

The Hydra Project

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

Final report 2006 – 2010



Published by the Hydra consortium August 2011



SIXTH FRAMEWORK
PROGRAMME



European Commission
Information Society and Media

Copyright © 2006, 2011 - The Hydra Consortium

The Hydra Project has no independent legal status but is the result of work performed by a consortium of independent partners, temporarily created with the aim to perform research work under contract with the European Commission.

Permission is granted to use material published in this leaflet for personal use only. Its use for any other purpose, and in particular its commercial use or distribution, is strictly forbidden in the absence of prior written approval. Notwithstanding this requirement, material may be downloaded or printed for use in connection with scientific work and reports and press reports on the activities of the Hydra Project and its partners.

The Hydra Project has made every reasonable effort to ensure that the content of this leaflet is accurate and complete. We do not give any warranty in respect of the timeliness, accuracy or completeness of material, and disclaim all liability for (material or non-material) loss or damage incurred by third parties arising from the use of content obtained from this leaflet.

Registered trademarks and proprietary names, and copyrighted text and images, are not generally indicated as such. But the absence of such indications in no way implies that these names, images or text belong to the public domain in the context of trademark or copyright law.

Due to trademark conflicts, the name 'Hydra' only refers to the research project called Hydra. The outcomes of the project are marketed as LinkSmart™ middleware.

The Hydra project was co-funded by the European Commission within the Sixth Framework Programme in the area of Networked Embedded Systems under contract IST-2005-034891.

For more information go to www.hydramiddleware.eu or www.linksmart.eu. Or contact our webmaster at webmaster@hydramiddleweare.eu.



From the Hydra Coordinator

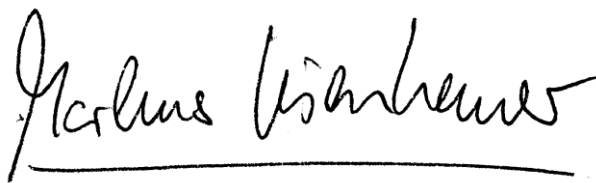
I would like to welcome you in the name of the Hydra Consortium to this document. It will provide you an executive overview of the very successful Hydra project and its achievements and introduce you to the LinkSmart middleware, its components, dedicated tools and the underlying concepts.

Hydra's project vision to create a middleware for networked mobile and embedded systems allowing producers to develop cost-effective and innovative applications for new and already existing devices is still valid and extends from the area of Networked Embedded Systems, to Internet of Things and cloud enabling. The very successful project has a large impact in terms of introducing new technology concepts (SOA) to networked embedded systems and the many approaches (self-management, security, middleware) delivered by the project have been copied and will be or are reused in many other projects.

In fact Hydra has delivered a comprehensive research roadmap for the Internet of Things and Services and is directly impacting the present EU RTD work programs. The impact can be seen in the number of new projects for research work that have been accepted for funding by the EU, included four very large Integrated Projects (i.e. ME3Gas (Artemis), REACTION, ebbits and Bridge (Security)). All projects are based on the LinkSmart middleware and the except for ME3Gas were ranked number one at the evaluation. Even more projects related to Hydra have been approved for funding and will follow (three more ranked number one at the evaluation).

The very impressive results of the Hydra project will be maintained and further developed by a non-profit foundation that will be led by involved Hydra consortium partners and that will be open also for external partners. You are welcome to join the foundation in order to push the open source development of the LinkSmart middleware and to establish a supporter community.

I wish you an insightful reading and look forward to interesting and fruitful feedback.



Dr. Markus Eisenhauer
Fraunhofer Institute for Applied Information Technology
Schloss Birlinghoven, Sankt Augustin (near Bonn), Germany



From the European Commission

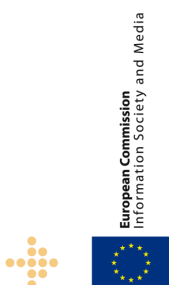
As part of the 6th Framework Programme for "Integrating and strengthening the European research area", the European Commission launched the 2005-06 Work Programme in November 2004 [1]. The changing environment for Information and Communication Technologies (ICT) research called for a new approach. As ICT was becoming more pervasive *"we see their growing impact all around us, in the way we live, work, play, and interact with each other. New ways of using ICT are at the origin of innovations in most products, services and processes"*.

The focus of IST in FP6 was on the future generation of technologies in which computers and networks were to be integrated into the everyday environment, rendering accessible a multitude of services and applications through easy-to-use human-machine interfaces. This vision of "ambient intelligence" places the user, the individual, at the centre of future developments for an inclusive knowledge-based society for all.

One of the research focus areas in the Work Programme was the "Embedded Systems" strategic objective with the aim to develop the next generation of technologies, methods and tools for modelling, design, implementation and operation of hardware/software systems embedded in intelligent devices. An end-to-end systems vision should allow us to build cost-efficient ambient intelligence systems with optimal performance, high confidence, reduced time to market and faster deployment.

A specific objective was to research and develop middleware and platforms for building secure, fault-tolerant Networked Embedded Systems where diverse heterogeneous physical objects cooperate to achieve a given goal. While the developed technology must be generic, it should be driven by an entire class of ambitious future applications, covering not only information handling but also monitoring/sensing and control. The middleware should thus hide the complexity of the underlying infrastructure while providing open interfaces to third parties for application development.

The Hydra project was highly relevant in this regard. The vision of the Hydra project to create the most widely deployed ICT middleware for intelligent networked embedded systems was fully in line with the overall objectives of developing tools for design, implementation and operation of software systems embedded in intelligent systems. The Hydra middleware – LinkSmart - also provides significant reduction in time to market and faster deployment of new types of devices and services. It will further allow SME's, with little development capacity, to take advantage of the growing market for intelligent devices and services.



Dr. Jorge Peirera
Hydra Project Officer
DG INFSO G3, Embedded Systems and Control
European Commission, Bruxelles, Belgium



Table of contents

From the European Commission Jorge Pereira, Hydra Project Officer	4
From the Hydra Coordinator	3
Executive Summary	6
A PROJECT METHODOLOGY BASED ON A USER-CENTERED DESIGN APPROACH.....	7
WP2: Iterative user requirements engineering.....	7
WP3: Specification of the Architecture	8
TECHNICAL OVERVIEW	14
WP6: SoA and MDA middleware	14
WP5: Wireless networks & devices	18
WP4: Embedded AmI architecture	22
WP7: Trust, privacy and security	23
PROJECT IMPACT.....	27
WP8: System Integration: Device and Application Development Tools	27
WP12: Training	29
HORIZONTAL ACTIVITIES	31
WP9: User Applications.....	31
WP 10: Validation & business framework.....	32
WP13: Dissemination and Exploitation.....	33
WP1: Project Management	34
References	34
Hydra project partners	35

Executive Summary

This document is a summary of the final report of the *Hydra research project*. It also constitutes a technical description of the *Open Source LinkSmart® middleware* for networked embedded systems, which is the official outcome of the Hydra project.

Due to trademark rights, the name "Hydra" can not be used for the middleware when marketed after the end of the project. So the partners registered a commercial name *LinkSmart*, which will be used throughout this document to refer to post-project artefacts, whereas *Hydra* only refers to the project-related events and artefacts.

Background

The Hydra project researched, developed, and validated a middleware platform for networked embedded systems that allows developers to develop cost-effective, high-performance applications for heterogeneous physical devices.

The Hydra project was a 54 month research and development project ending in December 2010 with 12 academic and industrial partners. The project was co-funded by the European Commission under the 7th Framework Programme.

Architecture

The middleware constitutes a software layer between the operating system of software enabled device and a user application that communicates with that device. The middleware provides protocols that execute on top of the transport layer and provide services to the application layer. "Hydra Managers" constitute the major building blocks that make up the middleware. A Hydra Manager encapsulates a set of operations and data that realise a specific functionality.

The LinkSmart middleware offers a large collection of reusable core software components to experienced developers. Based on these software components, programming abstractions allow for programming with well-known concepts from the field of networked embedded systems applications through reducing the complexity and details of the underlying implementation.

Technical features

The main technical components in the LinkSmart architecture are:

- Service-Oriented Architecture
- Model-Driven Approach
- 3-layered Discovery Architecture
- P2P-based Network Architecture
- Dynamic Runtime Architecture
- Context Management
- Self-* Management
- Security and Trust enabled
- Storage Management

All devices and services comprising the middleware have been integrated in a *Service Oriented Architecture (SoA)*, which effectively turns all devices into web services and thus provides extensive interoperability at the syntactic level, i.e. the capability of components to talk to each other regardless of the interface technology and their physical locations, is achieved by means of standard protocols from the world of web services, e.g. XML, XSL-t, SOAP, WSDL, XML Schema, WS-Security, WS-Addressing and several others.

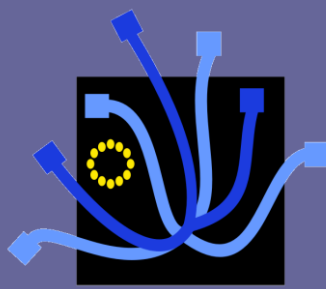
Due to the model-driven architecture, the middleware employs semantic technologies to manage metadata on devices and lower-level protocols to semantically resolve new devices as they enter a Hydra Network during run-time and automatically generate the software drivers for the web services.

The middleware distinguishes between powerful devices that are capable of running the Hydra middleware natively and smaller devices that are too constrained or closed to run the middleware. For the latter devices, proxies are used and once proxies are in place, all communication is based on the IP protocol. Ontologies are also used to create models of applications enabling context-related semantic support.

Project Outcomes

The tangible outcomes of the project are:

- The Software Development Kit (SDK)
- The Device Development Kit (DDK)
- The Integrated Development Environment (IDE)
- The LinkSmart Open Source Middleware
- The LinkSmart training package



1

A Project Methodology Based on a User-Centered Design Approach

WP2: Iterative user requirements engineering

User-Centered Design Approach

The Hydra project adopted an iterative (evolutionary) requirement engineering, specification and design methodology underpinned by a strong user-centric validation.

The methodology calls for comprehensive iterative requirements and stakeholder analysis based on initial requirements derived from scenario thinking. These requirements encompass the needs and priorities of the developer users as well as the wider diffusability and scalability requirements. The scalability requirements in turn had to take into account the technical and networking operational requirements as well as the testability, evaluation, marketability and exploitation of the middleware.

After the successful completion of a prototype cycle, each RTD work package analysed and reported their development results, RTD experiences, lessons learned in the development and integration work and other relevant knowledge gained during the development cycle. Moreover, knowledge gained from formal testing and system integration was collected together with latest development in technology, regulatory affairs and markets.

Once in every iteration cycle, the re-engineered requirements were documented in Change request re-engineering reports.

Overall, a total of 476 requirements were resolved and integrated into the LinkSmart middleware.

Requirements Engineering Process

The starting point of the iterative design process was a set of domain-specific Vision Scenarios delivering end-user visions of the future use of Hydra applications in three different domains: Building Automation, Healthcare and Agriculture.

Figure 1 shows an overview of the iterative approach.

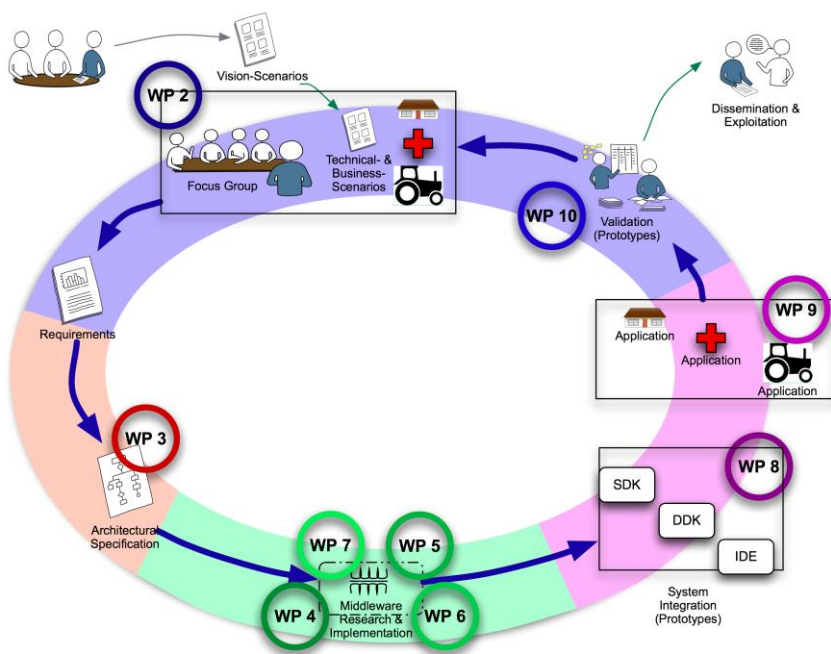


Figure 1 Overview of the iterative software development approach in Hydra

The Vision Scenarios were used to derive technical and business oriented usage scenarios that were discussed in focus groups with developer users. The result of this work was an initial set of requirements specifications for the Hydra middleware.

