existing applications, services, devices and standards". In the latter a survey was done over available devices in the three end-user domains of Hydra – Building Automation, eHealth and Agriculture. In this section we will specifically study what type of devices is needed in the different scenarios described in D2.1.

As a starting point for the following project phases it was decided by partners that the first requirements analysis should be based on the Building Automation domain. For this reason we decided to focus the present work on Building Automation scenarios in particular with a detailed analysis and to propose a more general approach for the other two end-user domains.

In Building automation we tried to identify the main systems and devices mentioned together with some of the features described. With this picture in mind we then searched for actual devices and systems matching the application context of each scenario identifying their (available) features and constraints.

The following paragraphs characterize each of the Building Automation scenarios from a device usage view. The appendix in Section 10 then provides additional details on the various devices referenced for each of the scenarios.

4.1 Scenario: "Walking the Dog"

As we went through the first scenario described in D2.1, "Walking the dog", we were able to draw Table 1 to sum up main devices and features mentioned:

Main Systems and Devices	Hydra Features
- Pumps	<p>Embedded systems provide means for controlling and optimising operating conditions</p> <p>External applications must know and are allowed to read data from the pumps and pumping systems</p> <p>The external requestor must present the necessary credentials before obtaining the information and there must be a trust model</p> <p>There's the need for automatic discovery and configuration of the pumps</p>
- Early warning system	<p>Easy interface to local and national systems, sensor networks meteorological and oceanographic surveillance systems, such as low altitude aerial and satellite based meteorological and oceanographic surveillance systems as well as several large-scale computer modelling systems.</p>

Table 1: Devices in scenario "Walking the Dog".

To define features and constraints in relation to pumps we based our analysis on the devices produced by Grundfos (www.grundfos.com), one of the biggest European manufactures in this field,

First of all, the project has experience with these types of pumps and secondly Grundfos is one of the most advanced pump manufacturers, among others involved in the ZigBee consortium.

In particular Grundfos produces MAGNA Series 2000 circulator pumps which are electronically controlled.

The MAGNA/UPE ranges of circulator pumps are specially designed for:

- heating systems up to 2100 kW ($\Delta t = 20^{\circ}\text{C}$) and
- domestic hot-water systems (stainless-steel or bronze pump housing).

details on this device are provided in Table A 1 in the Chapter 10 Appendix.

4.2 Scenario: Beehive

The second scenario is "Beehive" and introduces to the features and constraints description of some devices for building automation applications (Table 2).

Main Systems and Devices	Hydra Features
<ul style="list-style-type: none"> - Monitoring system for technical installations - Heating system - Electrical distribution - Cooling system - Water supply - Wastewater 	
<ul style="list-style-type: none"> - In-house lighting system - Burglar alarms - Cameras - heating systems - Door locks - Blinds 	Devices should allow users to access remotely and control things Every authorised vendor, subcontractor and service organisation should have to access the system with a new authentication procedure
<ul style="list-style-type: none"> - mobile devices 	Automatic update of daily tasks.
<ul style="list-style-type: none"> - Building systems 	The building systems report directly to the manufacturer central system. All systems in the building interoperate.
<ul style="list-style-type: none"> - security clearance system 	Automatic recognition of users
<ul style="list-style-type: none"> - mobile web tablet 	Information and data about every device is available everywhere. Every device can be tested, accessed and software can be upgraded from the web.

Table 2: Devices in scenario "Beehive".

The building automation technology provided by Honeywell is interesting to be mentioned being a solution that supports a wide range of systems and industry standards and property protocols, including:

- LonMark®
- BACnet®
- OPC
- Modbus® RTU
- Allen-Bradley PLCs
- Advanced Dynamic Data Exchange (ADDE)
- LonWorks® Network Services (LNS)
- Honeywell EXCEL 5000® environmental controls

EXCEL 5000® System Architecture

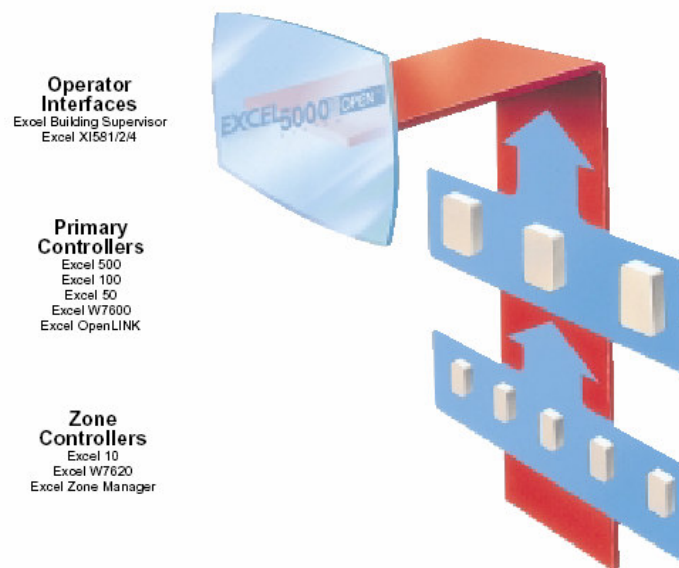


Figure 5: EBI Control system

This Building automation control system is built on different layers which create a networked system with no additional submodules or separate communication devices required.

Within a networked system, the so called "primary controllers" can serve as modem devices for the whole network. They can also be applied as a stand-alone controller with real-time clock.

The architecture is modular and can be expanded with new parts or systems supported to create a small control system (heating of a house) or a large/distributed control system (many buildings).

Reference devices in this scenario include a Primary Controller and a Zone Controller described in Table A 2 and Table A 3. The former is a freely programmable control and monitoring system designed specifically for building management, the latter is a lower level control unit which can be dedicated to different control functions: heat pumps control, unit ventilator controller, variable air volume controller etc.

Finally we also include a small temperature sensor (Table A 4) which can communicate with the controllers through the LONWORKS open standard.

4.3 Scenario Easy Does It

A third scenario "Easy does it!" introduces into the healthcare context which is the main theme in Healthcare scenarios. Anyway we decided to proceed here with a device constraints and actual features analysis.

Main Systems and Devices	Hydra Features
- Cognitive alarm system	Automatically interprets dynamic situations; adapts to different topologies, infrastructures, sensors; find and automatically interface to telephones, hearing aids, or any other device containing a microphone; is able to semantically process the captured sound.
- Wearable heart monitor device	Device supports wireless communication
- Cognitive monitoring system	Automatically detects and connects to available BP monitors Data can be stored in the relevant Electronic Patient Record
- Blood glucose monitor	A wireless device which can look for communication access points The device can communicate the stored measurements and detected abnormalities to the relevant health professionals The device can automatically identify and securely send requests to a diabetes care centre if needed
- Insulin box	The box records every time the insulin pen is removed The box is able to search for a suitable proxy, which can provide both communication and computing support

Table 3: Devices in scenario "Easy does it!".

For this scenario we present a collection of monitoring devices used in the healthcare sector. They have interesting features in terms of small power requirements and support for communication means. Specifically we describe:

- a wireless oximetry (Table A 8)
- a vital sign monitor (Table A 9)
- a blood pressure monitor (Table A 10)
- a scale (Table A 11)
- a sensor based communications platform for vital signs monitoring (Table A 12)

4.4 Scenario: Daredevils

The final scenario for building automation relates to some sample devices for home automation.

Main Systems and Devices	Hydra Features
- Automatic cat feeding device	
- Delivery box	Biometric security: ID card or biometric device. They could be combined also with a voice recognition system
- Refrigerator	Accessible on the Internet via a secure network connection
- Lawnmower	Automatic, provided with a remote control using a PDA. It supports its own wireless drivers but with open interfaces.

- Window cleaner	A robotic device, provided with a remote control using a PDA. It supports its own wireless drivers but with open interfaces.
- Electronic cookbook	Voice recognition, Web services support and online delivery to appliances
- Stove	Automatic system networked with other home devices
- Household goods	Networked and provided with value added services
- House alarms	Connected to the smart home system
- Outside door camera	Can store pictures on the house server
- Washing machine	Can connect with RFID tags on the clothes
- Built-in speaker system	
- Sprinkler system	
- humidity sensors, temperature sensors, sunlight sensor	Self-configurable sensors with wireless connection
- Electrical systems for controlling the windows and shades	Uses rules based decision support and relies on external sensors for micro-weather monitoring

Table 4: : Devices in scenario "Daredevils".

For this scenario we describe some home automation devices which present innovative solutions and support different type of sensors.

We have chosen an advanced home automation controller produced by Homeseer (Table A 5) which presents some cutting edge solutions that can be compared with the ones presented in the scenario.

Furthermore we describe a home motion sensor (Table A 6) supporting Z-Wave communication.

5. Device Performance Considerations

There are two different set of performance requirements regarding devices that need to be considered:

- *Device Communication and Control.* Performance when communicating directly with the device, i.e. invoking operations on the device, getting responses from the device and other types of communication, what is involved in terms of web service invocation?
- *Device Networking.* Performance issues raised by the infrastructure or network around the device, e.g., what are the requirements on a device for being discovered by others, to discover other devices, and once discovered what it takes to stay known in the environment.

5.1.1 Device Communication and Control

One of the major design goals of Hydra is to represent each device as a web service, running either as a proxy or directly on the device. An important issue then is how efficient and fast these web services can execute.

To be able to investigate this we are developing the *Limbo* compiler. The compiler will produce target code based on a Web Service Description Language (WSDL) description of the service combined with a semantic description of the device type, characteristics, resources etc. A reasonable outset for semantics is the device ontology of [Bandara 04]. The device ontology will furthermore be used in order to generate code that report resources on the device (using the Event Manager functionality as defined in D3.3 "Draft Architecture Design Specifications").

Preliminary experiments have been made with Limbo. Figure 6 below shows a comparison between Limbo ("DX") and two commonly used SOAP toolkits (Apache Axis and kSOAP) for the invocation of a simple web service. Correspondingly, Figure 7 shows comparison in terms of time use.

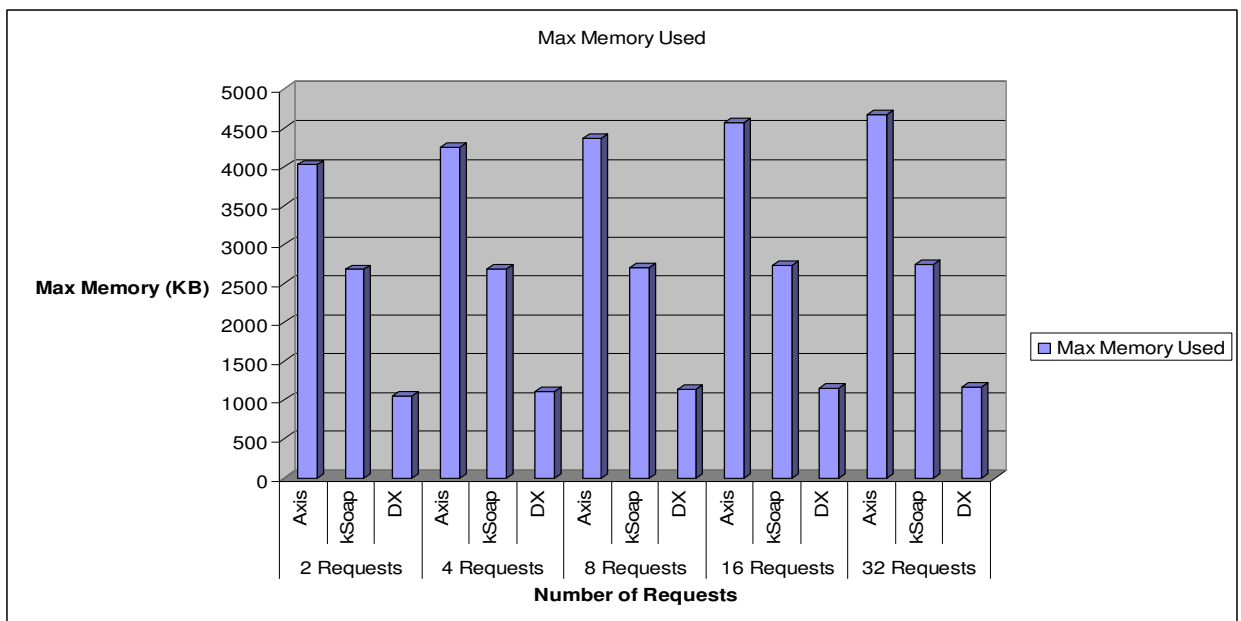


Figure 6: Limbo Resource Utilization.

