Applications of Semantic Technologies in AmI

Peter Kostelnik, Technical University of Košice, Faculty of Economics
Nemcovej 32, 042 00 Košice, Slovakia, e-mail: peter.kostelnik@tuke.sk
Tomas Sabol, Technical University of Košice, Faculty of Economics
Nemcovej 32, 042 00 Košice, Slovakia, e-mail: tomas.sabol@tuke.sk
Marian Mach, Faculty of Electrical Engineering and Informatics
Letna 9, 042 00 Košice, Slovakia, e-mail: marian.mach@tuke.sk

Abstract: Building and running AmI applications suffers from inflexible and/or difficult to achieve interoperability of basic technological elements. Semantic technologies represent a promising approach how to enable a flexible architectures providing wide range of interoperability (e.g. service-service, service-device, service-goal, etc.). The focus of the presented paper is on incorporating these technologies into the model driven architecture approach to designing and running AmI applications. Employment of semantics is illustrated for design-time as well as run-time. Transition from physical to semantic devices enables developers to work on a higher level when designing applications. Enabling semantics in run-time provides possibility for flexible discovering and resolving.

Key words: Ambient intelligence, Semantic technologies, Ontology, Semantic middleware, HYDRA project.

Introduction

Ambient Intelligence (AmI) represents a vision that technology will become invisible, embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context and autonomously acting [HYDRA D2.2, 2007]. In order to fulfil all these expectations, AmI must amalgamate results from several research areas, for example ubiquitous or pervasive computing (networking of numerous portable low cost devices), intelligent systems (learnable autonomous intelligence), human computer interfaces (multimodal ambient communication with humans), context awareness (reacting according to situational context), etc.

This vision can be realised only through a number of technologies applying modern hardware and software solutions. In particular, an AmI system requires the use of distributed devices and services c to create a pervasive technological layer, able to interact transparently with a user. A key factor for successful deployment of this technological layer is to ensure interoperability of all its elements. The interoperability is important both during runtime as well as at the design phase and affects end users as well as developers of AmI applications.

AmI aims at providing transparent and intelligent electronic services. Generally, these services can be numerous, diverse, and distributed in users' environment. In order to provide services appropriately, several issues have to be solved, for instance the discovery of services (goal and context oriented), invocation of services (context based parametrisation and data transformation), creating, composition and orchestration of services (mediation of their functional as well as non-functional characteristics), etc. Moreover, providing a service depends on devices which can be used to enable the service. It is prone to similar issues on a lower level – discovery, employment and composition of devices.

The problem can be illustrated on service discovery. Most of the existing service discovery mechanisms retrieve services based on whether their descriptions contain particular keywords. In the majority of the cases this leads to low recall and low precision of the retrieved results. The reason of the low recall is that query keywords might be semantically similar but syntactically different from the terms in service descriptions, e.g. 'buy' and 'purchase' (synonyms). The reason for the low precision is that the query keywords might be syntactically equivalent but semantically different from the terms in the service description, e.g. 'order' in the sense of a proper arrangement and 'order' in the sense of a commercial document used to request supply of something (homonyms). Another problem with keyword-based service discovery approaches is that they do not consider the existing relations among keywords – they are not able to perform 'approximate' search in addition to 'exact' search.

The reason is that keyword-based methods can completely capture neither the semantics of user's query nor characteristics (functional as well as quality-based) of services. One possible solution for this problem is to use an ontology-based retrieval. In this approach, semantic ontological models are used for modelling characteristics of services and classification of the services based on their properties. This enables incorporate semantic relations among terms used to describe service properties. Considering the semantics in the query-service matching process can improve the quality of the retrieved results.

In this paper we summarise research on the employment of semantic technologies in AmI applications and present our experiences with these technologies within the Hydra project ("Networked Embedded System middleware for heterogeneous physical devices in a distributed architecture") funded under the 6th Framework Programme of the EU.

2. European R&D Projects in Ambient Intelligence

Before providing more detailed information on the R&D project HYDRA, on the solution of which the authors of this paper participate, this section provides an overview of European research and development projects related to ambient intelligence [HYDRA D13.9, 2008]. The aim of this project overview is to illustrate main research issues and ambient intelligence application areas.

2.1 Smart Networks

Ambient Networks Phase 2

Website:	http://www.ambient-networks.org
Duration:	2006 - 2007 (24 months)
Funding:	FP6, IST

The project addresses vision of the future wireless world, which will be filled by a multitude of user devices, and wireless technologies with simple-to-use, anytime-anywhere network access affordable for everyone. The Ambient Networks project addressed these challenges by developing innovative mobile network solutions for increased competition and cooperation in an environment with a multitude of access technologies, network operators and business actors. It offers a complete, coherent wireless network solution based on dynamic composition of networks that provide access to any network through the instant establishment of inter-network agreements. The technical objectives of the project are: to define and validate a complete and coherent solution for ambient networking, based on a range of different scenarios and business cases; to set new standards based on the Ambient Networks solution for future context-aware; multi-domain mobile networks; to ensure the commercial viability by identifying business roles and interfaces as well as deployment concepts and to consider business scenarios that allow different size and types of players to compete and cooperate.