From the initial set of requirements, software experts specified the initial architectural specification, which drove the research and development work in middleware implementation and system integration. Software prototypes were demonstrated and validated in domain-specific settings with the aim to demonstrate the outcome of each cycle to developer users, end-users, project partners, reviewers, the research community, industry leaders, potential customers, etc. All results from validation and experiences gathered in the process were collected and used to refine the technical and business oriented scenarios, the requirements specifications, the middleware architecture as well as defining the new prototype specifications.

For each prototyping cycle, the middleware and the associated development tools progressively got more and more advanced as lessons were learned about developer user and end-user requirements. Evaluation of Lessons Learned from the previous cycles was taken into account. At the end of each annual cycle a set of prototypes were developed with the purpose of illustrating the following aspects of project progression:

1. Concept prototype for proof-of-concept
2. Software Development Kit (SDK) Prototype, incorporating part of Building Automation scenario
3. Device Development Kit (DDK) Prototype, adding further elements from the Healthcare scenario
4. Integrated Development Environment (IDE) Prototype, adding elements from the Agriculture scenario

Vision and technical scenarios

The first step was to develop the end-user Vision Scenarios for each user domain using a Delphi methodology for scenario building techniques. A series of one-day user workshops for each user domain was organised to bring together appropriate expertise and experience. The Vision Scenarios were used to derive Technical and Business oriented User Scenarios that were discussed in several focus groups with experienced developer users.

The Vision Scenarios proved to provide valuable input to the Technical Scenarios and the requirements gathering process. The Vision Scenarios were very useful for deriving requirements, which fully explore the uncertainties in long term projections in domains with rapid technological progress.

WP3: Specification of the Architecture

The Hydra middleware constitutes the software layer in-between the operating system and the applications. Another characterization in terms of the ISO OSI stack is that the middleware provides protocols that execute on top of the transport layer and provide services to the application layer. The specification of an appropriate system architecture for the middleware faces a complex mix of stakeholder, domain and technology requirements, while at the same time it should be as flexible, maintainable and extensible as possible to support as many future scenarios as possible. The application of an iterative process for the

Hydra project guaranteed a continuous update and revision of the system architecture with each iteration, in order to cope with such complexities.

The methodology applied for the specification of the software architecture was based on the standard IEEE 1471 "Recommended Practice for Architectural Description of Software-Intensive Systems" which defines core elements like viewpoint and view. In order to implement and execute this methodology, the specification of the system architecture follows the approach introduced by Rozanski and Woods [2].

Structural Overview

The software architecture is an abstract representation of the software part of the Hydra middleware. The architecture is a partitioning scheme, describing components and their interaction with each other. The upper-right part of Figure 2 gives a structural overview of the middleware and explains how the elements are logically grouped together. "Hydra Managers" constitute the major building blocks that make up the middleware. A Hydra Manager encapsulates a set of operations and data that realise a specific functionality.

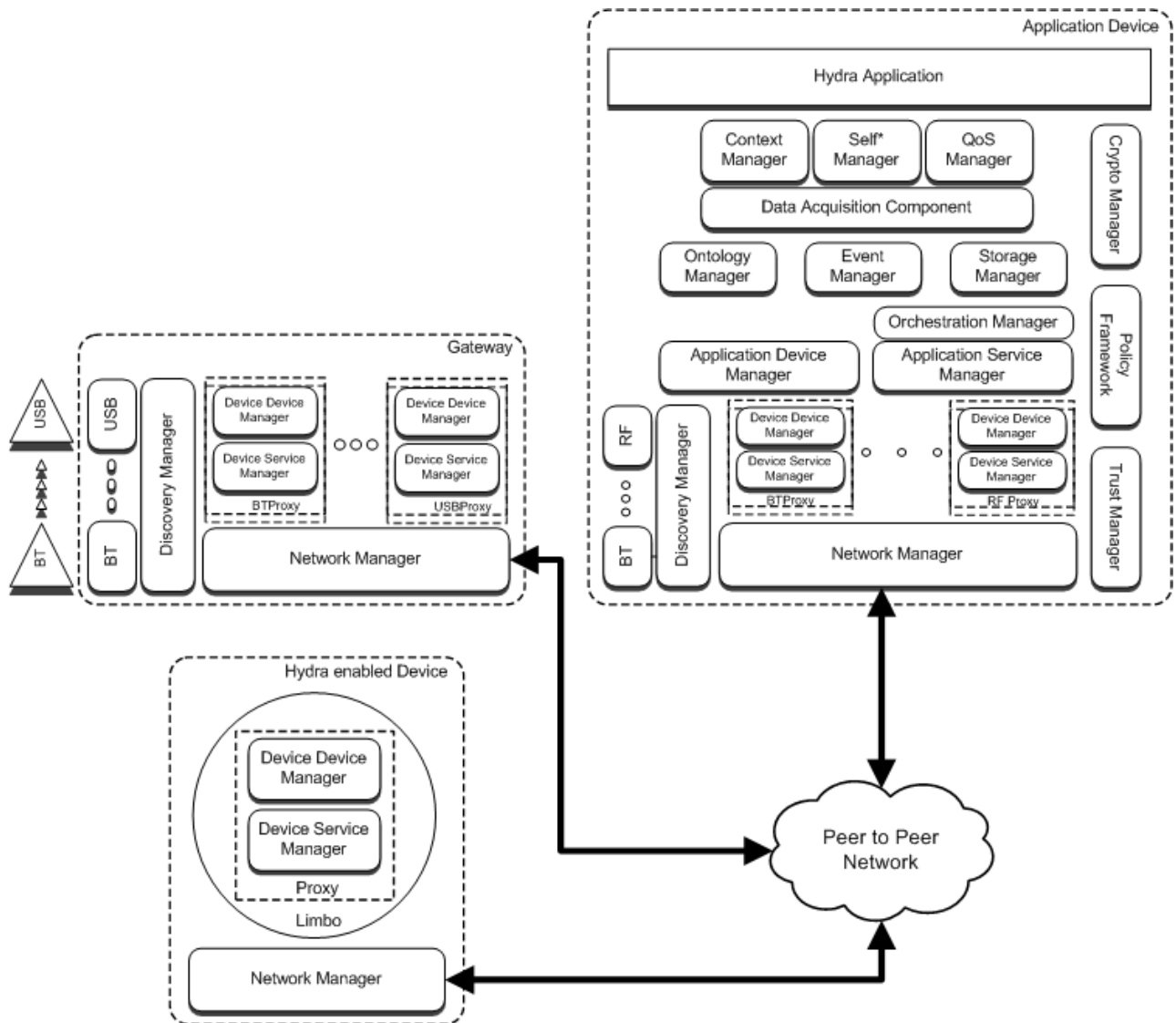


Figure 2 Allocation of Hydra Managers to Network Components

The Hydra Managers are enclosed by the physical communication layer and the application layer shown at the bottom and at the top of the diagram respectively. The physical layer realizes several network connection technologies like ZigBee, Bluetooth or WLAN. The application layer contains user applications which could comprise modules like workflow management, user interface, custom logic and configuration details. These two layers are not part of the Hydra middleware.

The Hydra middleware offers a large collection of reusable core software components to experienced developers. Based on these software components, programming abstractions allow for programming with well-known concepts from the field of pervasive and ambient computing through reducing the details of the underlying implementation. From the bottom to the top of the upper-right part of Figure 2, the middleware provides more and more programming abstraction and functionality for the developers:

- The Network Manager implements Web Service over JXTA as the Peer-to-Peer model for device-to-device communication.
- The Device Device and Device Service Manager in a bundle implement a service interface for a physical device, handle several service requests and manage the responses.
- The Discovery Manager automates and facilitates the discovery of devices in a Hydra Network.
- The Application Device and Application Service Manager provide programming interfaces and information for the different devices to the software developers.
- The Orchestration Manager supports the composition of services and workflows with a focus on energy efficiency aspects.
- The Ontology Manager is used by the Application Device Manager to get meta-information about devices and also semantically resolves what type of device has been discovered.

- The Event Manager provides a topic based publish-subscribe service in Hydra.
- The Storage Manager realises the persistent storage of information in the middleware. The Data Acquisition Component retrieves the data delivered by the sensors (via push or pull mode) and check the values for plausibility.
- The Context Manager allows for the definition of an application-dependent context model using key-value pairs or OWL/SWRL ontologies.
- The Self* Manager provides support for automating device management.
- The Quality-of-Service (QoS) Manager is a component that accesses and particularly processes all non-functional properties-data for services, components, devices, and network.
- The Crypto, Trust and Policy Manager take care for cryptographic operations, the evaluation of trust in different tokens and the enforcement of access control security policies.

Functional View

One to the major lessons learned during the elaboration of the software architecture was the strict following of design principles. Due to the complexity of the Hydra middleware and the large amount of contributors, who introduce their new ideas and conduct a constant refactoring of the architecture, it is essential, to keep in mind the predefined fundamentals for designing the architecture.

Therefore, the specification of the Hydra middleware architecture followed two important design principles, which explicitly influenced the software development process: structured design and separation of concerns. Both design principles aim at a minimization of inter-component coupling, and maximization of the intra-component cohesion, in order to increase the maintainability and understandability.

Inter-component coupling refers to the width and complexity of the interfaces between the components, and intra-component cohesion refers to the affinity or relatedness between the constituents of one component. The software components

of the middleware are internally loosely-coupled and show a strong cohesion among their internal constituents.

These design principles have been implemented in the functional view of a software architecture which defines the architectural elements that deliver the system's functionality. The Functional View contains functional elements, interfaces and external entities:

- *Functional Elements* constitute well-defined parts of the runtime system that have particular responsibilities and expose well-defined interfaces that allow them to be connected to other elements. A functional element can be a software component, an application package, a data store, or even a complete system.
- *Interfaces* are specifications, defining how the functions of an element can be accessed by other elements. An interface is defined by the inputs, outputs, and semantics of each operation offered and the nature of the interaction needed to invoke the operation.
- *External Entities* can represent other systems, software programs, hardware devices, or any other entity the system communicates with.

Applied to the Hydra middleware, the functional view defines the functional capabilities of the middleware, i.e. what the system is required to do.

Information View

During the constant evolution of the software architecture, the communication between the Hydra Managers turned out to be non-transparent. The introduction of an Information View tackled this lesson learned, since it specifically answers questions regarding what information is associated with each manager, how this information is represented and stored, and how this information is exchanged between the internal components of the manager. The information view on the software architecture required the modelling of data in order to illustrate and specify the composition of the middleware managers and the communication between them. The

models relevant for the Hydra middleware architecture are:

- *Static Data Structure Model* describing what kind of data the managers need for internal use and how this data looks like.
- *Data Ownership Model* describing which component is responsible for which data.
- *Information Flow Model* describing which data is exchanged between their internal components and the managers themselves.
- *Data Lifecycle Model* describing the transitions that data elements undergo in response to external events, i.e. the way data values change over time.

Deployment View

The initial version of the software architecture displayed a discrimination of the Hydra Managers into device and application elements. However, when working with the middleware it became clear that the managers can be deployed on any kind of network node. Since the Hydra components can be distributed over many different network nodes, it is almost impossible to specify an exact deployment model with delimited application- and device elements.

Thus, the deployment view on the software architecture provides a set of best practices that explain what a common Hydra Network can look like (Figure 2). The deployment view defines the physical environment in which the system is intended to run. This regards the selection of managers required for the operation of the desired application and the choice of the platform the respective manager will run on.

Integration of Devices in to a Hydra Network

The network components, that are present in a Hydra Network, comprise six basic entities:

- **Hydra-enabled device (HED):** this is a Hydra-compliant physical device that owns a software representation, i.e. a Hydra Device, in a Hydra Network. This device is not required to have IP

capabilities or to host the Hydra middleware.

- **Bridge:** this is a software component that resides in a Gateway and translates any non-IP communication into an IP based communication. It is used by Hydra-enabled devices with non-IP capabilities to communicate inside the Hydra Network.
- **Proxy:** this is a software component responsible for communicating with a resource-constrained device, understanding the technology used and the format of the data exchanged. It is deployed on a gateway.
- **Gateway:** this is a Hydra-enabled device with IP capabilities, which hosts proxies for resource-constrained devices. In order to communicate with such devices, a proxy running on the Gateway is needed. In this way, these devices appear as Hydra Devices in the Hydra Network.

In order to determine the way of how a new device is Hydra-enabled, i.e. incorporated into an existing Hydra Network, it has to be classified in one of five different capability categories D0-D4.

This classification prescribes the deployment procedure of (parts of) the middleware. First, we have to check whether the device can host the middleware or not. If the device is not powerful enough, it is a D0 or a D1 device. If the device can host a web service and has IP communication capabilities, it is a D1 device. Otherwise, it is a D0 device. On the other hand, if the device can host the middleware, we have to check if the device in question supports IP communication. If the answer is negative, we have a D2 device. If the answer is positive and the device can control D0 and D1 devices in the system, we have a D4. Otherwise, it is a D3 device.