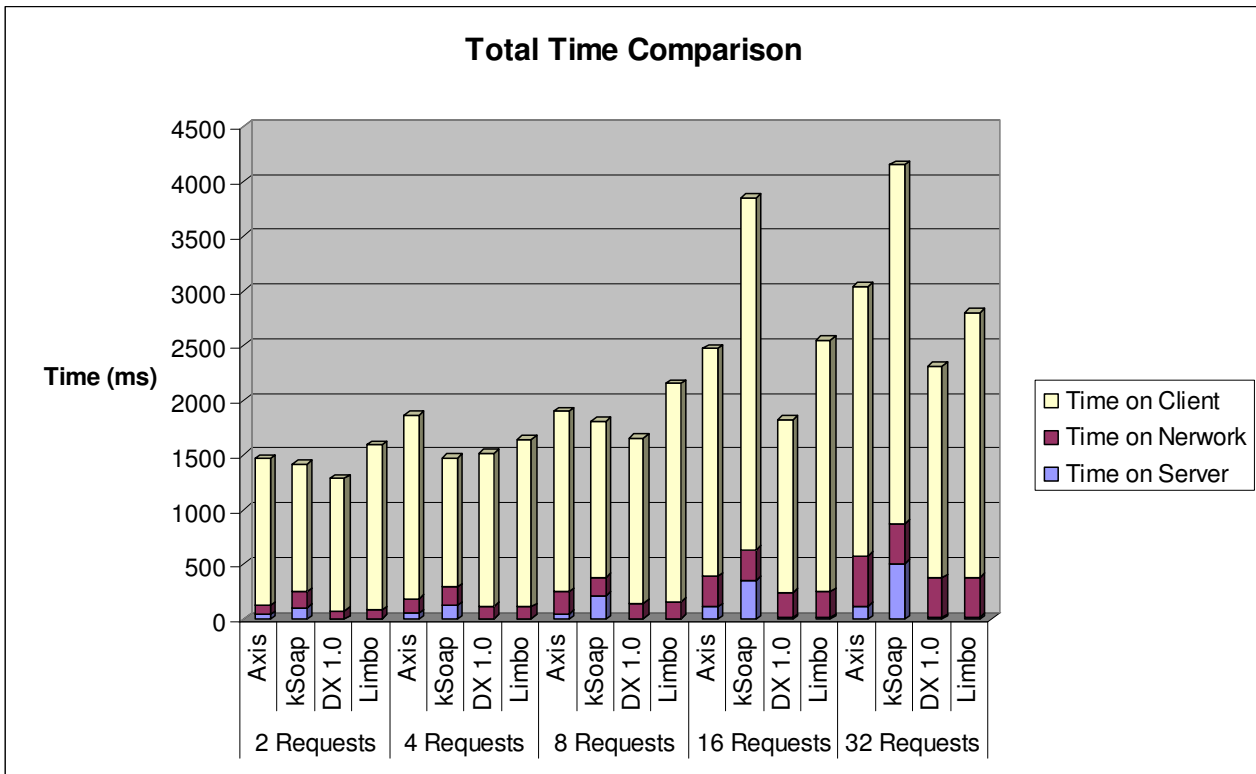


Figure 7: Limbo Time Efficiency.

This clearly shows that a specialised tool that has been designed specifically for generating web services for device has the possibility of outperforming more generic tools.

5.1.2 Device Networking

The diagram above illustrates efficiency aspects (time and memory used) when an application or another device is communicating directly with a device. In addition to this we also need to consider a number of other performance aspects which have to do with the environment/network of the device:

- How many resources does the discovery mechanism consume, i.e., what is required of the device to be discovered?
- How much resources are required of the device to remain as part of the network?
- How many simultaneous network connections can a device keep open?

We have investigated the UPnP architecture for device discovery and communication to get an understanding of the performance issues.

A key concept in UPnP is very compact and efficient UPnP stacks. One way to accomplish this is by tightly designing and controlling how the processor handles various UPnP tasks. *Microstacks* are designed to use a single thread to handle all of the UPnP network tasks. This approach does not only lead to greatly reduced runtime size, but also to improved performance when compared to multi-threaded stacks.

The design is simple, a single thread waits (blocks) on a select call, waiting for one or more events to occur. Once the select call unblocks, events are handled and the stack goes back to the select statement, waiting for the next event. In each loop, small tasks such as reading data from a socket, timeouts, search requests, and much more, are handled and state is updated.

A common misconception with single threaded software is that it can't handle multiple network connections, this is not true. A single device Microstack will handle all of the following with a single thread:

- One HTTP server socket
- Up to 5 HTTP sessions, these sessions are used for:
 - SOAP action invocation
 - Event subscription & management
 - Presentation page transfers (Web, images, media, streaming media, etc.)
- One SSDP¹ listen socket per network interface
- Up to 5 outbound event connections
- Timers for various management functions

From the network's point of view, all of the connections are handled simultaneously, and data is received and handled on all of the connections at the same time, in reality, a single thread does all of the work. This design is also built in such a way that the single thread will only loop if there is actual events to handle, otherwise, it will block for long periods of time. This is of great benefit to devices with CPU power management.

The single thread that is used to run the Microstack must be provided by the developer, when the start (ILibStartChain) method is invoked. The developer can adjust the thread priority and other settings of the thread before making the call to start the stack.

If the developer intends to perform a time consuming task as a result of a Microstack event, arguments of the event should be copied and a different thread should be used to handle processing of the event. If the Microstack thread is used to perform a time consuming task such as waiting for a large file to load, it is no longer free to handle UPnP events, and the stack will appear to freeze.

Microstacks vary in size depending on how they are implemented but they can be made as small as 60 Kbytes and still be functional.

¹ Simple Service Discovery Protocol

6. Devices in first Demonstrator

For the first demonstrator at project month 12 (described in deliverable D3.3) we intend to incorporate the following devices:

- HVAC system consisting of
 - Pump
 - Thermometer
- Door lock
- Mobile Phone
- Lamp

For each of them we give an UPnP device description, which describes the different services a device provides. Each service is described in a service description file.

6.1 Device Description for HVAC System

The UPnP definition of a HVAC system states:

“A HVAC system includes all the heating and cooling equipment necessary to independently condition a whole house or a region of a house. Large homes may include several independent systems. Small homes may only have one system. Individual systems may be zoned.

A forced air system may typically include a furnace with heat exchangers and a fan or a blower, an air conditioning compressor unit, filters, duct work, temperature sensors, temperature set points, mode controls and in zoned systems electrically controllable valve or dampers.

Hydronic systems are usually zoned and may typically include boiler, cooler, pumps, valves, radiators, temperature sensors, temperature set points and mode controls.

HVAC_System is a container for all the elements of a system. It is not necessarily a recognizable single physical device. All the elements are not necessarily exposed. This DCP is targeted at user control and does not expose operational details.

HVAC system may include independent controls for sub-regions or zones. This is called a Zoned HVAC System. HVAC_System is the root device and includes system level services and one or more HVAC_ZoneThermostat Devices. The Thermostats provide the zone level controls. Binding of the Thermostats to the System is implied by the hierarchical design of the System device.”

To simplify we will use a pump and a thermometer to represent the HVAC system in the demonstrator. The pump will be a Grundfos pump (described in the appendix 10.1).

```
<device>
<deviceType>urn:schemas-upnp-org:device:HVAC:1</deviceType>
  <friendlyName>IVT</friendlyName>
  <manufacturer>IVT</manufacturer>
  <manufacturerURL>http://www.ivt.se</manufacturerURL>
  <modelDescription>HVAC</modelDescription>
  <modelName>Grundfos Magna</modelName>
  <modelName>X1</modelName>
  <UDN>uuid:dac824ab-bca1-4d5c-93c5-578a0c697ba1</UDN>

  <serviceList>
    <service>
```

```

    <serviceType>urn:schemas-upnp-
org:service:HVAC_UserOperatingMode:1</serviceType>
    <serviceID>SystemUserMode</serviceID>
  </service>
</service>
  <Optional/>
  <serviceType>urn:schemas-upnp-
org:service:HVAC_FanOperatingMode:1</serviceType>
    <serviceID>SystemFanMode</serviceID>
  </service>
</service>
  <Optional/>
  <serviceType>urn:schemas-upnp-org:service:FanSpeed:1</serviceType>
    <serviceID>SystemFanSpeed</serviceID>
  </service>
</service>
  <Optional/>
  <serviceType>urn:schemas-upnp-org:service:TemperatureSensor:1</serviceType>
    <serviceID>OutsideTemperature</serviceID>
  </service>
</service>
  <Optional/>
  <serviceType>urn:schemas-upnp-
org:service:HVAC_SetpointSchedule:1</serviceType>
    <serviceID>SystemSetpointSchedule</serviceID>
  </service>
</serviceList>
<deviceList>
  <device>
    <deviceType>urn:schemas-upnp-org:device:HVAC_ZoneThermostat:1</deviceType>
  </device>
</device>
  <deviceType>urn:schemas-upnp-org:device:Pump:1</deviceType>
</device>
</deviceList>
</device>

```

6.1.1 Device Description of Pump

The following is the basic description for the pump component device,

```

<device>
  <deviceType>urn:schemas-upnp-org:device:waterPump:1</deviceType>
  <friendlyName>GrundfosPump</friendlyName>
  <manufacturer>Grundfos</manufacturer>
  <manufacturerURL>http://www.grundfos.com</manufacturerURL>
  <modelDescription>Pump</modelDescription>
  <modelName>Grundfos Magna</modelName>
  <modelNameNumber>2000</modelNameNumber>
  <UDN>uuid:dac824ab-bca1-4d5c-93c5-578a0c697ba1</UDN>
  <serviceList>

```

```

    <service>
      <serviceType>urn:schemas-upnp-
org:service:grundfosPumpService:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:grundfosPumpService</serviceId>
      <SCPDURL>_grundfosPumpService_scpd.xml</SCPDURL>
      <controlURL>_grundfosPumpService_control</controlURL>
      <eventSubURL>_grundfosPumpService_event</eventSubURL>
    </service>
  </serviceList>
</device>

```

6.1.2 Device Description of Thermometer (temperature sensor)

The UPnP definition of a thermometer system states:

"This service allows a temperature read from a temperature sensor. Control points or other devices may set and get an application value for this service. The following applications are defined:

- Room – Indoor room temperature
- Outdoor – Outdoor temperature
- Air Duct – Temperature inside an air duct
- Pipe – surface temperature of a pipe.

Manufacturers shall establish the allowable range of temperatures using the maximum and minimum allowed values. A Control Point or other device can find these values in the XML description. Control points or other devices may optionally establish a Name for this sensor."

```

<device>
  <deviceType>urn:schemas-upnp-org:device:Thermometer:1</deviceType>
  <friendlyName>Thermometer</friendlyName>
  <manufacturer>CNet Svenska AB</manufacturer>
  <manufacturerURL>http://www.cnet.se</manufacturerURL>
  <modelDescription>Measures temperature</modelDescription>
  <modelName>Thermometer</modelName>
  <modelName>X1</modelName>
  <UDN>uuid:74535361-f6af-4b4d-8e48-1cf2eaa863e5</UDN>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-org:service:Thermometer:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:CNetValueAddedThermometer</serviceId>
      <SCPDURL>_CNetValueAddedThermometer_scpd.xml</SCPDURL>
      <controlURL>_CNetValueAddedThermometer_control</controlURL>
      <eventSubURL>_CNetValueAddedThermometer_event</eventSubURL>
    </service>
  </serviceList>
</device>

```

6.2 Additional demonstrator devices

Finally we show the device descriptions for the lamp, door lock and mobile phone.

6.2.1 Device Description for Lamp

The UPnP definition of a lamp system states:

"The lighting control device is comprised of a Switch Power and a Dimming Service. The switching component is serially connected to the dimming component. The dimmer's settings will not be physically realized by the load until the switch is on.

The effect of the dimmer when the switch switches from "off" to "on" state is optionally programmable (OnEffect) and is triggered by the Switch Power Service's state. This means if the Status of the Switch Power Service switches from 0 to 1 then the OnEffect will be realized on the dimming component.