The communication inside the Hydra Network is performed between D2, D3 and D4 devices, as they are able to host the middleware and has IP communication capabilities. D2 devices need a Bridge into a D3 or D4 device, or to a dedicated server to be incorporated in the all-IP Hydra Network. D0 devices are controlled by

Proxies in the Gateways so they can be accessed by other nodes in the network. D1 devices cannot host the Hydra middleware but provide IP support. We can face two different situations with this type of devices: either we can embed a web service on it or not. In the second case, the developer has to create a Hydra service using the DDK tools and contact the service provided by the device directly. This service is a Proxy that will be deployed on a Gateway. On the other hand, if the device allows embedding a web service in it, we can use the Limbo tools provided by WP4 work to create a simple web service that runs on the device. A client will be also generated by the Limbo tool and it will be integrated in a Hydra service. The generated Hydra Proxy will be integrated again as part of a Hydra Gateway.

Allocation of Software Components to Network Nodes

Figure 2 shows the concrete deployment of Hydra Managers on the specific network components. Each device is connected through the Network Manager. The gateway/D4 device (top left) further hosts a couple of proxies for D0 and D1 devices. The application runs on a laptop as the dedicated application device (i.e. a centralized architecture). One or several other network components may act as gateways (Class D4 devices). A gateway runs several Discovery Managers, which are logically part of the Application Device Manager. The Discovery Managers discover non-Hydra-enabled devices (D0 and D1) and makes them available within the Hydra Network by providing a Proxy for them.

The Hydra OSGi Bundle Architecture

In order to implement the above mentioned design principles, the OSGi Service Platform has been applied to the Hydra architecture. Through its strict component-based approach, the OSGi implementation in the Hydra architecture enables the development with modular and exchangeable software components. The OSGi environment is responsible for lifecycle management of components, (local) service discovery, deployment and dependency management of components. Bundles can detect the addition or the removal of services via the OSGi service

registry. The OSGi implementation in the Hydra architecture applies these concepts and realizes Hydra Managers in Java as OSGi bundles.

Architectural Perspectives

During the constant evolution of the Hydra middleware it became obvious that runtime aspects need to be expressed and the interplay of architecture components needs to be demonstrated.

The introduction of new architectural perspectives was one of the major improvements of the last description of the architecture specification. Rozanski and Woods [2] propose and define several perspectives on the system architecture that ensure the quality properties of the architecture are recalled in the process of architecture specification. The architectural perspectives address cross-view considerations of these quality properties, and thus, the description of the architectural perspectives for the Hydra middleware is based on functionality that runs orthogonally with the views presented earlier.

Each perspective covers sequence diagrams of the messages sent between the Hydra Managers participating in the process. Six architectural perspectives were captured: Communication, Device Discovery, Security, Storage and Context Awareness, and Self Management Perspective.

Validation of the Hydra Software Architecture

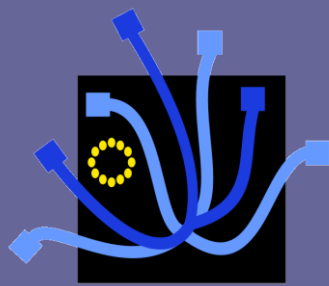
For the final iteration of the Hydra project, an assessment of the quality of the software architecture of the middleware was conducted. The demanded quality of the software architecture was early captured in requirements that requested the compliancy of the Hydra architecture with the OASIS Reference Model for Service-Oriented Architectures and with the Web Service Architecture (WSA). The compliancy with both of these specifications guarantees a high quality of the software architecture specification.

The OASIS Reference Model for Service-Oriented Architecture [3] is an effort to define the concepts that make up a SOA and it provides some guidelines that can be

applied to assess existing software architecture. This reference model does not describe implementation or architectural patterns, but it aims at providing a common semantics instead that can be applied to any kind of SOA. Web Service Architecture [4] (WSA) specified by the W3C Working Group defines the architecture of a Web Service based SOA implementation and provides a huge set of concepts and relationships to define all components of a Web Service model in a very exhaustive manner.

The concepts of SOA described by OASIS can be identified in the Hydra architecture, because it addresses the important concepts of services, identification, interaction etc. From the Web Service implementation side, Hydra makes heavy use of existing standards based on XML as proposed by WSA. Using such common standards like WSDL, UDDI, SAWSDL has several advantages e.g. being compliant with other application applying these standards.

Nevertheless, WSA does not dictate the usage of these technologies and Hydra employs custom implementations if the requirements call for new solutions. For example, UDDI did not fit to the Hydra way of having semantic devices and services in highly dynamic environments. Thus, custom implementations of service registry have been developed



2

Technical Overview

The main technical components in the Hydra architecture are:

- Model-Driven and Service-Oriented Architecture
- 3-layered Discovery Architecture
- P2P-based Network Architecture
- Context Management
- Self-* Management
- Security
- Storage Management
- Runtime Architecture

WP6: SoA and MDA middleware

Model-Driven and Service-Oriented Architecture

All of the devices and services comprising the Hydra middleware have been integrated in a *Service Oriented Architecture (SoA)*, which will provide, among other things, interoperability. The middleware thus becomes the link between web services and devices. Interoperability, which here is taken as the capability of components to talk to each other no matter what is the technology used to implement them or their physical location, is achieved by means of the usage of many specifications in the context of the web services world, including XML, XSL-t, SOAP, WSDL, XML Schema, WS-Security, WS-Addressing and several others. To summarise, the main purpose of the Service-Oriented Architecture in Hydra is to provide interoperability between devices at a *syntactic level*.

The Hydra middleware aims to interconnect devices, people, terminals, buildings, etc. As mentioned above, the Service-Oriented Architecture and its related standards provide interoperability at a syntactic level. However, one of the goals of the Hydra middleware is to provide interoperability at the *semantic level*. This is achieved through a semantic model-driven infrastructure, whereby services exposed by devices can be described and consumed by various Hydra applications. The overall SOA and MDA functionality is facilitated by:

- Device Modelling using Ontologies
- A Device Application Catalogue
- Automated Device Discovery
- Web Service Enabling of Physical Devices
- Publish-and-Subscribe Based Event Management

Device Modelling using Ontologies

The semantic model driven architecture (MDA) is exploited both in design-time and in run-time,

- At design time, developers are provided with rich class libraries and semantic descriptions in a Device Ontology.

- At run-time the middleware system uses metadata on devices and lower-level protocols in order to semantically resolve new devices as they enter a Hydra Network. The system is able to automatically generate the proper software drivers (Hydra devices and service managers) providing a web service interface to the devices.

semantic description of a device model has been found, then its device capabilities can be accessed.

Ontologies are used to model devices, their services, capabilities, security requirements but also the applications and parts of the middleware itself.

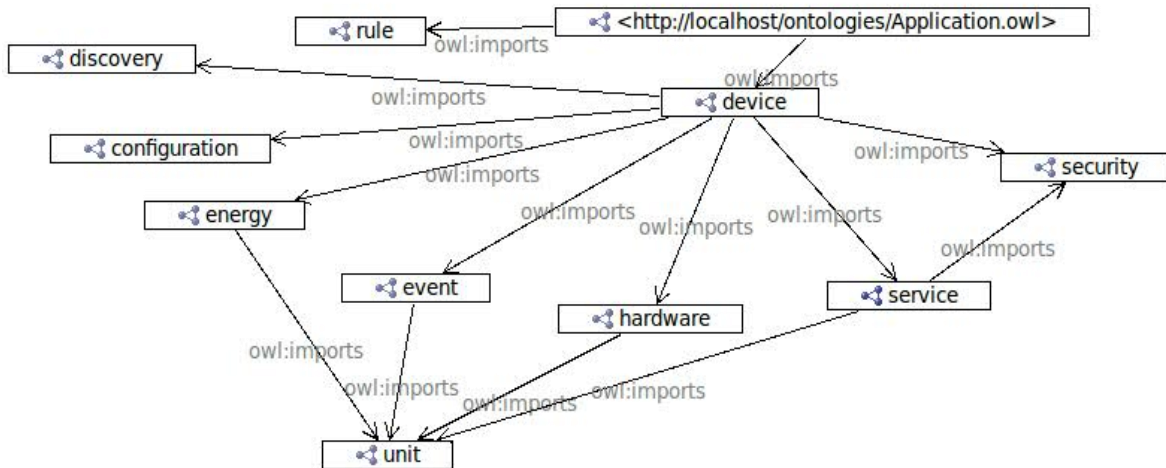


Figure 3 Relations between the Device Ontology subsets

A main feature of the Hydra middleware is to bring semantic web technologies down to the device level, i.e., each device can act as a semantic web service accessible by other devices, users and software applications.

In order to cope with the huge variety of capabilities of the devices to be integrated using the middleware, two broad options can be considered: a) to force every device to be compliant with some more or less flexible interfaces, or b) to have the middleware layer provide adaptation to whatever interface the devices offer.

Since choice a) will probably not be applicable neither to the present nor to the future real world, the Hydra project has opted for choice b) in such a way that the middleware is able to adapt to the variety of interfaces, information and operations that devices offer.

In order to implement semantic interoperability, the middleware has introduced descriptions for the devices in such way that an automatic agent can understand their capabilities and use them. Once the

The Hydra Device Ontology represents concepts describing device related information, which can be used in both design time and in run time. The basic ontology is composed of several partial models representing specific subsets of device information.

The initial device ontology structure was extended from the FIPA device ontology specification. The initial device taxonomy was extended from AMIGO project vocabularies for device descriptions.

The relation between the Device Ontology subsets is shown in Figure 3

The components of the Device Ontology can be shortly described as follows:

- Core Ontology (Device.owl)*: contains a taxonomy of various device types and the basic device description, manufacturer and model information.
- Device Capabilities*: represent the hardware properties (Hardware.owl, Network.owl) and software description (SoftwarePlatform.owl divided into DotNet.owl, Java.owl and OperatingSystem.owl ontologies)
- Device Services (Service.owl)*: describes the models of device services in the terms of operation names, inputs and

outputs. The device services are connected to the Quality of Service ontology (QoS.owl, QoSSpec.owl, Unit.owl) used to annotate the services and their parameters to several quality factors.

The services of a device are further divided in different categories, which are made available to the developer in the DAC: a) a generic set of services providing access to various device and service metadata and b) a number of device type specific services.

- *Device events* (Event.owl): provides the descriptions of events, which can be generated by the simple devices, as the alternative of providing the functionality. Events can be annotated to the quality of service ontology in the similar way as the services.
- *Device Malfunctions* (Error.owl): represents the various types of errors and failures which may occur when using the device at run-time
- *Self-* Properties* supporting models: models of state-machines tracking the run-time device/service state changes, model of device run-time request/response tracking (IPSniffer.owl, StateMachine.owl) and SWRL rules supporting mainly the self-monitoring and self-diagnosis processes.
- *Security Ontology* (securityMain.owl): represents the various security properties, such as protocols, algorithms (securityAlgorithms.owl), objectives and assurances (securityAssurance.owl), which may be attached to devices or services. To describe the security properties, the third party NRL ontology was reused, modified and connected to the device model.
- *Discovery models* (Discovery.owl): used for semantic resolution in the semantic discovery process.
- *Application model* (Application.owl): describes the model of application and the entities used in various applications, such as locations or persons (Location.owl, Coord.owl, SetLocation.owl, GraphLocation.owl)

The ontology architecture was designed to support the maintainability and future extensions of used concepts. The

ontologies have been developed using the OWL language. The references between more general and specific concepts and modules (related ontologies) are realized using the OWL import mechanism.

In design-time, every ontology module can be further extended by creating new concepts according to the needs of representation of the new information about new device types and models. The concepts can also be further specialized. For example, if a new device type is needed, the adequate concept in the device classification module can be further subclassed by more specialized concepts and the new properties can be added.

The device ontology is populated using the DDK tool, which is used to create new device types (in terms of concrete device models, e.g. an HTC3000 phone). The DDK tool creates only basic device information such as the manufacturer information, services and device low-level discovery information.

In order to enrich the device model semantic description, the Ontology Manager IDE has to be used to add all additional device properties, such as events, hardware or energy profile, security and QoS properties. In this way, a device model template is created. The Device templates created at design time are used at run-time to create application specific device instances. Each time a new device is discovered, the ontology is used to infer the most suitable device template using primarily the low-level discovery information and the Ontology Manager creates a device specific clone – a run-time instance assigned to the concrete physical device. Each physical device has its own run-time instance. Once the physical device leaves the network, the related ontology run-time instance is removed.

The amount of additional information added to device model enables the support of various semantic queries used in application logic or supporting the semantic devices behaviour.

Ontologies are also used to create models of applications enabling context-related semantic support. The Ontology Manager IDE behaviour is also driven by the ontology model.