In OnEffect the default state of the dimming device is set. If OnEffect set to OnEffectLevel the LoadLevelTarget is set to OnEffectLevel when Target is changed to 1."

```
<device>
  <deviceType>urn:schemas-upnp-org:device:BinaryLight:1</deviceType>
  <presentationURL>/</presentationURL>
  <friendlyName>Light (FUGLESANG)</friendlyName>
  <manufacturer>Intel Corporation</manufacturer>
  <manufacturerURL>http://www.intel.com</manufacturerURL>
  <modelDescription>Software Emulated Light Bulb</modelDescription>
  <modelName>Intel CLR Emulated Light Bulb</modelName>
  <modelNameNumber>XPC-L1</modelNameNumber>
  <modelURL>http://www.intel.com/xpc</modelURL>
  <UDN>uuid:6f1fc427-5476-43a0-8406-822de4848a56</UDN>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-org:service:DimmingService:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:DimmingService.0001</serviceId>
      <SCPDURL>_urn:upnp-org:serviceId:DimmingService.0001_scpd.xml</SCPDURL>
      <controlURL>_urn:upnp-org:serviceId:DimmingService.0001_control</controlURL>
      <eventSubURL>_urn:upnp-
org:serviceId:DimmingService.0001_event</eventSubURL>
    </service>
    <service>
      <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:SwitchPower.0001</serviceId>
      <SCPDURL>_urn:upnp-org:serviceId:SwitchPower.0001_scpd.xml</SCPDURL>
      <controlURL>_urn:upnp-org:serviceId:SwitchPower.0001_control</controlURL>
      <eventSubURL>_urn:upnp-org:serviceId:SwitchPower.0001_event</eventSubURL>
    </service>
  </serviceList>
</device>
```

6.3 Device Description for Door Lock

```
<device>
  <deviceType>urn:schemas-upnp-org:device:CNetDoor:1</deviceType>
  <friendlyName>CNet Door</friendlyName>
  <manufacturer>CNet</manufacturer>
  <manufacturerURL>http://www.cnet.com</manufacturerURL>
```

```

    <modelDescription>Sample UPnP Device Using Auto-Generated UPnP
    Stack</modelDescription>
    <modelName>CNet Door Model 1</modelName>
    <modelNumber>X1</modelNumber>
    <UDN>uuid:e553abad-bd53-4fa0-8263-dd8d16c7d82b</UDN>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service::1</serviceType>
        <serviceId>urn:upnp-org:serviceId:CNetDoor.001</serviceId>
        <SCPDURL>_CNetDoor.001_scpd.xml</SCPDURL>
        <controlURL>_CNetDoor.001_control</controlURL>
        <eventSubURL>_CNetDoor.001_event</eventSubURL>
      </service>
    </serviceList>
  </device>

```

6.4 Device Description for Mobile Phone

```

<device>
  <deviceType>urn:schemas-upnp-org:device:SMSService:1</deviceType>
  <friendlyName>NokiaPhone</friendlyName>
  <manufacturer>Nokia</manufacturer>
  <manufacturerURL>http://www.cnet.se</manufacturerURL>
  <modelDescription>SMS service</modelDescription>
  <modelName>Nokia N80</modelName>
  <modelNumber>N80</modelNumber>
  <UDN>uuid:0b4cbea2-fe21-47f9-b59a-5e8f7b76314a</UDN>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-org:service:SendSMSService:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:NokiaSmsService</serviceId>
      <SCPDURL>_NokiaSmsService_scpd.xml</SCPDURL>
      <controlURL>_NokiaSmsService_control</controlURL>
      <eventSubURL>_NokiaSmsService_event</eventSubURL>
    </service>
  </serviceList>
</device>

```

7. Quality Attribute Requirements

In the following sections, we derive requirements for WP4 using the same approach as applied in deliverable D6.1 "Quality Attribute Workshops". In doing so, we end up in creating a utility tree of quality attribute scenarios based on three sources:

- Requirements from the two previous sections
- Requirements from D2.5 classified as belonging to WP4
- Requirements from D2.5 not classified as belonging to WP4 but that have direct implications for WP4

The first section categorizes the D2.5 requirements according to ISO 9126 qualities (detailed in deliverable D6.1) if applicable. Then we present the utility tree for WP4 in Hydra.

7.1 D2.5 Requirements

7.1.1 WP3

[HYDRA-356] support for both a pull and push model (Portability/Adaptability)

[HYDRA-348] Detect errors in devices (Reliability/Fault Tolerance)

[HYDRA-337] UAAR: There should be a procedure/strategy for interfacing with non-HYDRA devices (Functionality/Interoperability)

[HYDRA-329] Middleware provides domain-independent services (Quality in Use/productivity)

[HYDRA-324] Systems built using HYDRA should be scalable in terms of devices communicating (Efficiency/Resource utilisation)

[HYDRA-323] Distributed Intelligence should not lead to resource-heavy systems (Efficiency/Resource utilisation)

[HYDRA-292] Self-diagnosis of devices (Maintainability/Analysability)

[HYDRA-234] The middleware should be self-descriptive (Usability/Learnability)

[HYDRA-233] Self-healing function of middleware (Reliability/Recoverability)

[HYDRA-209] Middleware has a service for providing information about the technical environment/infrastructure (Maintainability/Analysability)

[HYDRA-204] Devices have automatic error diagnostics (Maintainability/Analysability, Reliability/Fault tolerance)

[HYDRA-201] Self configurable devices (Portability/Installability)

[HYDRA-177] Dynamic scheduling of resource usage (Efficiency/Resource utilisation)

[HYDRA-170] Stateful and stateless communication (Efficiency/Resource utilisation, Functionality/Suitability)

[HYDRA-151] Devices send events when their status changes (Efficiency/Time behaviour)

[HYDRA-146] Report errors in devices (Reliability/Fault tolerance)

[HYDRA-136] Dynamic architecture (Portability/Adaptability)

[HYDRA-86] Self adaptability (Portability/Adaptability)

[HYDRA-19] Support of low-end devices (Efficiency/Resource utilisation)

[HYDRA-17] When applicable, middleware interfaces are exposed by WSA-compatible services

7.1.2 WP4

- [HYDRA-373] Semantically locate devices and information providers (Portability/Replaceability)
- [HYDRA-366] Web services should run on embedded devices (Efficiency/Resource utilisation)
- [HYDRA-334] UUAR: There should be support for developing auto-configuration of certain devices (Portability/Installability)
- [HYDRA-318] UAAR: Devices should be able to be added to the system at runtime (Portability/Installability)
- [HYDRA-317] UUAR: Support runtime reconfiguration (Portability/Adaptability)
- [HYDRA-315] UAAR: Devices should be able to contain user interface/management interface (Quality in Use/Effectiveness)
- [HYDRA-314] UAAR: Faults should be intercepted by middleware, notified to interested services (Reliability/Fault tolerance)
- [HYDRA-312] UAAR: Support profiling of devices' performance (Usability/Understandability)
- [HYDRA-311] UAAR: Special watchdog devices for monitoring availability
- [HYDRA-309] UAAR: Map device (e.g., name of it) to presentation of, e.g., a room
- [HYDRA-193] Support for natural interaction (Quality in Use/Effectiveness)
- [HYDRA-192] Context modelling
- [HYDRA-191] Intelligent location determination
- [HYDRA-190] Learning situational context
- [HYDRA-171] Learning user behaviour patterns

7.1.3 WP5

- [HYDRA-380] Ability to send a broadcast message to wake up sleeping devices (Efficiency/Resource behaviour)
- [HYDRA-371] Devices classes hierarchy
- [HYDRA-326] UAAR: Support (residential) gateway devices at least at physical level (Functionality/Interoperability)
- [HYDRA-284] Legacy components (Functionality/Interoperability)
- [HYDRA-272] Small devices (Efficiency/Resource utilisation)
- [HYDRA-271] Remove devices
- [HYDRA-267] Failure detection (Reliability/Fault tolerance)
- [HYDRA-261] Bridges between different technologies (Functionality/Interoperability)
- [HYDRA-144] Detect defective connection (Reliability/Recoverability)

7.1.4 WP6

- [HYDRA-369] Devices must have semantic description of its user interface (Quality in Use/Effectiveness)
- [HYDRA-365] Ability to self-adaptation (Portability/Adaptability)
- [HYDRA-359] Device ontology versioning (Portability/Co-existence)
- [HYDRA-322] UUAR: Support lightweight service composition (Efficiency/Resource utilisation)
- [HYDRA-143] Model-based reasoning about itself (Maintainability/Analysability)

[HYDRA-129] Support for Semantic Web Standards for Device Communication (Functionality/Suitability)

[HYDRA-114] Semantic enabling of device web services (Functionality/Suitability)

[HYDRA-113] Composition (of services and devices) (Functionality/Suitability)

[HYDRA-110] Device Categorisation

[HYDRA-98] Detection of device failures (Reliability/Fault tolerance)

[HYDRA-91] Any HYDRA device should have an associated description

7.1.5 WP7

[HYDRA-297] Secure data erasing on HYDRA enabled devices (Functionality/Security)

[HYDRA-239] Automatic service diagnostic for security relevant services (Functionality/Security)

[HYDRA-53] The availability of devices should be guaranteed (Reliability/Maturity)

7.2 Quality Attribute Scenarios

The requirements listed in Section 7.1 are in the following analysed and presented as Quality Attribute Scenarios. This approach is defined and further detailed in Deliverable D6.1.

Scenario(s):		[HYDRA-356] support for both a pull and push model
Relevant Quality Attributes:		Portability/Adaptability
Scenario	Stimulus Source:	Environment changes so that service needs to use a pull-based data exchange model instead of a push-based
	Stimulus:	Internal
	Artefact (If Known):	Component
	Environment:	Operation
	Response:	Pull-based data exchange is possible
	Response Measure:	Possible in 90% of cases without delay in processing
Questions:		
Issues:		

Table 5: HYDRA-356

Scenario(s):		[HYDRA-348] Detect errors in devices
Relevant Quality Attributes:		Reliability/Fault tolerance
Scenario	Stimulus Source:	Internal
	Stimulus:	Failure
	Artefact (If Known):	Device
	Environment:	Operation
	Response:	Failure is detected

	Response Measure:	90% of cases
Questions:		What are the precise circumstances under which 90% of failures can be detected? Who detects?
Issues:		

Table 6: HYDRA-348

Scenario(s):		[HYDRA-337] UAAR: There should be a procedure/strategy for interfacing with non-HYDRA devices
Relevant Quality Attributes:		Functionality/Interoperability
Scenario	Stimulus Source:	Developer
	Stimulus:	Wants to enable operation of non-Hydra device
	Artefact (If Known):	System
	Environment:	Operation (of SDK)
	Response:	Operation is enabled
	Response Measure:	75% of cases
Questions:		
Issues:		This relates to WP5 and all-IP communication. Implies that we need bridges to non-IP technology

Table 7: HYDRA-337

Scenario(s):		HYDRA-329] Middleware provides domain-independent services
Relevant Quality Attributes:		Quality in Use/productivity
Scenario	Stimulus Source:	Developer user
	Stimulus:	Needs to develop building maintenance scenario
	Artefact (If Known):	Hydra-based system
	Environment:	Operation (of SDK)
	Response:	Building maintenance scenario is implemented
	Response Measure:	A major part of needed services are available from Hydra
Questions:		
Issues:		This is of course a common characteristic of middleware. Initial definition of middleware services are outlined in D3.3

Table 8: HYDRA-329

Scenario(s):	[HYDRA-324] Systems built using HYDRA should be scalable in terms
---------------------	---

	of devices communicating [HYDRA-323] Distributed Intelligence should not lead to resource-heavy systems
Relevant Quality Attributes:	Efficiency/Resource utilisation
Scenario	Stimulus Source: User
	Stimulus: Wants to perform task
	Artefact (If Known): Device
	Environment: Normal operation
	Response: Task is performed
	Response Measure: Performed with reasonable delay 100.000 Hydra-based devices
Questions:	What is reasonable? The QAS only gives a broad estimate on the number of devices to be supported
Issues:	Can only be supported in a decentralized manner

Table 9: HYDRA-324

Scenario(s):	[HYDRA-292] Self-diagnosis of devices
Relevant Quality Attributes:	Maintainability/Analysability
Scenario	Stimulus Source: Device
	Stimulus: Failure
	Artefact (If Known): Device
	Environment: Operation
	Response: Detects and reports failure
	Response Measure: Failure capability efficiency is 98%
Questions:	What if (e.g.) 5% of all failure cases are network failures?
Issues:	

Table 10: HYDRA-292

Scenario(s):	[HYDRA-234] The middleware should be self-descriptive
Relevant Quality Attributes:	Usability/Learnability
Scenario	Stimulus Source: Developer
	Stimulus: Wants to learn to produce Hydra-based systems
	Artefact (If Known): SDK

	Environment:	Development time
	Response:	Understands Hydra middleware
	Response Measure:	After one week of experience, 9 out of 10 times
Questions:		
Issues:		

Table 11: HYDRA-234

Scenario(s):		[HYDRA-233] Self-healing function of middleware
Relevant Quality Attributes:		Reliability/Recoverability
Scenario	Stimulus Source:	Internal
	Stimulus:	Failure
	Artefact (If Known):	Of service
	Environment:	In operation
	Response:	Failure is intercepted by system, performance is re-established
	Response Measure:	Restore effectiveness is at least 65%
Questions:		Where does 65% come from?
Issues:		

Table 12: HYDRA-233

Scenario(s):		[HYDRA-209] Middleware has a service for providing information about the technical environment/infrastructure
Relevant Quality Attributes:		Maintainability/Analysability
Scenario	Stimulus Source:	System component
	Stimulus:	Needs to diagnose cause of communication delay
	Artefact (If Known):	System
	Environment:	Online
	Response:	Cause is identified through monitoring service
	Response Measure:	In general 95% of the technical infrastructure is monitored
Questions:		How is percentage of infrastructure measured?
Issues:		

Table 13: HYDRA-209

Scenario(s):		[HYDRA-204] Devices have automatic (error) diagnostics
Relevant Quality Attributes:		Maintainability/Analysability, Reliability/Fault tolerance
Scenario	Stimulus Source:	Service
	Stimulus:	Requests status of a device
	Artefact (If Known):	Device
	Environment:	Operation
	Response:	Receives status

Response Measure:	90% of cases a valid status
Questions:	What is "status" here?
Issues:	

Table 14: HYDRA-204

Scenario(s):	[HYDRA-201] Self configurable devices	
Relevant Quality Attributes:	Portability/Installability	
Scenario	Stimulus Source:	A user
	Stimulus:	Turns on a new device
	Artefact (If Known):	Device
	Environment:	Operation
	Response:	Device can communicate, is configured
	Response Measure:	No/minimal intervention by user in 80% of cases
Questions:		
Issues:	Some minimal intervention is required, e.g., for security	

Table 15: HYDRA-201

Scenario(s):	[HYDRA-177] Dynamic scheduling of resource usage	
Relevant Quality Attributes:	Efficiency/Resource utilisation	
Scenario	Stimulus Source:	System
	Stimulus:	Services that are part of the application use resources in excess of what is provided on the devices on which they run
	Artefact (If Known):	System
	Environment:	Normal operation
	Response:	Service resource use rescheduled, operation provided
	Response Measure:	Rescheduling in 80% of cases if a suitable substitute is available
Questions:		
Issues:	Rescheduling depends on mechanisms for this; will have to be investigated Need awareness of resources for this	

Table 16: HYDRA-177

Scenario(s):	[HYDRA-170] Stateful and stateless communication
Relevant Quality	Efficiency/Resource utilisation

Attributes:		
Scenario	Stimulus Source:	Developer
	Stimulus:	Wants to switch between stateless and stateful communication
	Artefact (If Known):	Component
	Environment:	Development time
	Response:	Switch supported
	Response Measure:	
Questions:		
Issues:		

Table 17: HYDRA-170

Scenario(s):		[HYDRA-151] Devices send events when their status changes
Relevant Quality Attributes:		Efficiency/Time behaviour
Scenario	Stimulus Source:	Device/service
	Stimulus:	Needs to know status of other device
	Artefact (If Known):	Device
	Environment:	Normal operation
	Response:	Status is known
	Response Measure:	If state has changed recently, device/service is notified via events
Questions:		
Issues:		

Table 18: HYDRA-151

Scenario(s):		[HYDRA-146] Report errors in devices [HYDRA-314] UAAR: Faults should be intercepted by middleware, notified to interested services [HYDRA-267] Failure detection [HYDRA-144] Detect defective connection [HYDRA-98] Detection of device failures
Relevant Quality Attributes:		Reliability/Fault tolerance Reliability/Recoverability
Scenario	Stimulus Source:	Internal
	Stimulus:	(Non-fatal) error

	Artefact (If Known):	Device/service
	Environment:	Operation
	Response:	Error is reported by device
	Response Measure:	If error is detected, and reporting is possible, reporting is done
Questions:		
Issues:		

Table 19: HYDRA-146, HYDRA-314, HYDRA-267, HYDRA-144, HYDRA-98

Scenario(s):		[HYDRA-136] Dynamic architecture
Relevant Quality Attributes:		Portability/Adaptability
Scenario	Stimulus Source:	Internal
	Stimulus:	System architecture needs to be changed from centralized to decentralized
	Artefact (If Known):	System
	Environment:	Maintenance
	Response:	System architecture is changed through dynamic migration
	Response Measure:	In 95% of cases
Questions:		
Issues:		'Dynamic migration' needs to be refined. For now we will assume weak mobility

Table 20: HYDRA-136

Scenario(s):		[HYDRA-86] Self adaptability [HYDRA-365] Ability to self-adaptation
Relevant Quality Attributes:		Portability/Adaptability
Scenario	Stimulus Source:	External
	Stimulus:	New hardware component available
	Artefact (If Known):	Service and hardware
	Environment:	Operation
	Response:	System adapts itself to new hardware; driver is downloaded
	Response Measure:	Adaptation to added/removed devices in 90% of cases
Questions:		
Issues:		

Table 21: HYDRA-86, HYDRA-365

Scenario(s):		[HYDRA-19] Support of low-end devices [HYDRA-272] Small devices
Relevant Quality Attributes:		Efficiency/Resource utilisation
Scenario	Stimulus Source:	Developer
	Stimulus:	Wants to build Hydra-enabled application
	Artefact (If Known):	Device
	Environment:	Development
	Response:	Application is deployed and can run on device in a Hydra-enabled system
	Response Measure:	Can be installed on low-end 32 bit devices (with less than 512 KB RAM), proxies otherwise
Questions:		
Issues:		The (necessary) lower limit of Hydra-enabling should be explored

Table 22: HYDRA-19, HYDRA-272

Scenario(s):		[HYDRA-373] Semantically locate devices and information providers
Relevant Quality Attributes:		Portability/Replaceability
Scenario	Stimulus Source:	Internal
	Stimulus:	Device running a service needed by an application is removed
	Artefact (If Known):	System
	Environment:	Operation
	Response:	Application updated to use similar service
	Response Measure:	Update is done automatically Match is done based on needed semantics of service Similarity can be specified by developers
Questions:		
Issues:		

Table 23: HYDRA-373

Scenario(s):		[HYDRA-366] Web services should run on embedded devices
Relevant Quality Attributes:		Efficiency/Resource utilisation
Scenario	Stimulus Source:	Middleware user
	Stimulus:	Implements service on device

	Artefact (If Known):	Device
	Environment:	Development time
	Response:	Service is implemented and runs
	Response Measure:	Service runs as web service on embedded device
Questions:		
Issues:		

Table 24: HYDRA-366

Scenario(s):		[HYDRA-334] UUAR: There should be support for developing auto-configuration of certain devices
Relevant Quality Attributes:		Portability/Installability
Scenario	Stimulus Source:	Maintainer
	Stimulus:	Needs to install Hydra-based system
	Artefact (If Known):	System
	Environment:	Setup
	Response:	Setup succeeds
	Response Measure:	Supported by middleware in defining auto-configuration properties Not in conflict with security
Questions:		
Issues:		

Table 25: : HYDRA-334

Scenario(s):		[HYDRA-334] UUAR: There should be support for developing auto-configuration of certain devices [HYDRA-318] UAAR: Devices should be able to be added to the system at runtime
Relevant Quality Attributes:		Portability/Installability
Scenario	Stimulus Source:	User
	Stimulus:	Installs a device
	Artefact (If Known):	System
	Environment:	Operational
	Response:	Device services are added to Hydra-enabled functionality
	Response Measure:	No downtime
Questions:		

Issues:	
----------------	--

Table 26: HYDRA-334, HYDRA-318

Scenario(s):	[HYDRA-317] UUAR: Support runtime reconfiguration	
Relevant Quality Attributes:	Portability/Adaptability	
Scenario	Stimulus Source:	System
	Stimulus:	Self-monitoring reveals performance problem
	Artefact (If Known):	System
	Environment:	Operation
	Response:	System is adapted to improve performance
	Response Measure:	No downtime
Questions:		
Issues:		

Table 27: HYDRA-317

Scenario(s):	[HYDRA-315] UAAR: Devices should be able to contain user interface/management interface [HYDRA-193] Support for natural interaction [HYDRA-369] Devices must have semantic description of its user interface	
Relevant Quality Attributes:	Quality in Use/Effectiveness	
Scenario	Stimulus Source:	
	Stimulus:	
	Artefact (If Known):	
	Environment:	
	Response:	
	Response Measure:	
Questions:		
Issues:	User interface issues are out-of-scope currently in Hydra	

Table 28: HYDRA-315, HYDRA-193, HYDRA-369

Scenario(s):	[HYDRA-312] UAAR: Support profiling of devices' performance
Relevant Quality Attributes:	Usability/Understandability
Stimulus Source:	Maintainer

	Stimulus:	Wants to know source of performance problem
	Artefact (If Known):	System
	Environment:	Runtime (degraded mode)
	Response:	Gets and understands profile of devices' performance
	Response Measure:	Maintainer can resolve 80% of performance problems
Questions:		
Issues:		

Table 29: HYDRA-312

Scenario(s):		[HYDRA-380] Ability to send a broadcast message to wake up sleeping devices
Relevant Quality Attributes:		Efficiency/Resource behaviour
Scenario	Stimulus Source:	Subsystem
	Stimulus:	Needs to save energy
	Artefact (If Known):	Subsystem
	Environment:	Operation
	Response:	Subsystem (of devices) is made to sleep
	Response Measure:	Wakeup possible via broadcast Does not violate other quality requirements
Questions:		
Issues:		

Table 30: HYDRA-380

Scenario(s):		[HYDRA-326] UAAR: Support (residential) gateway devices at least at physical level [HYDRA-284] Legacy components [HYDRA-261] Bridges between different technologies
Relevant Quality Attributes:		Functionality/Interoperability
Scenario	Stimulus Source:	Developer
	Stimulus:	Wants to interface to Bluetooth device
	Artefact (If Known):	Subsystem
	Environment:	Development time
	Response:	Integration is done

	Response Measure:	Assumptions of IP communication are not violated New link layer implemented as component (in gateway/proxy)
Questions:		
Issues:		

Table 31: HYDRA-326, HYDRA-284, HYDRA-261

Scenario(s):		[HYDRA-359] Device ontology versioning
Relevant Quality Attributes:		Portability/Co-existence
Scenario	Stimulus Source:	Maintainer
	Stimulus:	Installs new device ontology
	Artefact (If Known):	Device
	Environment:	Operation
	Response:	Ontology is installed
	Response Measure:	Conflicts avoided/detected
Questions:		
Issues:		

Table 32: HYDRA-359

Scenario(s):		[HYDRA-322] UUAR: Support lightweight service composition [HYDRA-113] Composition (of services and devices)
Relevant Quality Attributes:		Efficiency/Resource utilisation Functionality/Suitability
Scenario	Stimulus Source:	Developer
	Stimulus:	Wants to compose service from different, embedded devices
	Artefact (If Known):	Subsystem
	Environment:	Development time
	Response:	Services are composed
	Response Measure:	Able to compose service on lowest range of embedded devices supported by Hydra
Questions:		
Issues:		

Table 33: HYDRA-322, HYDRA-113

Scenario(s):		[HYDRA-143] Model-based reasoning about itself [HYDRA-239] Automatic service diagnostic for security relevant services
---------------------	--	---

Relevant Quality Attributes:		Maintainability/Analysability Functionality/Security
Scenario	Stimulus Source:	System
	Stimulus:	Subsystem malfunctions
	Artefact (If Known):	Subsystem
	Environment:	Online
	Response:	Malfunction identified
	Response Measure:	Middleware able to detect status in 90% of cases
Questions:		
Issues:		"90%" needs qualification to be testable

Table 34: HYDRA-143, HYDRA-239

Scenario(s):		[HYDRA-129] Support for Semantic Web Standards for Device Communication [HYDRA-114] Semantic enabling of device web services
Relevant Quality Attributes:		Functionality/Suitability
Scenario	Stimulus Source:	Developer
	Stimulus:	Want to use device web service
	Artefact (If Known):	Device web service
	Environment:	Development
	Response:	Service incorporated into application
	Response Measure:	Device web service made suitable (even if not from the outset) through semantic descriptions
Questions:		
Issues:		

Table 35: HYDRA-129, HYDRA-114

Scenario(s):		[HYDRA-297] Secure data erasing on HYDRA enabled devices
Relevant Quality Attributes:		Functionality/Security
Scenario	Stimulus Source:	User
	Stimulus:	Wants to erase data
	Artefact (If Known):	Device
	Environment:	Online/offline
	Response:	Data is erased

	Response Measure:	Erasure is secure
Questions:		
Issues:		Data erasure is hard

Table 36: HYDRA-297

Scenario(s):		[HYDRA-53] The availability of devices should be guaranteed
Relevant Quality Attributes:		Reliability/Maturity
Scenario	Stimulus Source:	Internal
	Stimulus:	Failure
	Artefact (If Known):	Resource
	Environment:	Operation
	Response:	Similar resource found and used
	Response Measure:	Timely and with high probability
Questions:		
Issues:		This is a very general quality requirement

Table 37: HYDRA-53

7.3 Utility tree

The utility tree for Hydra WP4 is shown in Figure 8 . Subsequent figures show the utility sub-tree for each relevant ISO 9126 (sub-) characteristic in a more readable way.