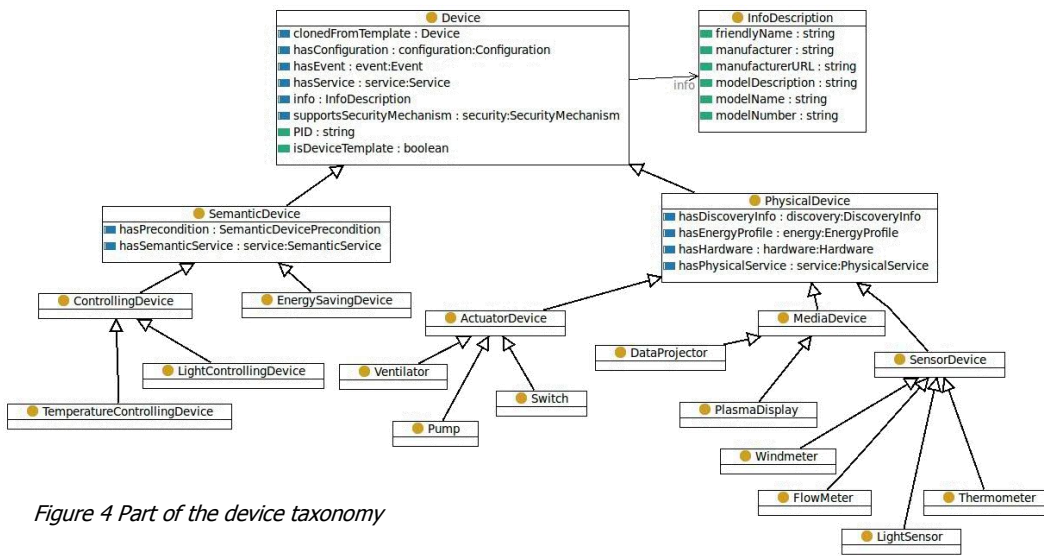


Figure 4 Part of the device taxonomy

Device Application Catalogue

The Application Device Manager handles all knowledge regarding devices that have been discovered and are active in the Hydra Network. The Application Device Manager knows about devices from a network perspective but does not handle the locations or context of the devices. The Application Device Manager's main functions are discovery of new (and existing) devices, semantically resolves the device type and available services based on the Device Ontology, creates a service interface for the device, manages semantic device descriptions, provides semantic device aggregation and manages the Device Application Catalogue (DAC).

Device Discovery is one of the major functions of the Application Device Manager and the aim is to discover new devices in the network. It will support user-initiated discovery as well as automatic schemes. DAC keeps track of and manages all devices that are currently active within one application. It is a view on the Device Ontology. It can be queried about existing devices and their status. It can also provide service interfaces for the different devices upon request. The DAC will also keep track of when the device entered the system, when it was last heard of and its current state.

The Application Device Manager is responsible for generating a service interface for a certain device. It will create web services as well as UPnP services.

Web Service Based Device Communication

Hydra based applications are built by programming networked ambient intelligent

devices. Devices are made programmable by the middleware through proxies as well as by embedded components.

Whatever the method, it is transparent to developers, as they access all devices based on a pure service and event based programming model. In order to support open and dynamic networks, the device protocols need to provide descriptions of the capabilities of the supported devices. This includes device identity and functional interfaces (services) and possibly also additional information such as details about the manufacturer, the model and the version.

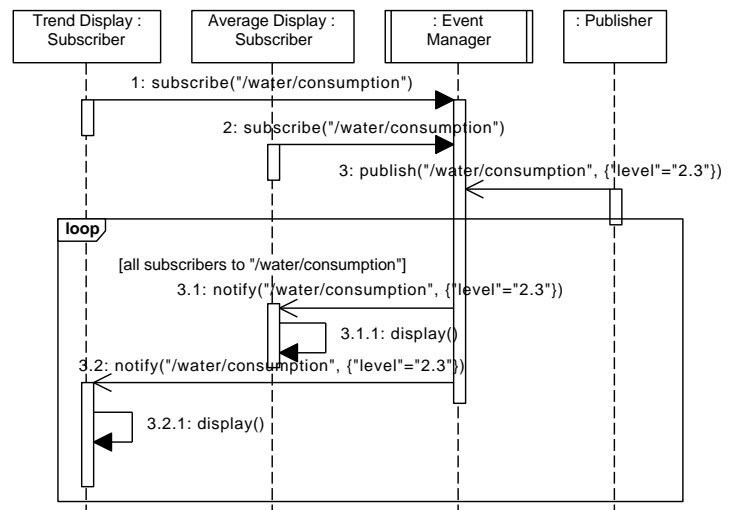


Figure 5 Event Management Scenario

Powerful instruments for device modelling and description are central in the Hydra architecture, as in all networks of devices and the "Internet of Things". .

A Hydra Device is a software representation of a physical device. This representation is either implemented by a proxy running on a gateway device, or, by embedded Hydra Managers on the actual device. A Hydra Device is said to Hydra-enable a physical device.

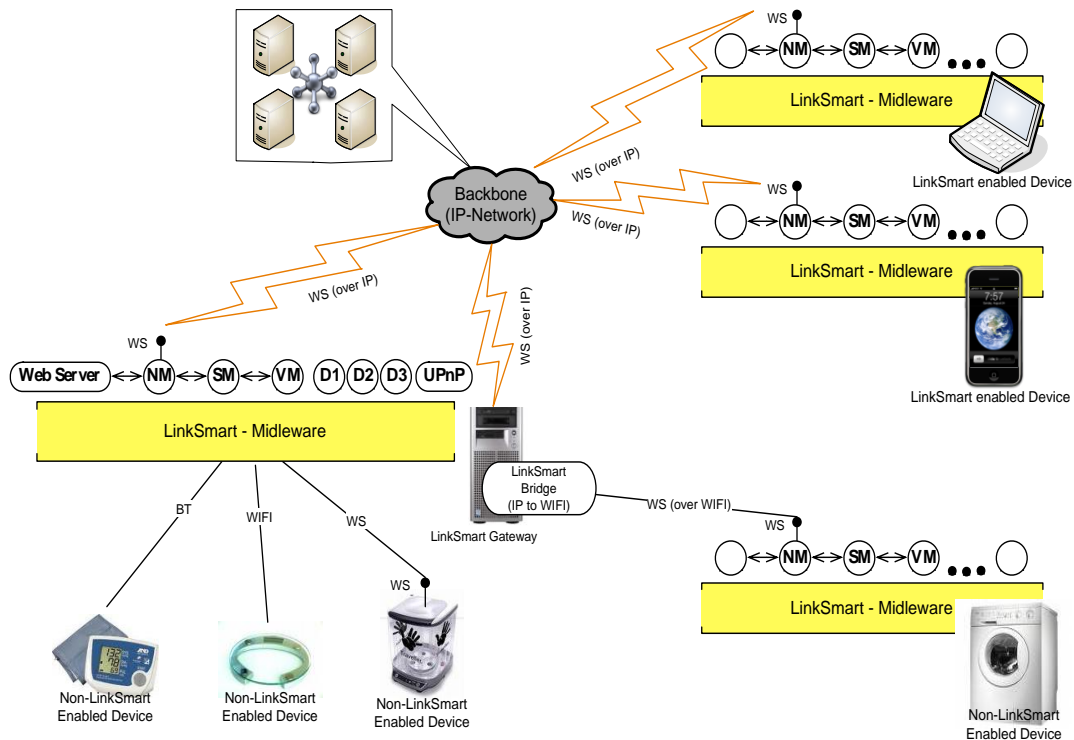


Figure 6 Communication inside a Hydra Network

Web Service Based Eventing

The Event Manager implements publish/subscribe in the middleware. Publish/subscribe is a (distributed) communication paradigm in which senders (publishers) and receivers (subscribers) of messages (events) are loosely coupled through decoupling in space, time, and synchronization. Decoupling in space means that publishers and subscribers can reside in different processes or on different nodes. Decoupling in time means that publishers and subscribers do not need to be running at the same time. Decoupling in synchronization means that there are no requirements on publishers waiting for subscribers to receive messages or vice versa.

The Event Manager provides decoupling in space and synchronization through a content-based publish/subscribe mechanism. In this type of publish/subscribe, subscribers subscribe on topics and receive events that are published by publishers on that topic through a notification mechanism.

Figure 5 shows a typical interaction with the Event Manager. In the scenario, a Water Meter (a publisher) periodically publishes its measurements on the topic /water/consumption. The different user interfaces (Trend Display and Average Display) are subscribers and are notified when the Water Meter publishes events.

WP5: Wireless networks & devices

Device Network

The Hydra middleware distinguishes between powerful devices that are capable of running the Hydra middleware natively and smaller devices that are too constrained or closed to run the middleware. For the latter devices, proxies are used and once proxies are in place, all communication is based on the IP protocol.

Figure 6 presents an example of a Hydra Network and illustrates the two cases: On the right we have a device terminal that can host the middleware and is able to establish communication with services on the platform.

On the left, the devices cannot operate the middleware (because they are too resource constrained or have proprietary interfaces). In this case, proxies are created on the node (in this case a mobile phone). The proxies virtualizes the device vis-à-vis the platform. Any service will think it is communicating with the device, where in fact it is communicating with the proxy.

3-layered Discovery Architecture

The Hydra middleware provides a discovery architecture that builds on UPnP technology. The approach implements a three layered discovery architecture that includes physical device detection, UPnP network announcement and semantic resolution of devices against a device ontology.

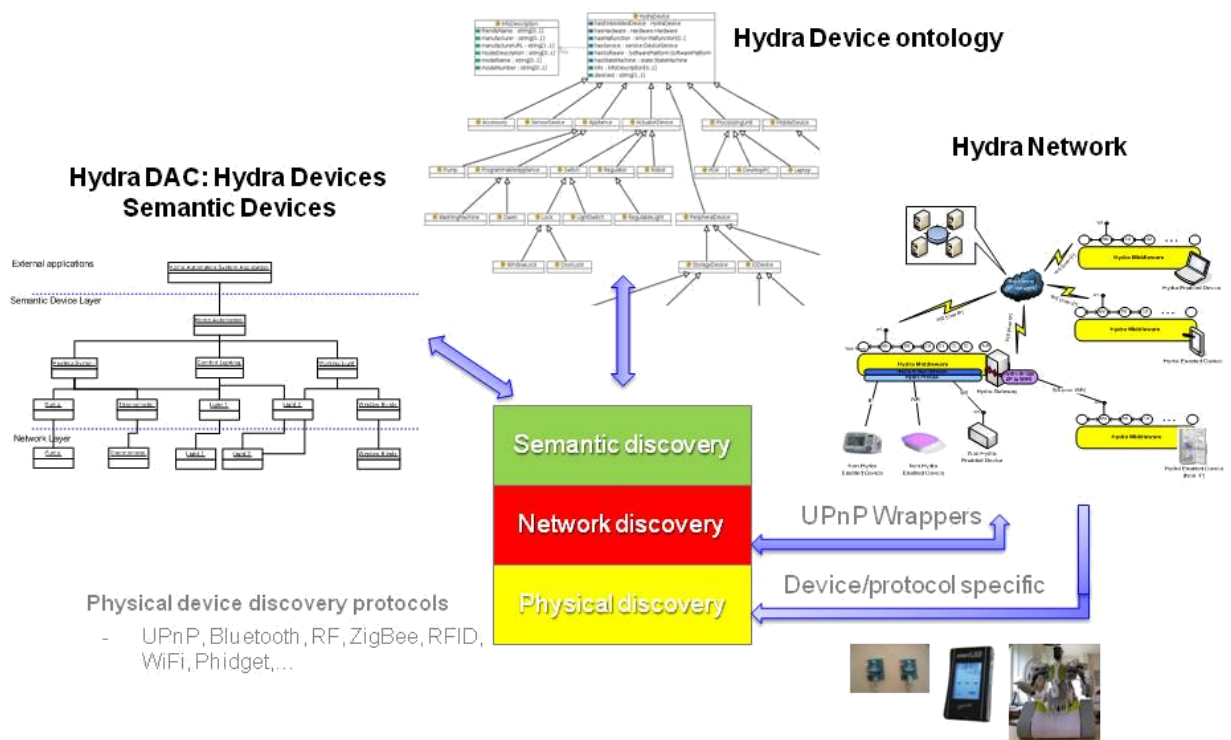
The model driven architecture (MDA) implements the device discovery process. This architecture is structured in three layers abstracting the discovery functions. The discovery process operates both locally and remotely, so that devices that are discovered in a local Hydra Network can also be discovered in a peer Hydra Network over the P2P protocol implemented by the Hydra Network Manager.

discovered physical devices in the local network and to peer networks.

At the top-most layer the device type is resolved against the Device Ontology and is mapped to some Hydra Device type. It is then placed in the Device Application Catalogue (DAC). If an application subscribes to events regarding this type of device, it will be notified that the device is available and has been placed in the DAC.

The middleware provides: 1) discovery mechanism, 2) low level protocols, 3) service execution, 4) virtualization, and 5) security and trust policies which can directly be used by the developer of applications.

The middleware incorporates support for self-discovery of devices. When a Hydra enabled device is introduced the middleware is able to discover and



configure the device automatically.