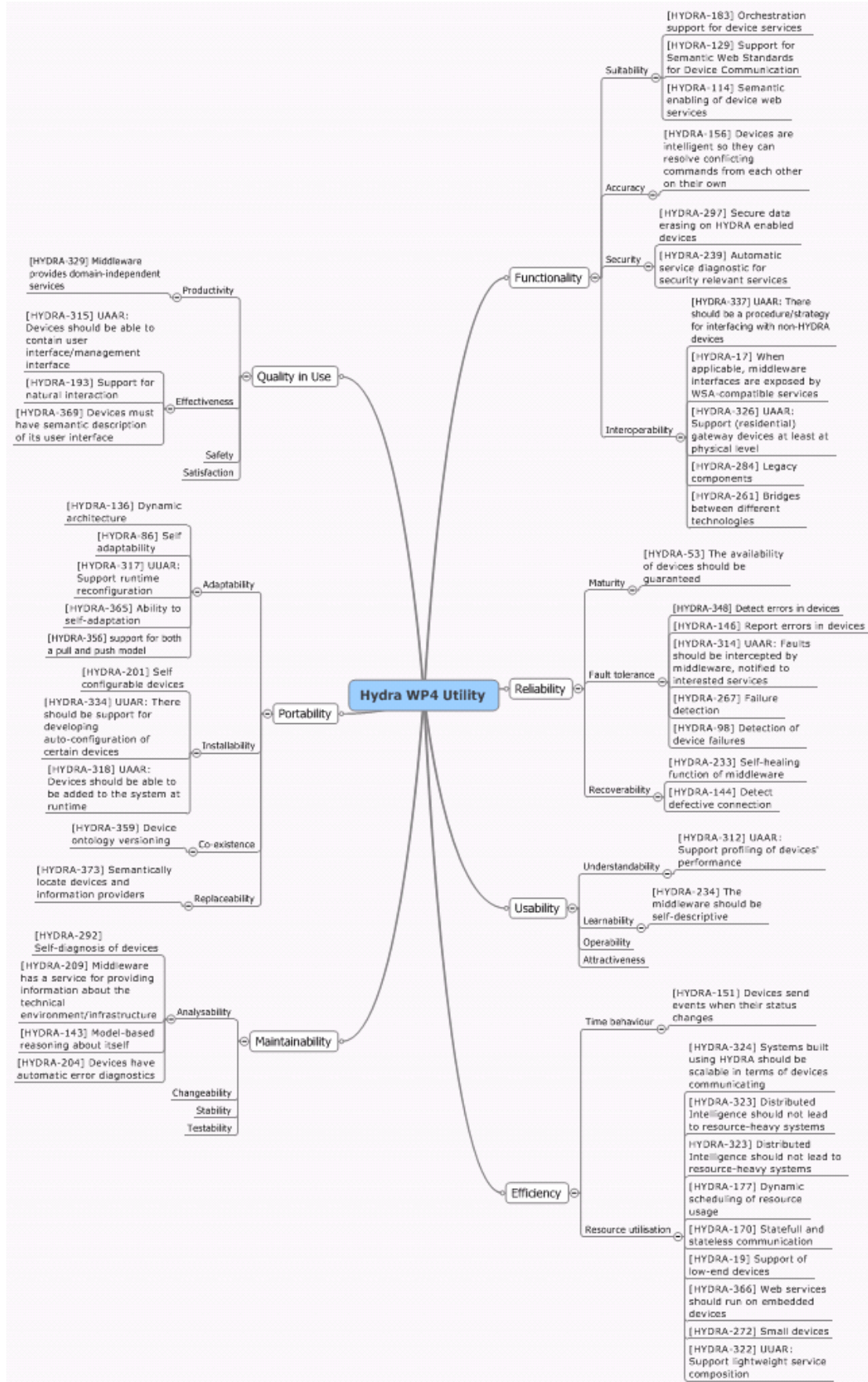


Figure 8: Hydra WP4 Utility

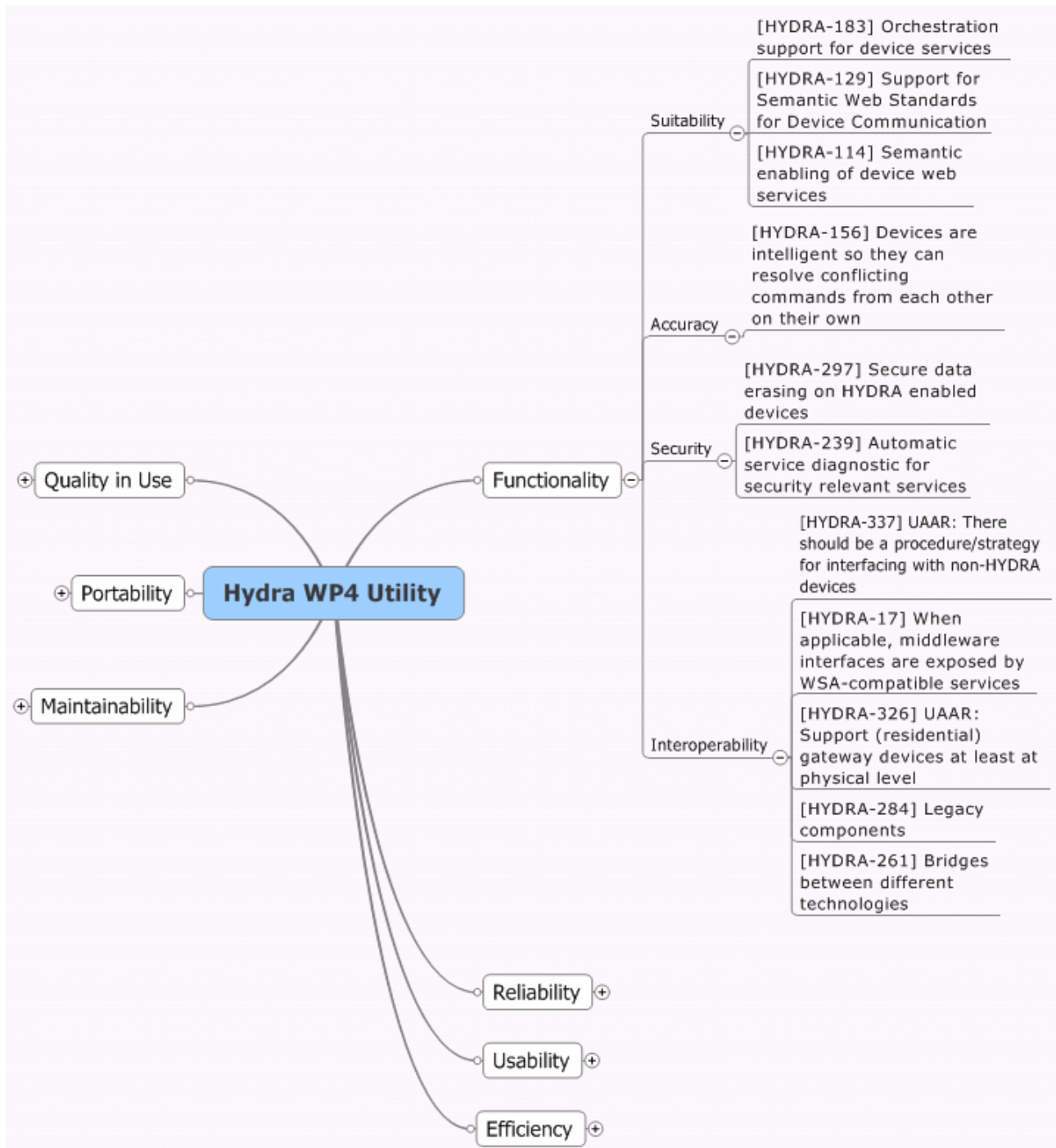


Figure 9: Hydra WP4 Utility - Functionality

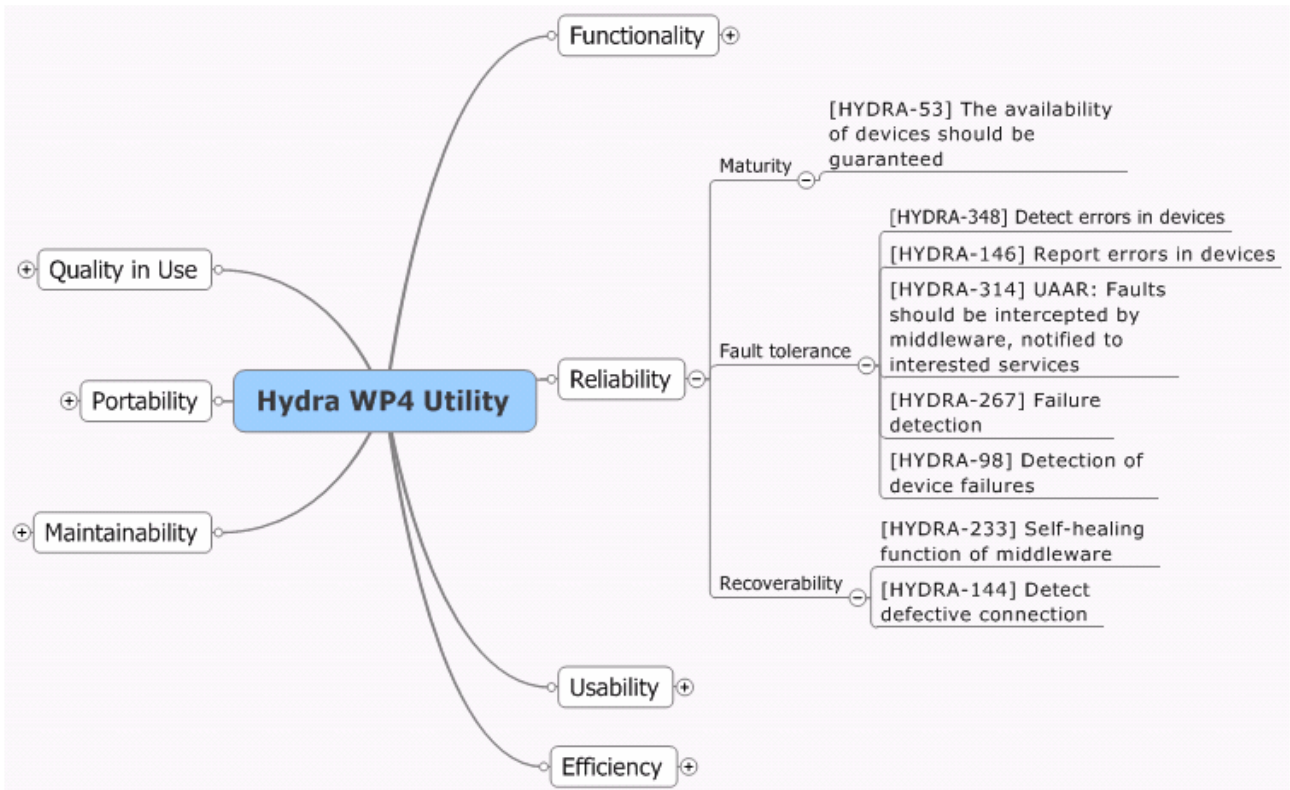


Figure 10: Hydra WP4 Utility – Reliability

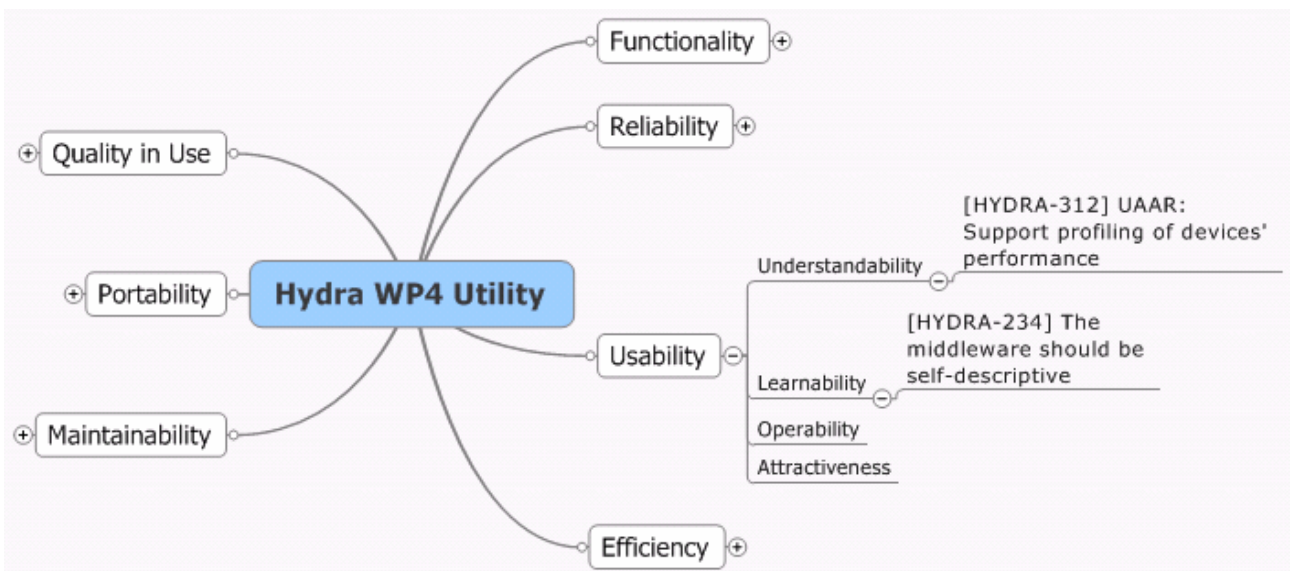


Figure 11: Hydra WP4 Utility – Usability

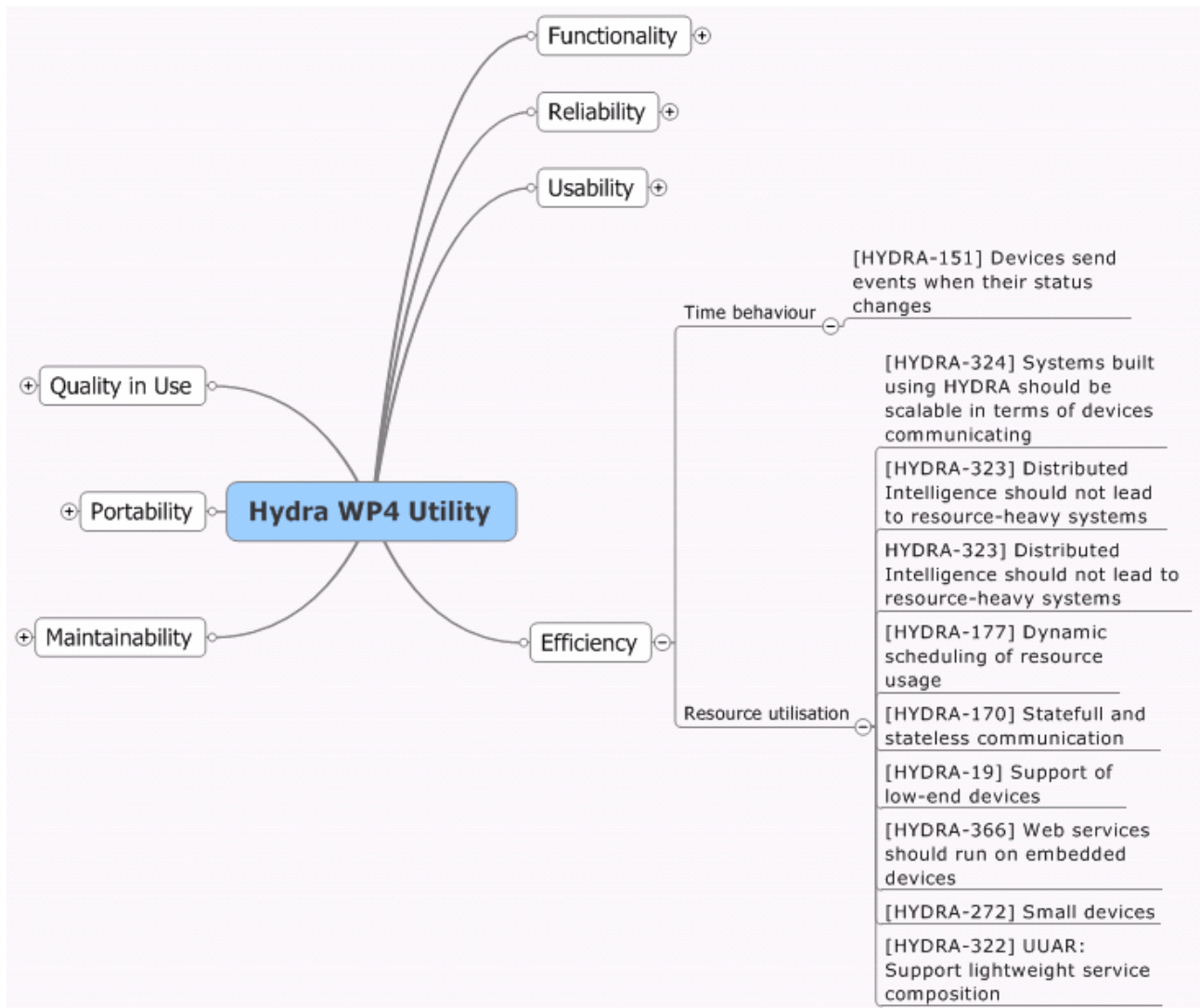


Figure 12: Hydra WP4 Utility – Efficiency

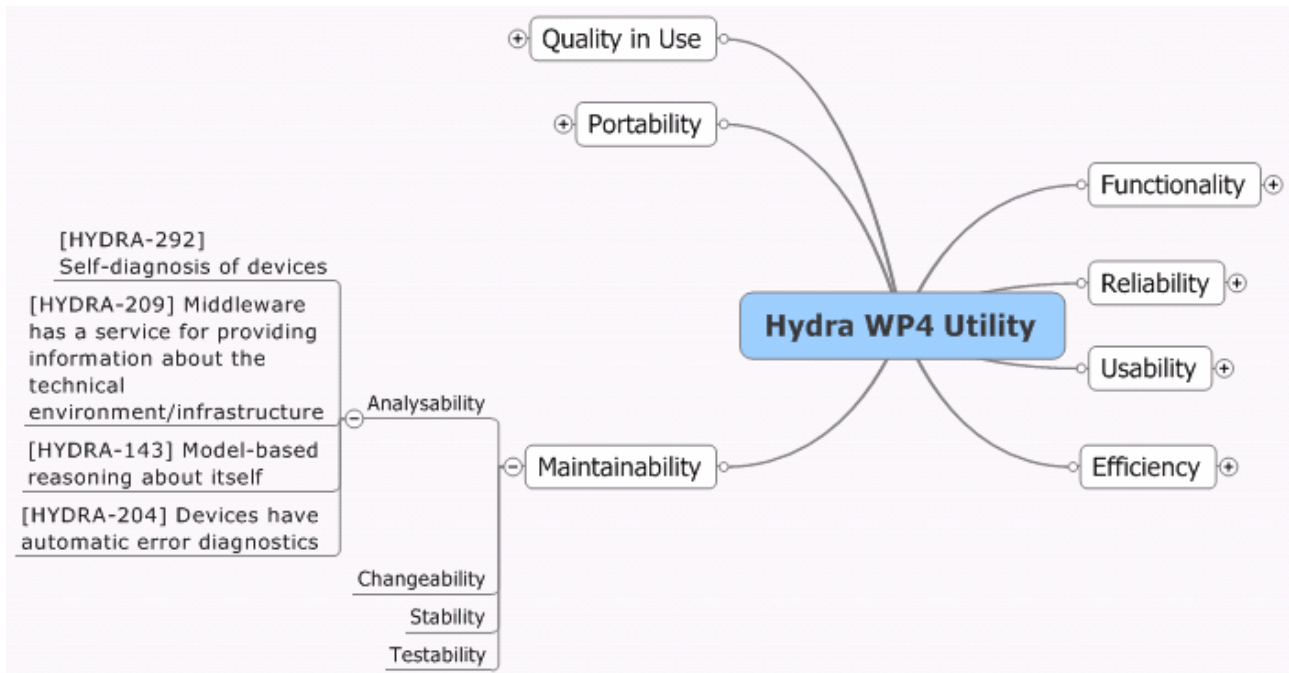


Figure 13: Hydra WP4 Utility – Maintainability

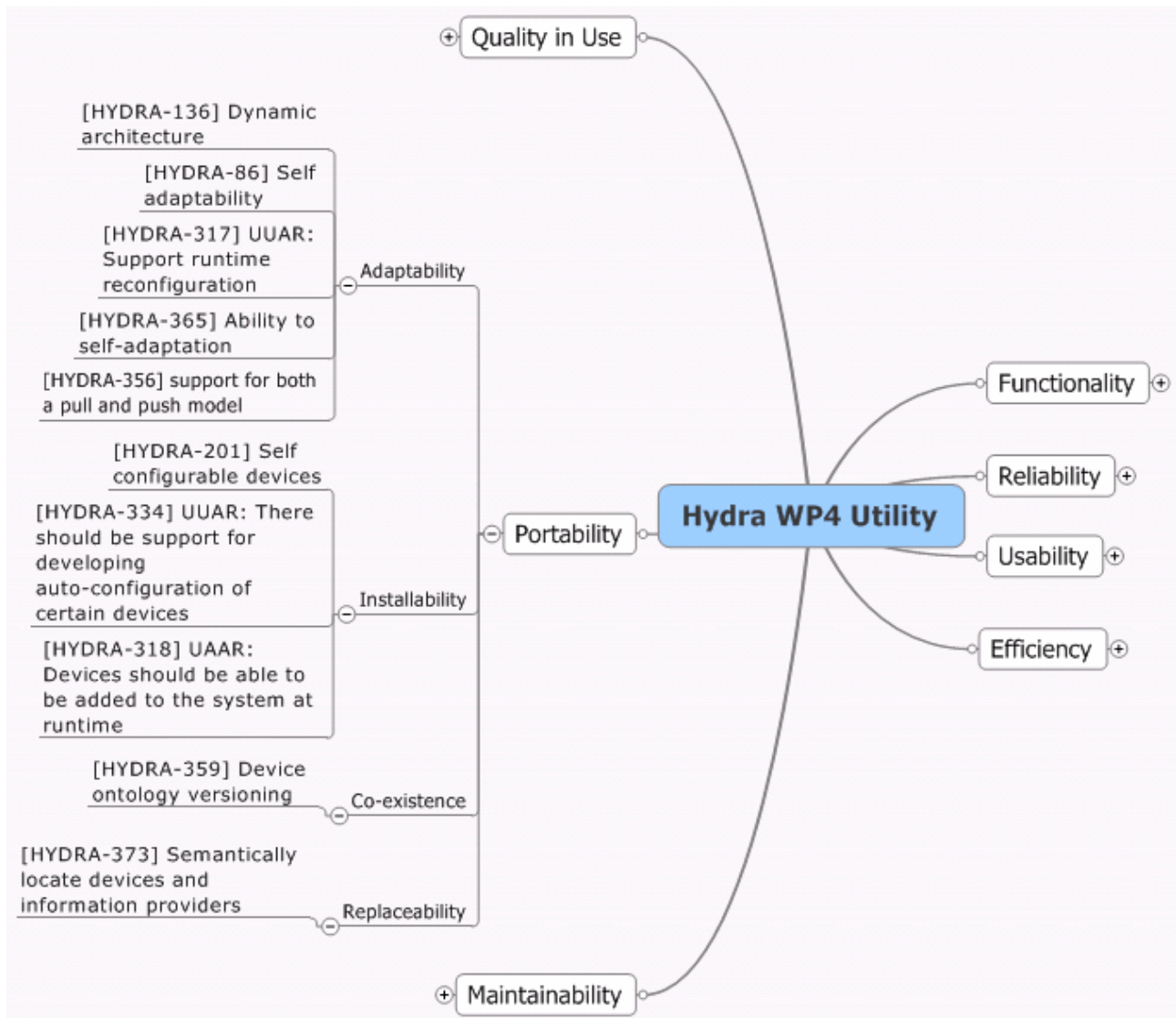


Figure 14: Hydra WP4 Utility – Portability

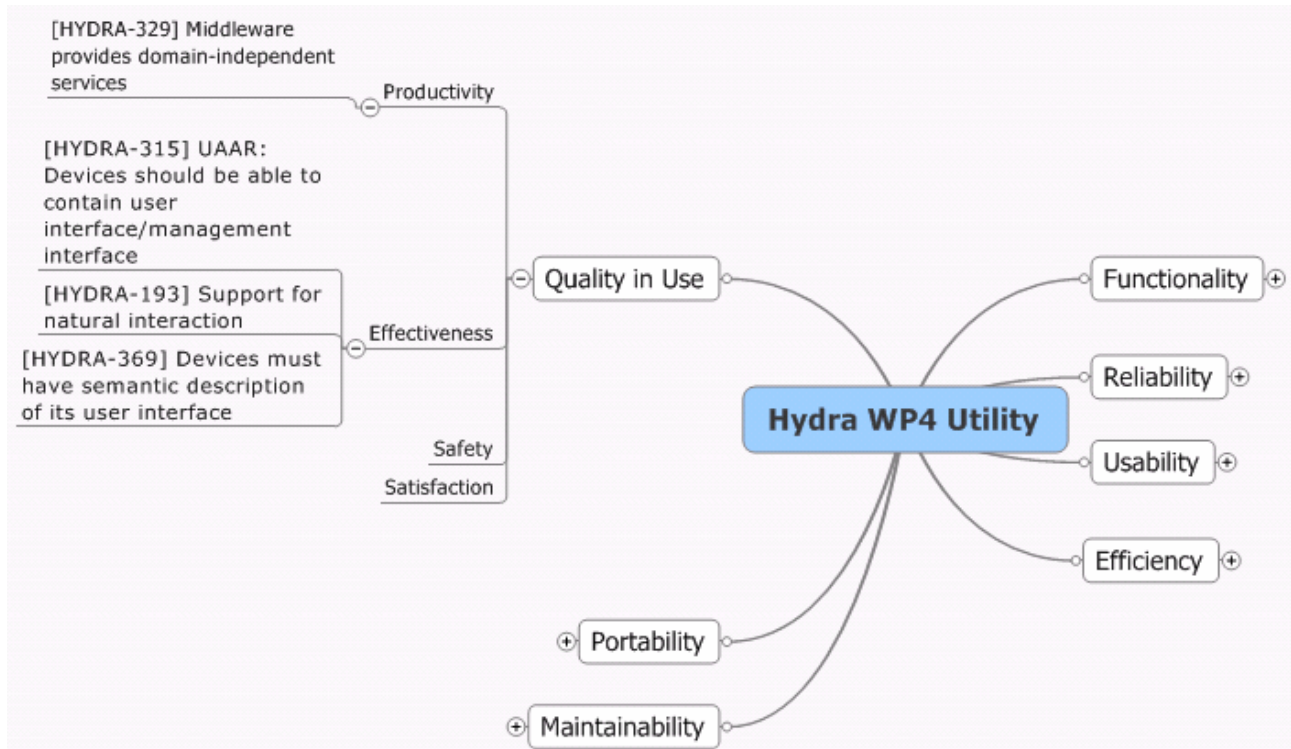


Figure 15: Hydra WP4 Utility – Quality in Use

7.4 Discussion

In this section, we discuss implications of the quality requirements. First of all, one can observe that few quality attribute requirements exist for Maintainability (in particular developer users have had few requirements to testability which could be assumed to be central to complex middleware as Hydra). The quality requirements that do exist pertain to self-* properties, decisions on which will be outlined in the next section. Furthermore, the same is the case for requirements on Usability (in particular Understandability and Learnability).

With respect to Quality in Use, the requirements pertaining to user interaction have been deemed to be out of scope for the moment. Still there is a productivity requirement to provide domain-independent service (HYDRA-329). The initial decision is to provide the Managers as defined in D3.3 as basic services.

Considering Efficiency, the main issue is reconciling the requirements for the support of low-end devices with the requirements to base part of the middleware on web services. Here it is necessary to experiment on placing web services on powerful nodes ("proxies") and interfacing to devices in device-specific ways as well as placing web services directly on devices. The latter will potentially improve Interoperability and Suitability for Hydra-based systems, but current web service technology does not support web services on the devices with the resources that the requirements envision.

Moreover, there is in general (as the questions and issues above suggest) a need for more accurate definitions of response measures; something that will need to be revisited throughout the project. For the design decision of using web services (see next section) this will be an outcome of the first demonstrator. Our approach to resolving possible conflicts are discussed in the next section.

8. Conclusion

This deliverable has described features and constraints of devices and technologies which can be used in the design and implementation of embedded AmI solutions in Hydra.

From the initial state-of-the-art analysis we concluded that UPnP would provide a good starting point for device descriptions. Existing design toolkits (e.g., from Intel) can readily be applied by the project for the first demonstrator versions of Hydra device discovery and control protocols. Furthermore, there are example device descriptions and control protocols available from manufacturers and the various UPnP initiatives (www.upnp.org). These will provide a basis for the build-up of a Hydra device ontology. As described in this and in previous deliverables (D3.1) UPnP is an established technology, based on prevailing Internet protocols (e.g., IP, DHCP) and standards (such as XML), and supported by major IT vendors and device manufacturers. At the same time we observe that challenges in devices heterogeneity, security and scalability as put forward by Hydra, will require further development of the UPnP architecture.

After that we considered the end-user domains of Hydra and the specific devices that we expect to be faced with. We then analysed which devices are needed in our specific scenarios and what kind of features we expect them to provide.

In chapter 6 we analysed the needs of the demonstrator to build for month 12 and the devices we have selected to integrate into the demonstrator. For each device we exactly define the services it will provide and the interface it will offer. With all this in mind we then performed a quality attribute analysis mapping the relevant Hydra requirements to a set of quality attributes and quality attribute scenarios for the embedded AmI architecture in WP4.

In relation to the two defined prototyping tasks in WP4 of Hydra, the following categories of requirements are of particular interest:

1. T4.2 Embedded Services: Efficiency
2. T4.3 Self-* Properties: Maintainability and Portability

This is the first iteration on requirements, but a set of informed choices have been made for both 1. and 2. For each choice, prototypes will be built and following a quantitative approach we will investigate whether the choice is reasonable. An example of this is to compare quantitatively the resource and time efficiency of proxy-based and device-based web services as outlined above.

In the following we will briefly describe these choices:

T4.2 Embedded Services. As already was described in chapter 5, all in all the requirements points to the need for investigating web services on embedded systems. To do this we are developing the *Limbo* compiler. The compiler will produce target code based on a Web Service Description Language (WSDL) description of the service combined with a semantic description of the device type, characteristics, resources etc. A reasonable outset for semantics is the device ontology of [Bandara 04]. Furthermore, the device ontology will be used in order to generate code that report resources on the device (using the Event Manager functionality as defined in deliverable D3.3).

Preliminary experiments have been made with *Limbo*, which clearly shows the need for a specialised tool for generating web services for devices.

T4.3 Self- properties.* The base level for self-* properties are self-monitoring which in the light of the decision above boils down to how to monitor web services. Due to the varying range of devices and resource availability, our initial decision is to provide a uniform mechanism for reporting system events (again through the Event Manager). Given this we want to pursue using an online Coloured Petri Nets [Jensen, 1997] simulator that will allow developers/users to describe reactions to system events.

9. References

- [Amigo 06]
IST Amigo Project, Amigo Middleware Core: Prototype Implementation and Documentation, Deliverable 3.2, IST-2004-004182, 2006
- [Bandara 04]
A. Bandara, T. Payne, D. Roure, and G. Clemo. An ontological framework for semantic description of devices. In *The Semantic Web - ISWC 2004*, 2004.
- [CCPP]
Composite Capabilities/Preference Profiles Information Page, W3C, <http://www.w3.org/Mobile/CCPP/>
- [FIPA 02]
FIPA Device Ontology Specification, <http://www.fipa.org/specs/fipa00091/SI00091E.html>, 2002
- [Hansen 07]
Hansen, K.M. D6.1 Quality Attributes Scenarios. Deliverable D6.1, Hydra, IST 2005-034891, 2007.
- [Jensen, 1997]
Jensen, K. *Coloured Petri Nets--Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts*. Springer-Verlage, 1997
- [MFS 99]
A Syntax for Describing Media Feature Sets, IETF RFC-2533, <http://www.ietf.org/rfc/rfc2533.txt>, 1999
- [MPEG 02]
MPEG-21 Overview (v.5), ISO/IEC JTC1/SC29/WG11/N5231, <http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm>, 2002
- [MQ 02]
Media Queries, W3C Recommendation, <http://www.w3.org/TR/css3-mediaqueries/>, 2002
- [Schmidt, 2002]
Schmidt, D.C. Middleware for real-time and embedded systems. *Communications of the ACM*, 45(6), pp 43-48.
- [SMIL 01]
Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, <http://www.w3.org/TR/2001/REC-smil20-20010807/>, 2001
- [SyncML]
The SyncML Initiative, <http://www.syncml.org/>
- [TCN 98]
Transparent Content Negotiation in HTTP, IETF RFC-2295, <http://www.ietf.org/rfc/rfc2295.txt>, 1998
- [UAPProf 07]
OMA User Agent Profile V2.0 Approved Enabler, http://www.openmobilealliance.org/release_program/uap_v2_0.html, 2007
- [UPnP]
Universal Plug and Play, <http://www.upnp.org/>
- [WUFRL]
Wireless Universal Resource File, <http://wurfl.sourceforge.net/>

[WVI]

Wireless Village Initiative, <http://www.wireless-village.org/>

10. Appendix Device Examples

10.1 Building Automation

<p>MAGNA UPE - CIRCULATOR PUMPS</p> <p>These devices are circulator pumps designed for circulating liquids in heating systems with variable flows where it is desirable to optimise the setting of the pump duty point. The pumps are also suitable for domestic hot-water systems.</p>
<p>Features</p> <p>In general this type of pump supports communication to/with an external system requiring for control or monitoring of the pump; this communication provides means to access to:</p> <ul style="list-style-type: none"> - speed control of pump or change of set point; - reading of pump data; - start/stop, fault indication or forced control to max. or min. curve. <p>Note: The communication options depend on the pump type.</p>
<p>COMMUNICATION</p> <p>MAGNA pumps can be fitted with an expansion module enabling communication via external signals (signal transmitters).</p> <p>Different types of expansion modules are available in respect to different pumps:</p> <ul style="list-style-type: none"> - GENI module; - Relay module; - LON module. <p>The GENI module enables serial communication via an RS-485 input. The communication is carried out according to the Grundfos bus protocol, GENIbus, and enables connection to the GRUNDFOS Pump Management System 2000, a building management system or another type of external control system.</p> <p>The relay module offers functions as : external start/stop, fault, ready and operating indication via signal relay, fault indication, ready indication.</p> <p>The LON module offers the possibility of connecting the pump to a LonWorks® network. The module is used for data transmission between a network and the MAGNA pump.</p> <p>Furthermore pumps have a function incorporated which is the R100, designed for wireless communication. The R100 communicates with the pump via infra-red light.</p> <p>The R100 can be used for the following functions:</p> <ul style="list-style-type: none"> - reading of operating data; - reading of fault indications; - setting of control mode; - setting of 0.1 m head increments; - selection of external set point signal; - allocation of pump number making it possible to distinguish between pumps in connection with parallel operation via bus; - selection of function for digital input.

Table A 1



<p>Excel 500® - Primary Controller</p> <p>The Excel 500 system is a “primary” controller, freely programmable that can be used as a stand-alone controller or as a part of a network. The Excel 500 System provides energy management and control function via LonWorks bus to Honeywell Lonworks I/O modules, 3rd-party LonWorks I/O modules, or 3rd-party LonWorks devices.</p>		
<p>Features</p> <p>In addition to control applications for heating, ventilation, and air conditioning (HVAC), the device performs a wide range of energy management functions, including optimum start/stop, night purge, and maximum load demand.</p> <p>In details here follows a list of special features supported:</p> <ul style="list-style-type: none"> - Programmable Functions (Alarm Handling, Protection Password) using Honeywell CARE (Computer Aided Regulation Engineering) - HVAC (heating, ventilation and air conditioning) - Energy Management functions(optimum start/stop, night purge and maximum load demand) - Up to 4 Building Supervisor can be connected via the system bus - Add-on Modules for additional function(Computer Communication, Analog/Digital I/O, Smart I/O) 		
<p>CPU</p> <p>Memory specifications: 64KB EPROM (boot) 256KB RAM 1 MB Flash EPROM (firmware and application) 4KB EEPROM</p>		
<p>COMUNICATION</p> <ol style="list-style-type: none"> 1. C-BUS (Honeywell proprietary technology): The C-BUS transmits data between the system controllers, gateways to 3rd-party system, and building supervisor at 9.6 Kbaud up to 76.8 Kbaud. The max. C-bus length is 1.200m or 4.800m using a repeater. A switch is provided for selectable termination. There is a maximum of 30 controllers or devices per C-bus. 2. LonWorks Bus: The Excel 500 uses an FTT-10A Free Topology Transceiver, transmitting data at 78 Kbaud using LonTalk® protocol. Cable length from 320 to 2200m. 3. Eternal Interface / Modem : A 9-pin Sub-D connector, RS232, is provided as a serial port to connect an external interface (e.g. a PC-based operator and service software) or a GSM modem or ISDN terminal adapter for dial-up access at a transmission rate of up to 38.4 Kbaud. 		

Table A 2

<p>Excel 10® WT7750A - Zone Controller</p>		
<p>Description</p> <p>This device is the Constant Volume Air Handling Unit (CVAHU) Controller. The CVHAU is a LonMark compliant device designed to control single zone and heat air handlers: Provides control of small-packaged, single-zone rooftop air handling units and heat pumps with staged heat/cool.</p>		
<p>Features</p>		

<ul style="list-style-type: none"> - Heat pump control: Up to four compressor and up to four stages of auxiliary heat; - Up to four stages of electric or gas fired heating; - Up to four stages of mechanical cooling; - Floating hot water or steam heat; - Pulse width modulated heat; - Floating chilled water cooling; - Pulse with modulating cooling; - Floating mixed air damper output; - Pulse width modulating mixed air dumper output; - Occupancy sensor override and window open override.
CPU
<p>Motorola or Toshiba 3150 Neuron® processor, containing three 8-bit CPU.s. Each Neuron has a unique 48-bit network identification number.</p> <p>Memory The Controller uses a 64KB ROM/PROM (16KB reserved for network operations, 48KB usable for control algorithm code). 512 bytes EEPROM 2KB RAM</p>
Communication
<p>LONWORKS Bus communications is supported. The Excel 10 controller uses a Free Topology Transceiver (FTT) transformer-coupled communications port running at 78 kilobits per second (Kbps). The maximum LONWORKS Bus network length is 1524m. With the addition of a Router, the maximum length of the LONWORKS Bus network can increase to 3048m. The maximum number of nodes per LONWORKS Bus segment is 60. The maximum number of nodes per Q7750A Zone Manager FTT network is 120.</p>

Table A 3

T7770A-G Temperature Sensor Wall Module
Description
<p>The T7770A through G are a family of direct wired wall modules for use with the controllers we have mentioned in previous tables. All models have a space temperature sensor; some models have set point adjustment, override with LED, and fan switch.</p>
Features
<p>Sensor accuracy : 5° to 37° Dimensions (H/W/D): 5-1/16 x 3-1/8 x 1 in. (128 x 80 x 25 mm).</p>
Communication
<p>LONMARK bus communications port</p>

Table A 4

PRO 100 – Home automation controller
Description
<p>The device is an advanced home gateway that’s designed to support integration and automation of the major home systems. It can be used to monitor and control lighting, appliances, HVAC, climate, security, telephones, irrigation, window shades and home entertainment equipment.</p>
Features
<ul style="list-style-type: none"> - The controller has a built-in web server for remote access anywhere, the entire system can be monitored and controlled from the web - Supports the widest variety of user interfaces, including in-wall scene controllers, touch screens, wireless remotes, PDAs, Pocket-PCs, cell phones and voice (microphones). - It is compatible with a wide variety of technologies including Lutron®, Leviton®, Z-Wave®, UPB™, Insteon®, X10 and many others.