Figure 7: The 3-layered Discovery Architecture is part of the Hydra MDA

The lowest discovery layer implements the protocol specific discovery of physical devices. This is performed by a set of specialized discovery managers listening for new devices at gateways in a Hydra Network. The second layer uses UPnP/DLNA technology to announce

P2P-based Network Architecture

The Network Manager implements the Device to Device (D2D) communication between middleware instances. In this context, we define the communication as data exchange between devices "inside" a Hydra Network, which are Hydra enabled and have IP communication capabilities.

In Figure 6 above was shown how Device To Device communication takes place inside the Hydra Network. This communication is based on a P2P architecture, where JXTA was chosen as the reference implementation in which the D2D communication will rely on. The Network Manager is composed by different sub-managers, named:

- Routing Manager, is in charge of sending and receiving the information packages between Hydra instances
- Identity (HID) Manager, manages the service identifiers (HID and Crypto HID in the Hydra Network)
- Backbone Manager, is in charge of managing the P2P network
- Session Manager, manages the session control in communications between Hydra instances

The SOA is implemented using web services, where information is encapsulated in SOAP messages. Devices and applications running the middleware offer and consume services. Traditional WS architectures are based on client-server architectures, where the server is an always-on end system with a well known endpoint address, which should be known by clients beforehand (using either service descriptors or UDDI registries). However, in a pervasive and distributed environment, all device and service endpoints cannot (and for security reasons should not) always be known a priori.

In Hydra, a SOAP tunneling approach proposes a way to replace the client-server architecture for a distributed one, using the Network Manager P2P platform. In this architecture, all the peers will act as clients and servers at the same time, and will be able to offer and consume services in a transparent manner.

Moreover, the Network Manager has been extended to provide multimedia content exchange between DLNA devices. This functionality has been incorporated in the Backbone Manager as the P2P network is used for content transmission.

The Network Manager (on a device running the Hydra middleware) acts as a proxy for the DLNA devices, providing a similar functionality as the SOAP tunnel but for multimedia content, using JXTA sockets. When a content request from a DMR (Digital Media Renderer) device reaches the Network Manager multimedia interface, it is routed to the destination DMS (Digital Media Server) using the overlay network and the HID addressing mechanism. When the request reaches the DMS the desired content is returned to the DMR for playing using the same communication path. Using this functionality, any application running on top of the middleware is able to search for content on any DMS in the Hydra Network and play this content on any DMR device.

The structure of the Network Manager is shown in the diagram in Figure 8.

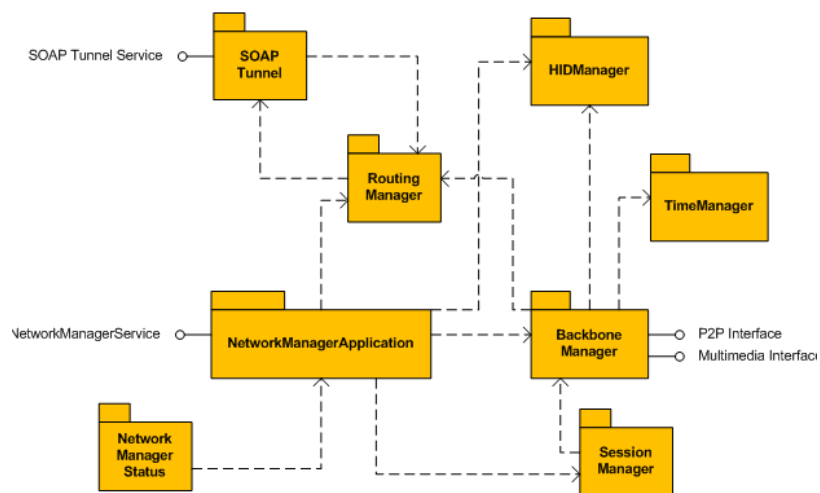


Figure 8 Network Manager component diagram

Context Management

In the Hydra project, we use context information, retrieved from sensors and put together by low historic level context information, in various parts of the middleware and applications. Acquired data is also used in the Policy Framework, as input to Quality of Service (QoS) functions and for enabling self-*-properties. For this

purpose, a Context Awareness Framework (CAF) has been developed as part of the middleware. The CAF is not a part of the security architecture as such, but provides essential information for a number of the security mechanisms described below.

Figure 9 shows the Context Awareness Framework. The graphical rule creation application uses data provided by the Device Application Catalogue (DAC) which holds information about accessing the devices, like method names and ids to access them. The information put together by the rule creation application is used as an input to configure the Data Acquisition Component (DAQC), a separate component used by the Context Manager. Together they form the Context Awareness Framework.

The Context Manager models contextual information and runs a Rule Engine to perform the active situational awareness and reasoning, as defined by applications.

Memory' architecture. Rules inserted into the Rule Engine, and evaluate the modeled context to trigger actions. The Rule Engine used is the DROOLS Rule Engine, allowing for object-oriented context modelling. Additionally, the Rule Engine supports Complex Event Processing, so that it can support rules with temporal reasoning and sliding windows over events.

Device Contexts are created in the Context Manager once the Device Discovery process between the Device Application Catalogue, the Discovery Manager and the Ontology Manager has been completed. The Context Manager receives the resolved device definition, from which the new context is modeled, and then data from the device is subscribed to.

Application Contexts are installed by the application, containing the context definition and a set of associated rules to perform the application specific context reasoning. These rules specify context-

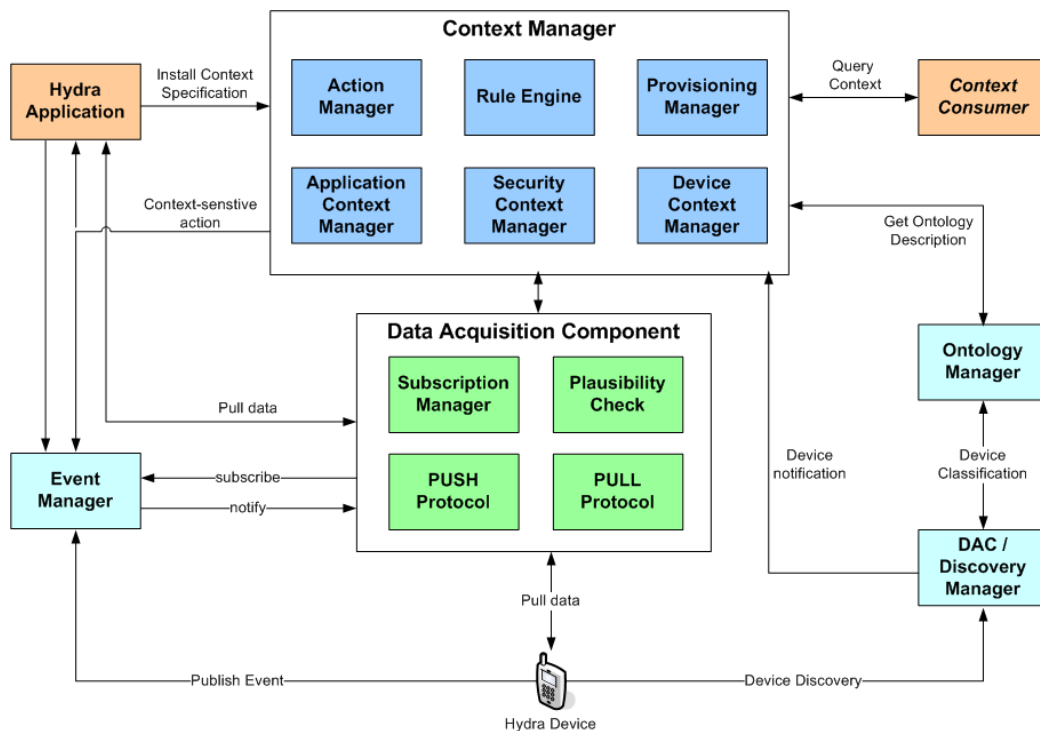


Figure 9 Context Awareness Framework

It provides a query interface for provisioning context as encoded XML. The Data Acquisition Component retrieves the data that is to be modeled by the Context Manager.

The Rule Engine maintains the modeled contexts in a blackboard-styled 'Working

sensitive actions as outputs, which include publishing an event to the Event Manager and calling a Hydra-enabled service.

The Device Discovery Manager identifies a device by retrieving the services and information provided by the device. It then connects to the Ontology Manager, which

infers the device class based on the given information and the device ontology. Finally, the information is put in the DAC and passed on to the DAqC.

The Data Acquisition Component has all the information about the device to be contacted and passes on the parameters stored in the Device class, it can decide when to contact the device and at which frequency in order to get the latest status information from the device and populates the device class. Each time new values/parameters are added to the device class, this structured data is passed to the Context Manager.

The Context Manager will contact the Ontology Manager to update the object of the corresponding ontology class with the latest information that it has in-hand and in-return the Ontology Manager sends the Context Manager with the latest Context Information related to the data being updated (in other words, the Ontology Manager sends information from classes which are linked to the class being updated). After obtaining the latest live contextualized data, the Context Manager sends the data to the Storage Manager, which maintains the history of the device status.

WP4: Embedded AmI architecture

Self-* Management

For ambient intelligence systems to be really useful, they arguably depend on the ability to reason about and modify their own state to adapt to context changes. As an example, Quality of Service (QoS) requirements may imply the need to keep a system running for as long as possible given that some devices may run out of battery. As different communication protocols have different QoS properties, in terms of power consumption throughput, reliability, etc., the prolonging of system life can then be achieved through the switching of communication protocols among some of the devices in order to save battery.

To achieve this, the Hydra Self-* Manager builds on the three layer self-management architecture of Kramer and Magee as shown in Figure 12.

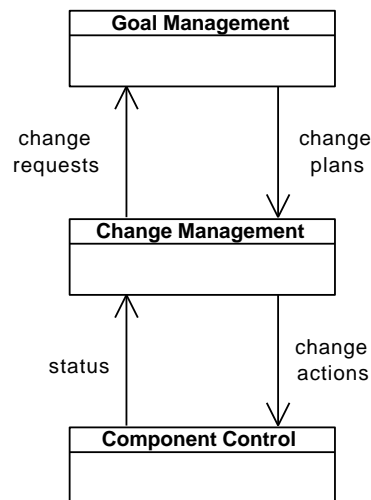


Figure 10 Hydra Three-Layer Self-Management Architecture

These three layers (in order of computational expense) are:

1. *Component Control*: The lowest layer is the Component Control layer. It is responsible for retrieving information about the state of the system, e.g., which/what services exist and what their states are. It is also responsible for actuating low-level change commands issued from higher layers, e.g., installing and starting a service. This is also the layer in which actual system/application services run
2. *Change Management*: The middle layer is the Change Management layer. It is responsible for detecting situations that need to be managed, and to perform that management according to pre-determined schemes by issuing commands to the component control layer. A scheme in this context can be a plan such as a set of rules reacting on events from the Component Control layer
3. *Goal Management*: The top layer is the Goal Management or planning layer. When a situation is detected for which there is no applicable pre-existing scheme in the change management layer, the goal management layer is responsible for computing a new scheme, or plan, e.g., an AI planner can be used to dynamically generate a reconfiguration plan that is sensitive to the constraints set by the current system state and policies. Ideally, high-level policies express how to create plans.

A typical deployment is shown in Figure 12. Communication between layers is done using the Hydra Event Manager. Within the Component Control layer, Architectural Query Language (AQL) sensors implement status sensing while Architectural Scripting Language (ASL) interpreters implement change actions. On the Change Management layer, an OWL ontology (SeMaPS) and supporting libraries model the runtime context of systems. The

Reasoner component uses SeMaPS to perform SWRL reasoning on runtime states. Upon need for plan change, the Goal Management layer Optimizer is invoked, finding an optimized system state. The Planner uses the current system state and the optimized system state to find an ASL script to take the system into the optimized state.

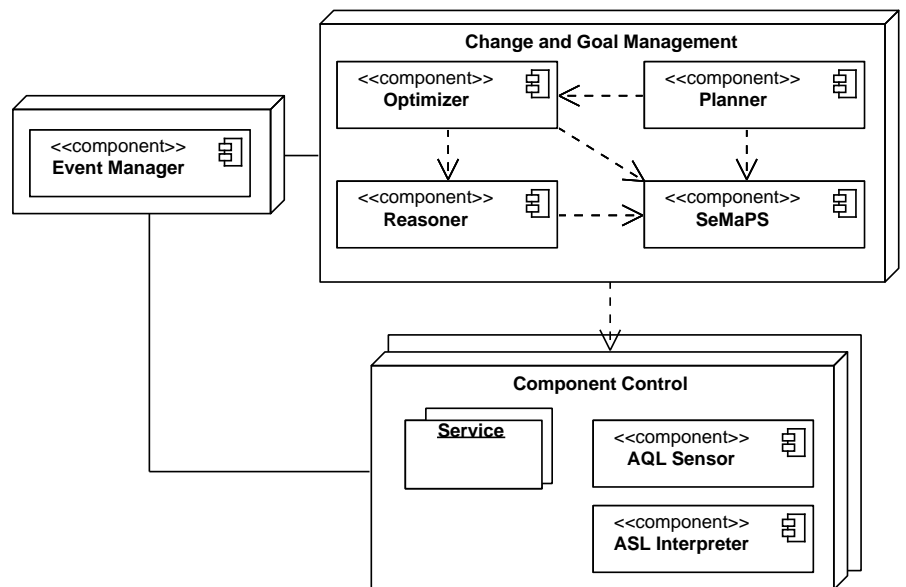


Figure 11 Self-Management Deployment

WP7: Trust, privacy and security

Security Framework - Access Control Policy Framework

A foundational aspect of the security framework architecture is the support for access control at every layer of the middleware. Hydra provides a dedicated set of components for controlling access to services provided at the middleware level, realised as a dynamic, flexible and extensible access control mechanism to facilitate interoperability whilst ensuring that only authorised principals are allowed access to protected resources. Additionally, the Access Control Policy Framework can be utilised at application level.

The Hydra Access Control Framework is policy-driven: access control policies are used to define and enforce resource access security. Hydra uses the declarative eXtensible Access Control Markup Language (XACML) format to define and evaluate

access control policies. The Hydra Access Control Framework implementation realises the sub-tasks of an access control policy framework as defined in the XACML processing model specification: policy definition, policy administration, policy retrieval, policy information, policy evaluation and policy enforcement.

the selected Policy Decision Point (PDP), which evaluates the Access Request against XACML policies stored in a Policy Repository, and returns a decision to the PEP. While evaluating a request, policies may refer to attributes that aren't present in the request itself, but can be retrieved from other sources, namely Policy Information Points (PIPs).

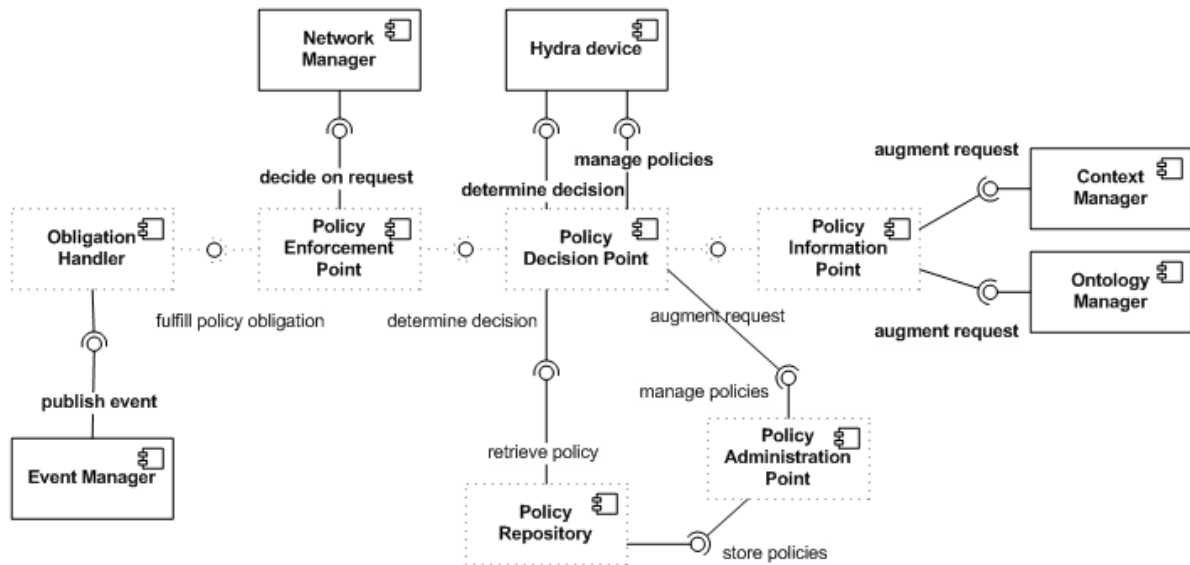


Figure 12 Access Control Policy Framework.

The Network Manager as the gateway component for communication with Hydra services / devices provides the natural point of interception for the Access Control Policy Framework. By intercepting communication at each Network Manager, all Inside Hydra communication can be evaluated *before* it is forwarded on to the recipient and *after* the communication has been securely transmitted through Inside Hydra communication. This is shown in Figure 14, where the process begins when the Network Manager receives a call to one of it's hosted services. It forwards the credentials of the call - Subject and Resource HIDs with their associated CryptoHID certificates, as well as the SOAP Message payload - to the Policy Enforcement Point, for a Permit / Deny decision to be returned.

The Policy Enforcement Point (PEP) formulates the credentials into an XACML Access Request by parsing the SOAP Message to extract the method of the action call as well as the content of any arguments. The PEP submits the request to

Two PIPs were developed, as shown in Figure 12, interacting with the Ontology and Context Managers, to retrieve attributes specified by access control policies (e.g. Application Contexts). The PDP features an extension mechanism, so that additional PIPs can be easily implemented and dynamically integrated with a PDP at runtime.

The Access Control Policy Framework also features an extensible Obligation handling mechanism for handling obligations specified in a policy. Obligations are typically application-specific actions that are to be taken in addition to enforcing an access control decision, for instance logging denied access requests. As with PIPs, these can be easily created and added to the middleware runtime. The middleware bundles several default obligation handlers, such as an obligation handler that fires an event to the Event Manager for further propagation and processing.

Storage Management

The Hydra Storage manager is designed as a generic storage abstraction for storing data over a Hydra Network, in which devices or other applications shall be able to access this storage. The architecture was designed to support a wide area of storage, e.g. block, file system or key-value storage, each persistent or non-persistent.

Figure 13 shows the main components. The Storage Manager is modelled as a device responsible for any kind of configuration of storage on a Hydra enabled device.

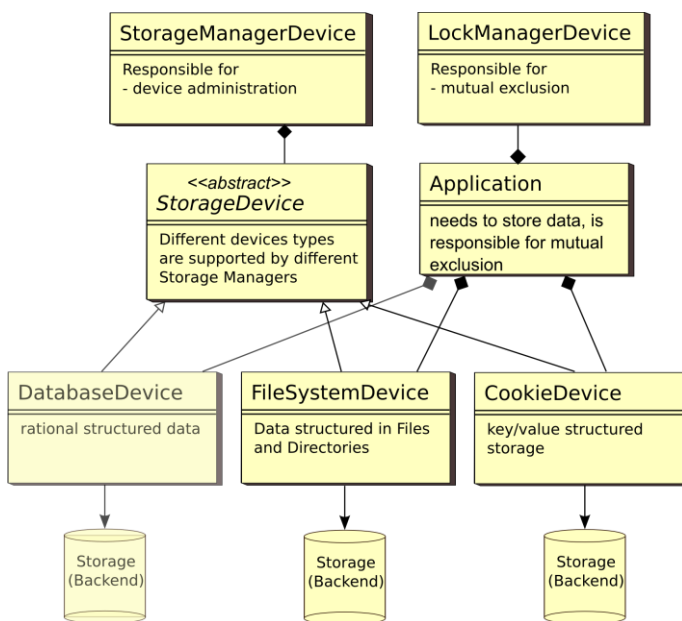


Figure 13 Basic components of Hydra Storage Architecture

The storage itself can be accessed using some kinds of storage device. Such devices differ in the way they represent storage. A Block Device can be used to access sequential data stored in block devices or single files. A File System Device could provide data structured in files and directories. Furthermore, a Database Device can be used to access data stored in databases.

The implemented devices are realized as UPnP-Devices created by Limbo (web service compiler), implemented in Java and deployed as a set of OSGi bundles.

Figure 14 shows the implemented devices, which can be used by the applications to store data.

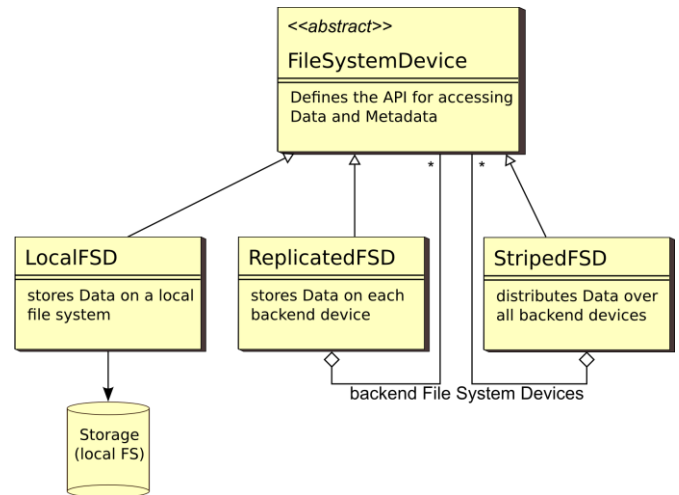


Figure 14 File System Devices

The Local File System Device is a very small implementation that delegates the File System Device API down to a local file system (or a directory in a local file system).

The Replicated File System Device and the Striped File System Device take a number of existing File System Devices as backend storage. While the Replicated File System Device mirrors all data on each backend device, the Striped File System Device distributes its data over the backend devices without redundancy.

Hydra Runtime Architecture

Figure 15 below illustrates local discovery of physical devices as well as P2P discovery between two local networks and is described by the following points:

- A Bluetooth phone comes into a local network (lower right).
- The phone is discovered by the Bluetooth Discovery Manager running on site B. The Bluetooth Discovery Manager extracts as much information from the phone as possible and forwards it to the Ontology Manager at site B.
- The Ontology Manager reasons and concludes it has found a device of type Basic Phone, it instructs the Bluetooth Discovery Manager to create a proxy and an interface for such a device.
- The BT Discovery Manager creates a Hydra Device that consists of a Phone DeviceManager and a Bluetooth Device
- Service Manager. The Hydra Device exposes the phone functions (SendSMS, ReadSMS) as web and UPnP services

Hydra run-time architecture: application with remote (P2P) device instance

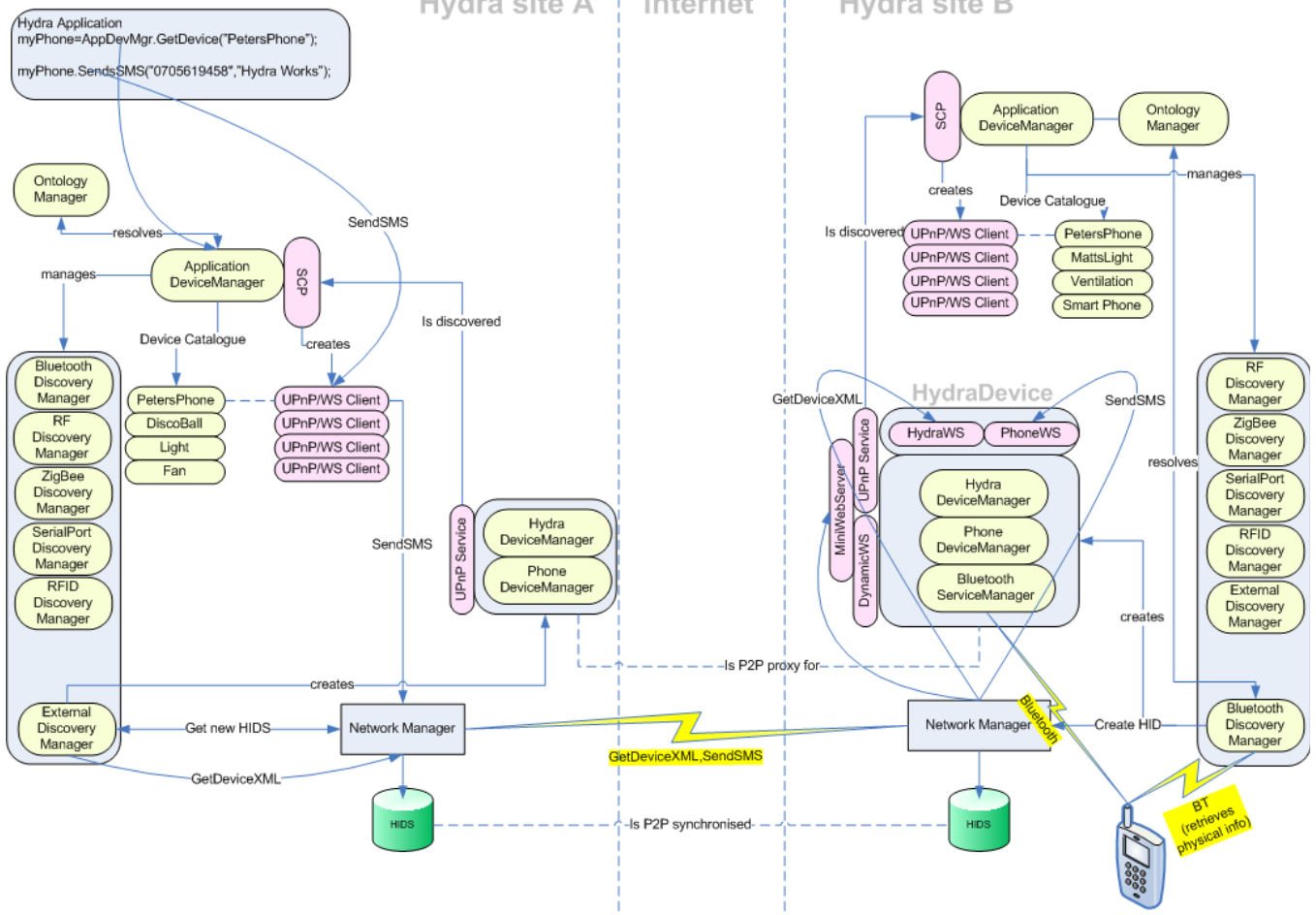
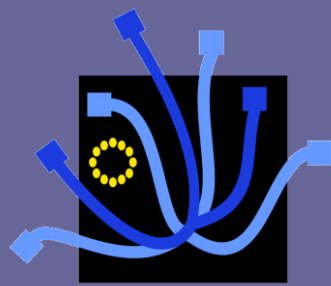