<ul style="list-style-type: none"> - It is designed for voice control by microphone and telephone; any computer in the house can be used to control the system with the voice. - Creates alerts, announcements and reminders by email, telephone and text message. - Supports complete telephone messaging with caller ID announcements (with optional hardware). - Includes calendar-based functions for easy scheduling. - Automation events may be created 'on-the-fly' with voice commands. - It is provided with protection from virus, spyware and malware attacks. - Includes true 'trigger-based' automation with advanced conditional options. - Built-in power failure recovery feature runs events missed during outages when power is restored. - Includes multiple serial, USB and Ethernet ports for integration with the most commonly used technology interfaces. - An API (Application Programming Interface Specification) is available that allows for the creation of custom software "plug-ins". These plug-ins allow the software to be extended to support custom hardware.
CPU
The processor is a 1.5 GHz Celeron® M with no fans. Operating System: Embedded Windows XP
Memory
Flash Storage: 2 Gb (DOM) DDR Memory: 1 Gb
Communication
Serial Ports:(4) 9-pin RS232 USB Ports:(2) USB 2.0 Network:(2) 10/100 Ethernet PS2 Ports: Keyboard & Mouse Audio: Mic in / Line out Supported standards UPB (Universal Powerline Bus) X-10 Z-Waves Restricted to a list of supported Hardware Company (maybe need a third part plug-in to use)

Table A 5


Z-Wave Motion Sensor (HomePro)
Description
<p>The motion sensor runs on 4 AAA batteries (included) and is wireless. The unit can control up to 5 Z-Wave devices directly, or it can be programmed to send commands directly if more complex events can be triggered. The unit includes a swivel mounting bracket for easy aiming. The lens is adjustable in two positions. Position 1 is wide angle and has a max range of 39 feet. Position two is long range, about 46 feet. The unit will only send a motion command once, when it detects motion. It will then send one "No Motion" command after a programmable period of time that it does not detect any motion (default is 2 minutes).</p> 
Features
<ul style="list-style-type: none"> - Requires 4 AAA batteries (included) - Programmable "no motion" timeout between 2 and 255 minutes - Supports Z-Wave associations - Can directly control up to 5 Z-Wave devices with ON and OFF commands (e.g. a light switch) - Battery life 1 year approximately
Communication
Z-Wave Open Standard Communication

Table A 6


Mitac Mio A701	
Description	
<p>This pocket-PC phone is a medium-class device in the market but offers a number of significant features:</p> <ul style="list-style-type: none"> - an integrated GPS receiver; - good computational power; - good battery life. 	
Features	
Connectivity	
Bluetooth 1.2 Class 2	
Slot SDIO/MMC	
USB client 1.1 (miniUSB port)	
GPS Reciver	
Chipset SiRFStarIII	
Built-in Antenna	
Supports SiRF InstantFix	
Battery: Rechargeable Lithium-ion battery 1320 mAh	
Battery Life: 9h (with no software application), 4:45 h (intensive software application)	
CPU	
Intel Xscale PXA270 520 MHz, Operating System : Windows Mobile 5 for Pocket PC Phone Edition	
Memory	
128 MB ROM	
64 MB RAM	
Communication	
GSM/GPRS: 900/1800/1900 MHz GPRS call B, Multi-slot 10	

Table A 7

10.2 eHealth

Avant 4000 - Wireless Wearable Oximetry	
Description	
This device is a cutting-edge oximetry using a wireless solution. This system with Bluetooth Wireless Technology provides a lightweight wrist-worn patient module (Avant 4100) which wirelessly sends data to a small tabletop display, improving patient mobility and reducing bedside clutter.	
Features	
Patient module <ul style="list-style-type: none"> - Nonin's PureLight® sensor technology - Oxygen saturation range : 0%÷100% - Pulse rate : 18 to 300 pulse per minute - Measurement Wave Lengths: <ul style="list-style-type: none"> Red : 660 nanometres @ 2mw nominal Infrared : 910 nanometres @ 3mw nominal - Weight: 125g with batteries Display module <ul style="list-style-type: none"> - Numeric display : 3-digit LED's, Tricolour (red, green, amber) - Dimension : 18.4cm wide x 14cm high x 11.4cm deep - Weight 1kg 	
Memory	
Avant 4100 (Patient module): none Avant 4000 (Display module): 33.5 hours of data sampled	
Power& Battery life	
Patient module Internal Power: Two 1.5 volt AA batteries Continuous Operation : Minimum 120 hours Display module Internal Power: 7.2 volt battery pack (6 cells) Continuous Operation : Minimum 18 Hours Recharge : 4 hours	
Communication	
The wrist worn device communicates with the Display through Bluetooth technology. Bluetooth Compliance: version 1.1 Operating frequency: 2.4 to 2.4835 GHz Communication: serial port Network Topology: Point to Point Output Power: < 1.1 mW Antenna type: internal	

Table A 8

Propaq CS – Vital sign monitor
Description
The Propaq CS monitor is a bedside vital signs monitoring solution. It is easy to use for all clinicians, engineered for patients of all ages. Its features include: full patient alarms and intuitive equipment alerts, trending, programmable defaults and long battery life.
Features

<p>Main functions: Two channels of ECG, Heart/pulse rate, Smartcuf®, non invasive blood pressure, temperature, motion tolerant pulse oximetry, impedance respiration, full patient alarms, equipment alerts, trending on all parameters.</p> <p>This device networks via fixed line or a wireless link to a Central Station becoming part of a Flexible Monitoring solution.</p> <p>The Central Station provides a map of your care unit, including up to 60 monitored patients, their status and location. The system is available with 96-hour</p> <p>It's possible to integrate a printer.</p>
<p>Communication</p> <p>Modem – The Modem Propaq CS provides simple telemedicine via standard telephone lines.</p> <p>Wireless Communication option</p> <p>FlexNet Network : 2.4 GHz Wireless Local Area Network (WLAN) and 10/100 Base-T Ethernet Network</p> <p>Modulation : GFSK, Frequency Hopping Spread Spectrum (FHSS) IEEE 802.11 compliant</p> <p>Every Access Point support max. 7 Wireless Propaq CS monitors</p>

Table A 9


A&D UA767 Series - Blood pressure monitor	
Description	
<p>The professional blood pressure monitor (UA-767 series) is available with wired or with Bluetooth wireless technology to give system integrators a seamless connection to an Access Point (e.g. personal computer, personal digital assistant (PDA), home hub, etc.).</p>	
Features	
<ul style="list-style-type: none"> - Professional accuracy via oscillometric method - Clinically validated - One button operation - Provides time and date stamp - Each monitor has unique serial number and unique BT id - Meets ANSI/AAMI SP10 standards - Send real-time blood pressure measurements to the Access Point. - These devices can also operate in a batch-mode to send a number of measurements with time and date in a single request command 	
Memory	
<p>UA-767PC: 126 blood pressure readings</p> <p>UA-767PBT: 40 blood pressure readings</p> <p>Power& Battery life</p> <p>About six months with one daily measurement using 4 type AA alkaline batteries.</p>	
Communication	
<p>UA-767PC : Wired Communication with PC via built-in RS-232C port</p> <p>UA-767PBT : Wireless Bluetooth communications</p>	

Table A 10

A&D UC-321 Series - Scales
Description


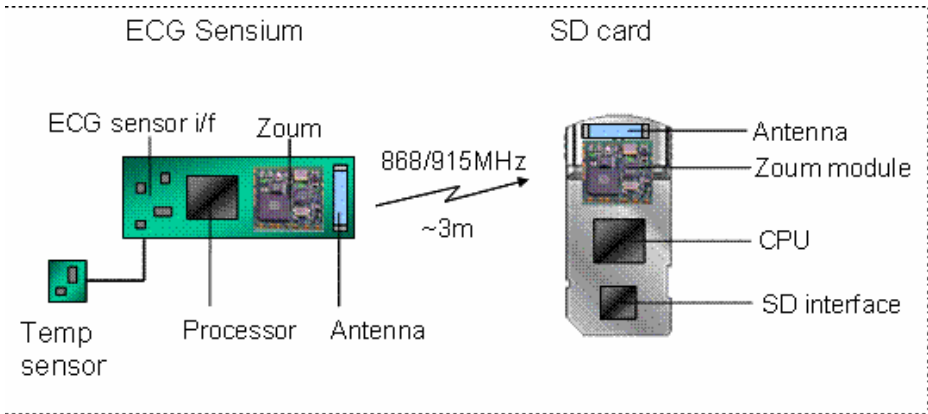
<p>The UC-321 series are one of the thinnest and lightest scales on the market, measuring less than 1" thick.</p> 
<p>Features</p> <p>UC-321BT - UC-321BT</p> <ul style="list-style-type: none"> - Maximum Capacity 200kg - Motion Tolerance <p>UC-321P</p> <ul style="list-style-type: none"> - Maximum Capacity 150kg
<p>Memory</p> <p>UC-321BT UC-321PBT : Memory storage for 40 readings UC-321P : Memory storage for 31 readings</p> <p>Power& Battery life</p> <p>Approximately 2,000 measurements using 4 type AA batteries</p>
<p>Communication</p> <p>UA-321PL / UA-321P : Wired Communication with PC via built-in RS-232C port UA-321BT / UA-321PBT : Wireless Bluetooth communications</p>

Table A 11

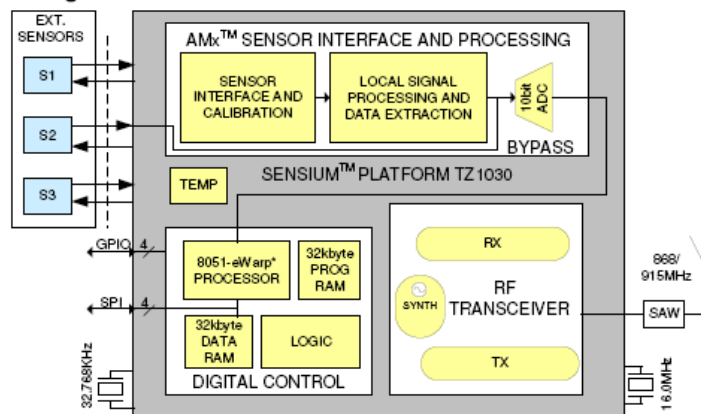
<p>ECG Sensium platform</p>
<p>Description</p> <p>The Sensium is an ultra low power sensor interface and transceiver platform for a wide range of applications in healthcare and lifestyle management. The device includes a reconfigurable sensor interface, digital block with 8051 processor and an RF transceiver block. On chip program and data memory permits local processing of signals. This capability can significantly reduce the transmit data payload.</p>

<p>Features</p> <ul style="list-style-type: none"> - Together with an appropriate external sensor, provides ultra low power monitoring of: <ul style="list-style-type: none"> • ECG;

- Temperature;
 - blood glucose
 - oxygen levels.
- It can also interface to 3 axis accelerometers, pressure sensors and includes a temperature sensor on chip.
 - The ECG sensor has a data rate of 50Kbps
 - The temperature sensor has 10bit resolution

CPU

The platform TZ1030 includes a 8051 processor and an RF transceiver block. On chip program and data memory permits local processing of signals. Very low level analog signals from the sensors are pre-processed by micropower circuitry (amplification, filtering, data conversion, data compression, modulation etc.) before being transmitted as an RF signal to the base station.

Block Diagram of TZ1030



Communication

The wearable sensor module incorporates an ultra low power RF transceiver (868/915 MHz RF)

The SD card plugs into any standard PDA or cell phone running Windows Mobile 5, allowing the ECG, temperature and other vital sign data to be received, processed and displayed via an application software.

Wearable sensor nodes support a range of sensor generating data at rates up to 50kbps.

The base station can be linked to up to 8 target stations, each monitoring multiple physiological signals on the body.

Battery life is typically 5 days from a 1.4V 675 Zinc air battery.

Table A 12

10.3 Agriculture

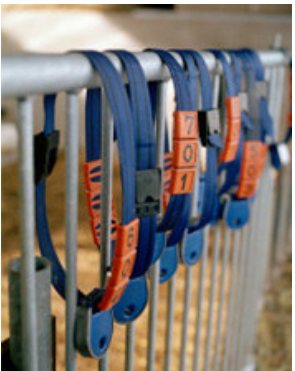

DeLaval ALPRO - activity meter	
Description	This activity meter system registers the animal's movement pattern via its neck mounted tag. A neck mounted tag is designed to give best cow comfort compared to a leg mounted tag.
Features	<p>The system comprises three main components:</p> <ul style="list-style-type: none"> - Neck mounted activity tags which register the cows movement pattern and forward the information on a hourly basis. - Antenna installation which receives the tags information for further processing. - An ALPRO processor delivering breeding attentions lists.
	
Communication	The activity tags send data via RF link (433.92 MHz in Europe) to a receiving antenna. A receiver converts data from the activity meter to a proprietary protocol.

Table A 13

Geonics EM38 – Conductivity meter	
Description	<p>This instrument was designed to measure the electrical conductivity of the soil to about 1.2m depth using the quadrature response of the ground to an electromagnetic wave.</p> <p>For larger-area surveys, the EM38 can be easily mounted on a platform and towed behind a vehicle. Real-time (RT) data acquisition, with direct connection to computer-based acquisition systems, is available with an optional modification.</p>
	
Features	<p>Support for the collection and processing of data from any Ground Conductivity Meter is available with any Data Acquisition System.</p> <p>Continuous or station measurements can also be taken from a standing position using the optional carrying handle with trigger and cable for connection to the DL720 data logger.</p> <p>The data logger in turn stores data, which is later transferred to a PC for contouring or profiling.</p>

Memory
<p>The data logger system (DL-720) includes an Omnidata Polycorder 700 series data logger, interconnecting cables to interface with the device and DAT software (IBM compatible) for data storage and reduction.</p> <p>Power& Battery life</p> <p>Battery Life: 30 hours continuous with a 9V battery.</p>

Table A 14