Figure 15 Discovery in the Hydra P2P architecture

- The Discovery Manager then dispatches the Hydra Device and uses the Network Manager to create a Hydra identifier, HID, which is registered with the Network Manager.
- The Hydra Device uses an UPnP broadcast message to announce itself in the local network. The Hydra Device is discovered by the Application Device Manager, who updates the Device Application Catalogue.
- The Hydra Device is now fully functional and available for applications and other devices in the local network on site B.
- The Network Managers in site B and in site A are using P2P techniques to synchronize their own databases of Hydra identifiers.
- An External Discovery Manager is running on site A and will discover that a new device has appeared on site B. It uses the SOAP tunnelling mechanism of the Network Manager to query about the device description of the remote device. Network Manager B receives this request and resolves it using its internal HID database. It results in a local web service call being made to the Phone's generic Hydra Web Service.
- The result of the WS call is then returned to the External Discovery Manager at site A, which now has enough information to create a local proxy for the Phone on site B.
- In the same way this local Hydra Device is discovered in the local network at site A and registered in the DAC.
- One thing that now needs to be done is to bind the Phone to a local application identifier. Applications running in site A need to be able to refer to the devices they need to use, without knowing their physical address or IP-address. These bindings are set up by the Application Device Manager through a rule set provided by the application developers.
- In this case the Phone is now bound to the local identifier "Peters Phone" and an application on site A can invoke services on the phone referring to it as "Peters Phone". When the call SendSMS comes from site A it is routed using P2P, SOAP tunnelling and local web service invocation on site B.



3

Project Impact

The tangible outcomes of the Hydra project are:

- The Software Development Kit (SDK)
- The Device Development Kit (DDK)
- The Integrated Development Environment (IDE)
- The LinkSmart Open Source Middleware
- The LinkSmart training package

WP8: System Integration: Device and Application Development Tools

The outcome of the Hydra project is an Integrated Development Environment (IDE) composed of two parts: (i) a set of tools and class libraries for application developers, called the Software Development Kit (SDK), and (ii) tools intended to facilitate for device developers to make their devices Hydra compliant, the Device Development Kit (DDK). The IDE integrates tools on two main platforms, Eclipse and .Net.

Whereas the SDK is focused on the development of applications of devices, the purpose of DDK is to adapt various physical devices for use by application developers. Many elements of the platform are of course common to both the SDK and the DDK.

As the Hydra name cannot be used for the middleware after the end of the project due to trademark rights, the registered commercial name for the Hydra middleware, LinkSmart™, will be used. The project name of Hydra has also been replaced in all parts of the software's source code with the name LinkSmart.

The LinkSmart Software Development Kit (SDK)

In practice the functions and tools of the SDK support the application developer in the following tasks:

- Finding available devices in the network and initiate discovery.
- Browsing the device ontology.
- Learning about the capabilities of devices, including their energy profiles, through the DAC service listings and in the Device Ontology.
- Initiating and adding devices to the DAC.
- Defining an energy policy for the application.
- Writing application logics based on the services of devices in the DAC.

The main functions of the SDK are implemented by

- A set of Application Templates
- The SDK Class library
- XML schemas for specific vocabularies

The main tools are the DAC Browser and the Ontology Manager, managing the Device Application Catalogue and the Device Ontology respectively. Each manager provides a graphical user interface as well as web service API.

The LinkSmart Device Development Kit (DDK)

LinkSmart Service Compiler (Limbo)

Web services are increasingly adopted as a service provision mechanism in pervasive computing environments. Implementing web services on networked, embedded devices raises a number of challenges, for example efficiency of web services, handling of variability and dependencies of hardware and software platforms, and of device state and context changes.

To address these challenges, the LinkSmart Service Compiler has been developed. Web Ontology Language (OWL) ontologies are used to make the Service Compiler aware of its compilation context, such as targeted hardware and software. At the same time, knowledge of device details, platform dependencies, resource/power consumption etc. is built into the supporting ontologies, which are used to configure the Service Compiler for generating resource efficient web service code.

Device Creation

The main tools used when creating device code in LinkSmart are:

- Intel Service Author for UPnP Technologies
- LinkSmart .Net DDK tool
- The Ontology Manager

The Service Author is used for creating the service methods and producing an SCPD that will be used as input for the final code generation. The actual code generation is done in the LinkSmart .Net DDK tool. It is also where the actual configuration of device type and other settings are done. In this process the Device Ontology is used to select the device type of the device using the ontology class browser. A complete

Visual Studio project is created with the necessary LinkSmart references and the device template is added to the ontology. In the ontology manager tool we can see the information that was added to the ontology by the DDK for the device template.

The LinkSmart Integrated Development Environment (IDE)

The final outcome of the project is the Integrated Development Environment (IDE) which provides an integrated entity with functions and tools from the SDK and DDK, complemented by additional functions and interfaces of the IDE. The IDE integrates tools on two main platforms, Eclipse and .Net.

The LinkSmart Open Source Middleware

LinkSmart has thus been published under the well-recognized and respected Lesser GNU Public License (LGPL). This license differs from the infectious GNU Public License (GPL) in that the software may be used as a library even in commercial software. LinkSmart itself, however, has to remain open and be free of non-free software. Changes to the LinkSmart middleware must be made public themselves. All source files have been updated with a license notice. The LinkSmart code base has been cleaned up, removing software libraries under licenses incompatible with the LGPL.

In the beginning, LinkSmart will be hosted and distributed through renowned Sourceforge.net. Hosting is free but commercial use is prohibited. The website <http://linksmart.sourceforge.net/> has been created for this purpose.

Releasing the Hydra project's software code base as a promising open source project involves more than putting a Zip file with the sources on a publicly accessible server on the Internet. Although such solution could be understood as fulfilment of the project's open source delivery obligation, it would not be in the sense of the European Commission or the research consortium, who wants to contribute significantly to the state of the art. A trivial way of publishing the software, without support or an infrastructure that embraces future change

and bug fixing, would mean that the code is "dead on arrival", useless to software developers and our fellow researchers. For a convincing release, however, there is a multiplicity of different problems and competing goals that must be brought in line.

First, one important aspect was to eliminate the legal conflicts that can arise from using certain third-party software inside LinkSmart, be it proprietary software or open source libraries. These third-party licenses can also restrict the licenses under which LinkSmart can be published. Second, LinkSmart's software license should permit future and other current EU projects to freely use the software. Next, there is a legal risk associated with damages caused by malfunctions and failures of the software. Contributors to the software as well as parties that distribute the software are generally liable for damages. An "as is" clause in the software's license is only partially effective. The publication strategy of LinkSmart should therefore reduce legal risks for research consortium members.

An open source publication typically gives anyone the right to use, change, and redistribute the software on their own terms. However, the consortium partners still have an interest on integrating the software with their businesses. They want to use the software themselves. But also consulting, contracted development, and selling of software licenses are additional ways to exploit project results. The open source license has to be compatible with exploitation plans.

Yet going open source means not only risks but also opportunities to harness the enormous capacities of a skilled open source community that makes contributions free of charge. Also, if other projects that use LinkSmart donate their changes back to the project, large synergies between the projects are unleashed. It is important to attract free developers and other projects, and avoid annoying these supporters, for example by publishing under an Open Source Initiative (OSI) certified and internationally accepted license, by providing high quality code, or by keeping entry barriers to usage low.

The LinkSmart Association

The LinkSmart consortium members have founded a non-profit LinkSmart foundation that supports the future development of the LinkSmart smart, and takes care of the required infrastructure. Until this is legally established, agreements between the partners govern liability issues. A "contributor agreements" has been signed by partners and other contributors to transfer software copyrights fully to the LinkSmart foundation.

The LinkSmart Association's activities shall be driven by the promotion of scientific exchange and information between universities, institutes, enterprises and organisations in the field of networked embedded systems and Internet of Things and the maintenance and further development of the middleware resulting from the Hydra project.

In order to be active in the Association, it is necessary to become a member. Membership is open to all legal entities of public or private law that agree to the Association's objectives, feel committed to those objectives and can credibly demonstrate that they are in a position and willing to contribute to those objectives. Membership carries a membership fee to be determined.

The membership entitles the members to have a preferential status and the original consortium partners will provide various services to the members free or at reduced rates. The LinkSmart Association can be contacted at <http://association.linksmart.eu>

Proprietary results of the project

The Integrated Development Environment (IDE) including the Software Development Kit (SDK) and the Device Development Kit (DDK) contains various proprietary components of the consortium partners, which are not released as Open Source.

Please consult the LinkSmart website for more information on what the project partners can offer in terms of software tools and consultancy services. The website can be found at <http://www.linksmart.eu>.

WP12: Training

The objective of training is to guarantee the long-term impact of the Hydra project, through comprehensive training of the different target audiences. These audiences include consortium members, device & software developers and business development managers.

Throughout the course of the project, training workshops aimed at device and application developers were held. These workshop events aimed at providing 'hands-on' training, with technical documentation, tutorials and practical examples for attendees. Typically, these workshops featured extensive Q&A sessions, where attendees could target the information most important to them. Finally, training sessions for business development managers were held giving a high-level overview of the benefits that the LinkSmart middleware provides. These sessions also presented business modelling challenges, together with solution approaches related to the Internet of Things and Services scenarios enabled by the LinkSmart middleware.

In the final year of the project, six training workshops were successfully held in various European locations including Italy, Sweden and Germany. Some of these workshops contributed to the dissemination of the project, through the training of the partners involved in new projects with members of the project consortium, that are based on, use, or extend the LinkSmart middleware

Furthermore, to support the open source publication of the LinkSmart middleware, an e-Learning website has been created.

Figure 18 shows the created e-Learning platform, which uses Moodle. The e-Learning website provides a central point of access for all learning objects generated over the course of the project, including documents, APIs, presentations, FAQs, screen casts / videos, tutorials and downloadable examples. It provides a personalised learning experience based on the perspective of the user.

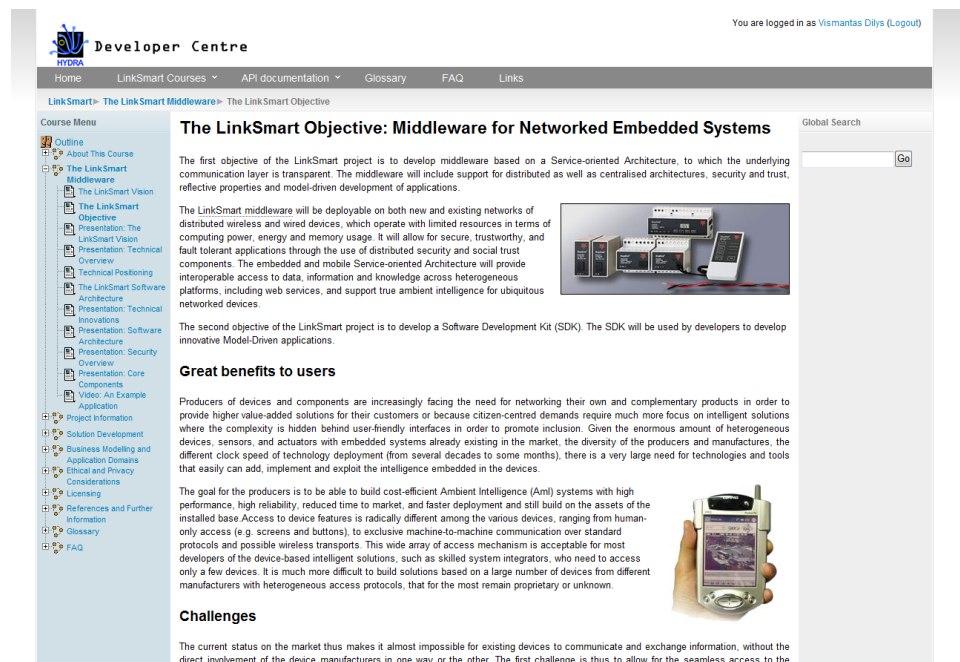
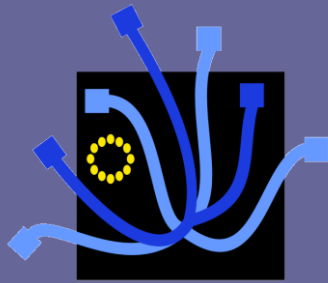


Figure 18: Hydra e-Learning platform

The aggregated training materials cater for multiple audiences, from business managers to device / software developer who wish to use the middleware, but also for open source developers that would want to learn and develop it further. For example, Business Managers are presented with the information most critical to them to evaluate the LinkSmart middleware, whereas device / software developers are provided with in-depth technical learning and training material.



4

Horizontal Activities

In addition to the research and development activities, the project carried out various horizontal activities:

- User Applications
- Business Models
- Dissemination
- Project Management

WP9: User Applications

Three specific user application market segments (domains) have been identified and investigated. The definition of market segments has been important to focus the project's activities on segments which seem to be more attractive than others and which have a deep interest in the project's results.

Building Automation: The demand for innovative devices wired and wireless, mobile and stationary is boundless. Fast growing technology areas include energy efficiency, building automation, smart homes, media capture and playback and communications. These areas will continue to be among the highest growing areas due to new standards and formats and smaller, more capable devices. Software content will increase as more features are packed into smaller devices and new needs for energy and environmental controls will emerge.

Healthcare: The market for telemedicine equipment and application is large and fast growing, driven by a rapidly aging population in the developed world and the need to manage their healthcare efficiently. Telemonitoring can support patients and health professionals. Its use can allow symptoms and abnormal health parameters to be detected earlier than during a routine or emergency consultation, and corrective measures thus to be taken before more serious complications appear. Telemonitoring has already been acknowledged as valuable tools in disease management in several clinical areas.

Agriculture: Embedded Systems in the agricultural environment usually have communication abilities that guarantee the possibility of remote control, supervision and management of physical data (e.g. meteorological parameters) and operational procedures (e.g. the use of fertilisers and other chemicals to preserve farming products). The growing need to manage the environmental impact of agriculture – e.g. minimising the use of pesticides, controlling the use of irrigation – and the public demands of traceability will result in an expansion in the use of ICT in agriculture.

WP 10: Validation & business framework

Business Modelling

An integral part of the Hydra project was the analysis and development of realistic business model for developer users and service providers. New research into defining and measuring value creation in dynamic constellations based on LinkSmart middleware was needed. This should lead to innovative business structures involving content providers, service providers, device manufacturers, and system integrators in collaborative efforts. The goal was to provide the business models, which can be used for customers and users of the LinkSmart middleware, and to instantiate them in realistic, sustainable business cases in the selected domain.

As the name indicates, value modelling focuses on value creation; how value is created, by whom and for whom. It is thus foremost a strategic tool with the aim of identifying new business opportunities and how the firm can position itself strategically to derive maximum benefits from new and emerging opportunities. Value modelling is thus very suitable for engineering radical strategic changes including new product strategies and organisational infrastructures. However, it does not provide much help in defining the most optimal business process implementation.

Value modelling was found to be very useful for analysing and describing the strategic intent of actors in the business environment, because it focus on the value propositions, the value of the offerings to other actors, and thus lays open the strategic foundation for business decisions. Converting a business model into a real business case would use the process model approach in order to identify how processes should be carried out and by whom and thus leading the way to the establishment of realistic revenue models.

Business modelling was performed in each domain: Building Automation, Healthcare and Agriculture.

In the building automation domain, the idea of installing LinkSmart middleware into home automation infrastructure was introduced. Through various discussions and workshops with external experts the focus within the building automation domain was adjusted to the topic of intelligent energy management in a home automation setting. The integration of so called smart meters into home automation infrastructures interconnected and enabled by the LinkSmart middleware is seen as a big opportunity to optimize energy consumption on device level and to gain more energy efficiency in buildings.



In Healthcare, the application areas are almost unlimited and specific implementations will be determined by actual customer requirements at the time of deployment. The business modelling work successfully focused on telemonitoring using the LinkSmart middleware and created sustainable business models for a monitoring platform that connects sensors and devices in the patients' sphere with healthcare professionals and informal carers as well as emergency and crisis management teams in the carers' sphere.

The future of LinkSmart in the Agriculture domain is in the interoperability assured from the middleware in order to apply effectively the remote sensing control of the farm appliances and monitoring devices. In the context of a future typical crop or livestock management application, the network shall be connected with the LinkSmart middleware that integrates the bi-directional gateways, the information flow and the sensor networks. The application is able to provide an instant overview of the farm situation and retrieve and analyse data and elaborate complex algorithms offering support decision tools.



Validation Process

The project's validation objective was to show the effectiveness of the middleware and prototypes from the point of view of a software developer. The aim was that the developer should be able to easily deploy applications based on the middleware.

A validation framework was developed early in the project to serve as a baseline for how, when and by whom validation was going to take place and a validation plan was developed for each prototype. The validation methodology consisted in the verification of to what extent the fit criterion for each requirement had reached the threshold level, or whether the requirement had been partly met or had not been met.

105 requirements out of the total of 476 requirements were assessed in the three validation cycles (assessment started in the second year). The overall validation showed that 95% of the selected requirements had been fully implemented and 5% had been partly implemented.

WP13: Dissemination and Exploitation

Dissemination

To reach a high awareness among different target audiences and to promote the commercial exploitation of project results it was essential for the Hydra project consortium to follow a clearly predefined dissemination and exploitation strategy.

The dissemination aim was to achieve high quality results and high publicity among the scientific community by concentrating on first class conferences and renowned journals. Despite the focus of dissemination was shifted from quantity to quality objectives in the course of the project a large number of conference papers, reviewed articles and a book chapter were published. Also contributions to various roadmaps and position papers as well as teaching material (both written and on-line) were issued.

At the time of writing the official Hydra project website is having more than 2.000 unique visitors per month and more than

450.000 page hits since the start of the project. The 5 most popular papers in the download section have been downloaded altogether more than 11.500 times.

Furthermore, a newsletter has been published every six months during the project. Each newsletter has been sent to more than 400 commercial and scientific contacts. The newsletters are also available for download from the Hydra project website. More than 3.600 downloads have been registered in total for the newsletters.

The various demonstrator showcases and prototypes of the LinkSmart middleware that have been developed in the course of the project show clear and relevant real-world use cases in three application domains. The prototypes have been presented regularly to business related audiences at several high-class fairs like CeBit, GSMA Mobile Summit, as well as high class conferences, open days and customer business workshops.

The project has been mentioned in press reports all over the world including China, Australia, USA and Canada. The Hydra project was also mentioned on television. The German TV channel ZDF has featured a Hydra report on its show "Drehscheibe".

The project also received medals and honorary statements at the CeBIT fair and at the large ICT2008 conference in Lyon.

All in all, the Hydra project was very successful in dissemination and a lot of positive feedback was achieved from the audience at several high class events. All of the predefined dissemination targets could be met very successfully.

Exploitation

With the agriculture, building automation and healthcare sector three domains were identified as demonstration domains.

Several exploitation opportunities have been identified in the three domains, e.g. building automation, healthcare, and agriculture. The project partner CNet has introduced a commercial application called "EnergyInBalance" which allows users to monitor the energy consumption of individual devices and appliances in their home. In the healthcare domain, In-JeT has launched LinkWatch™, a commercial

telemedicine platform for patient monitoring and feed back. There are currently several applications for mobile phones under development in order to remotely monitor health data from a body sensor network of a person. TNM, an IT service provider from Denmark, plans to integrate the LinkSmart middleware in a farm management solution.

Clustering

In terms of policy framework, the project has delivered a comprehensive research roadmap for the Internet of Things and Services and is directly impacting the present EU RTD work programs.

The impact can also be seen in the number of new research projects that have been accepted for funding by the EU, including two large Integrated Projects (REACTION and ebbits), several STREPS, Artemis projects (BEMO-COFRA, Bridge, SeemPubs) and CIP-PSP projects (InCASA). All projects are proposed by partners in the Hydra consortium and are based on the LinkSmart middleware. All proposals were ranked as number one or two at the evaluation.

Several other projects related to the LinkSmart middleware have been approved for funding with other partners.

References

[1] Work Programme 2005-2006, European Commission, 2004

[2] Rozanski, N. and Woods, E. (2005): Software systems architecture: working with stakeholders using viewpoints and perspectives, Pearson Education

[3] OASIS Reference Model for Service-Oriented Architecture, <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html>

[4] Web Service Architecture (WSA) specified by the W3C Working Group, <http://www.w3.org/TR/ws-arch/>

[3] Moodle (<http://moodle.org/>) is an Open Source Learning Management System (LMS)

WP1: Project Management

The Hydra project was managed by a Project Manager elected from one of the partners. After the end of the first year, it was decided to transfer the management of the project to partner Fraunhofer FIT. FIT's successful project management activities ensured that the project properly coordinated its multi-party, multi-disciplinary approach and that the work was completed within the terms of the contract with the European Commission.

A Project Management Board, consisting of one representative member from each partner, was vested with the executive authority for the overall management and running of the project, and the resolution of any major problems that arose.


The Project Manager was assisted in the management task by a Technical Manager appointed from partner CNet and the chairman of the project's Technical Board, from In-JeT.

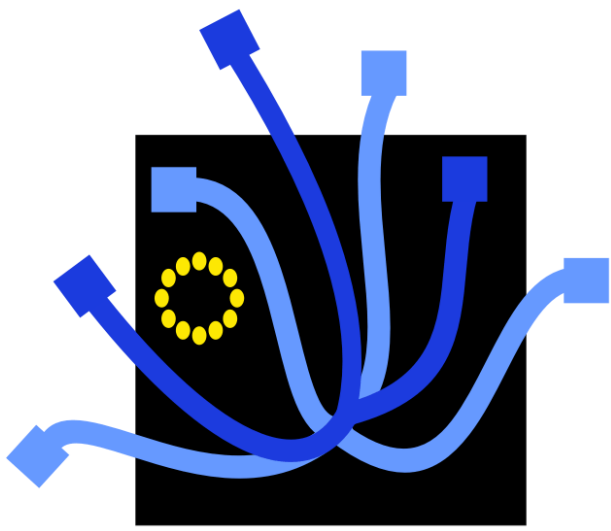
The management structure ensured that the technical and quality requirements of the project were fully implemented, that resources are deployed in an optimum way

An industrial advisory board was established with members from Nokia, Motorola, Siemens, and TNM. All members showed great interest in the project results and contributed positively to the identification of exploitation routes.

In the final year of the project it was decided and agreed to prolong the duration of the project from 48 months to 54 months to allow for the finalization and an in-dept quality check of the middleware in preparation for the following exploitation phase as LinkSmart.

Hydra project partners

	Fraunhofer Institute for Applied Information Technology	DE
	The "Information in Context Group" (ICON) has an acknowledged reputation in the areas of context-awareness, adaptive systems, context modelling, mobile services. www.fit.fraunhofer.de	
	CNet Svenska AB	SE
	CNet is a software house specialising in semantic-based knowledge and content systems with semantic interoperability between heterogeneous services. www.cnet.se	
	Fraunhofer Institute for Secure Information Technology	DE
	SIT is one of the pioneers within the field of IT-Security in Germany and Europe and has experience in development and promotion of security technologies. www.sit.fraunhofer.de	
	In-JeT ApS	DK
	In-JeT ApS is a concept developer and system integrator in Pervasive Computing and has extensive knowledge in concepts, user needs and business model creation. www.in-jet.dk	
	T-Connect s.r.l	IT
	T-connect is engaged in research and development of wireless applications on third generation platforms (UMTS/WLAN) for mobile communications services. www.t-connect.it	
	Telefónica I+D	ES
	Telefónica I+D was formed in 1988 to contribute to the technological innovation of its parent company, by performing research and development activities. www.tid.es	
	University of Aarhus	DK
	The object-oriented software systems group has experience in programming languages, software architecture, and software development tools. www.daimi.au.dk	
	Innova S.p.A.	IT
	Innova S.p.A. is a private company specialised in Technology Transfer services www.innova-eu.net	
	University of Reading	UK
	The Intelligent Media Systems and Services Research Laboratory provides a centre of gravity for research within Systems Engineering and Information Technology. www.reading.ac.uk	
	Siemens IT Solutions and Services	DE
	Siemens IT Solutions and Services (SIS) is one of the world's leading providers of solutions and services in the area of electronic and mobile business. www.c-lab.de	
	Technical University of Kosice	SK
	Technical University of Kosice is the second largest technical university in Slovakia with about 13,000 BSc and MSc students and about 650 PhD students. www.tuke.sk	
	University of Paderborn	DE
	The University of Paderborn has a strong background in computer science, information systems and computer applications. www.uni-paderborn.de	



LinkSmart

LinkSmart[®] Middleware

Effective development of applications
in Networked Systems based on a
Service oriented Architecture

The original project was co-funded by the European Commission within the Sixth Framework Programme

- Contact us at webmaster@linksmart.eu
- For more information go to: <http://www.linksmart.eu>
- Download the Open Source LinkSmart software from <http://linksmart.sourceforge.net>



SIXTH FRAMEWORK
PROGRAMME



European Commission
Information Society and Media