Smart Embedded Network of Sensing Entities (SENSE)

Website:	http://www.sense-ist.org/
Duration:	2006 - 2009
Funding:	FP6, IST

The SENSE project will develop methods, tools and a test platform for the design, implementation and operation of smart adaptive wireless networks of embedded sensing components. The network is an ambient intelligent system, which adapts to its environment, creates ad-hoc networks of heterogeneous components, and delivers reliable information to its component sensors and the user. The sensors cooperate to build and maintain a coherent global view from local information. Newly added nodes automatically calibrate themselves to the environment, and share knowledge with neighbours. The network is scalable due to local information processing and sharing, and self-organizes based on the physical placement of nodes

2.2 Intelligent Home Environment

Ambient intelligence for the networked home environment (AMIGO)

Website:	http://www.hitech-projects.com/euprojects/amigo/index.htm
Duration:	2004-2008
Funding:	FP6, IST

The AMIGO project aims at development of open, standardized, interoperable middleware and attractive user services, thus improving end-user usability and attractiveness. The project will show the end-user usability and attractiveness of such a home system by creating and demonstrating prototype applications improving everyday life, addressing all vital user aspects: home care and safety, home information and entertainment, and extension of the home environment by means of ambience sharing for advanced personal communication. The project will further support interoperability between equipment and services within the networked home environment by using standard technology when possible and by making the basic middleware (components and infrastructure) and basic user services available as open source software together with architectural rules for everyone to use. Context-awareness concepts and principles are at the heart of the AMIGO project.

2.3 Ambient Assisted Living

Complete Ambient Assisted Living Experiment (CAALYX)

Website:	http://caalyx.eu
Duration:	2007-2009 (24 months)
Funding:	FP6, IST

Ambient Assisted Living (AAL), as a specific user-oriented type of "Ambient Intelligence", aims to prolong the time people can live in a decent more independent way by increasing their autonomy and self-confidence, by allowing them to discharge normal everyday activities, by improved monitoring and care of the elderly or ill person, by enhancing their security while ultimately saving resources. CALALYX’s main objective is to develop a wearable light device able to measure specific vital signs of the older person, to detect falls and to communicate automatically in real time with his/her caregiver in case of an emergency, wherever the older person happens to be, at home or outside.

Old people's e-services at home (OLDES)

Website:	http://www.oldes.eu
Duration:	2007-2010
Funding:	FP6, IST

The OLDES project will offer new technological solutions to improve the quality of life of older people. OLDES aims at developing a very low cost and easy to use entertainment and health care platform designed to ease the life of older people in their homes. In order to achieve this, new concepts developed in Information Technologies will be integrated and adapted. OLDES will provide: user entertainment services, through easy-to-access thematic channels and special interest forums supported by animators; and health care facilities based on established Internet and tele-care communication standards. The system will include wireless ambient and medical sensors linked via a contact centre to social services and health care providers. OLDES will also cover the definition, implementation and evaluation of a Knowledge Management (KM) program, an advanced user profiling system that will enhance the communication between all the stakeholders of the system. The system will be tested at two different locations: Italy over a group of 100 elderly (including 10 suffering with cardio disease) and Czech Republic over a group of 10 diabetic patients. OLDES puts older people at the centre and makes their needs the main priority in all developments.

Other R&D projects on ambient assisted living:

- Perceptive Spaces Promoting Independent Ageing (PERSONA), http://www.aal-persona.org/
- Mainstreaming on ambient intelligence (MonAMI), http://www.monami.info/
- An intelligent interactive services environment for assisted living at home (INHOME), http://www.ist-inhome.eu/

2.4 Security and Privacy

Privacy in an Ambient World (PAW)

Website:	http://www.cs.ru.nl/paw
Duration:	2003-2007
Funding:	Dutch Ministry of Economic Affairs, IOP GenCom programme

The objective of the PAW project was to develop a privacy protecting architecture that can provide full privacy of the user in an ambient world. A two-tier approach was applied to prevent unwanted collection of data about a user and his actions. Techniques from secure computing were extended to provide private computing. To control the authorised dissemination of data about a user, licensing techniques similar to those used in digital rights management were studied. Since ambient systems are characterised by their low resources and capabilities, PAW's key challenge was to develop an efficient architecture. An advantage of the architecture proposed is decentralization, where no central authority for controlling is necessary. But unfortunately an agent can use data in an un-authorised way and can be held responsible only after the misuse of the data (which is not an ideal solution for real world scenarios).

Security Expert INITiative (SEINIT)
Website: http://www.isoc.org/seinit/portal/index.php?option=com_content&task=view&id=53&Itemid=28
Duration: 2003-2006
Funding: FP6, IST
The overall objective of the Security Expert INITiative (SEINIT) project is to ensure a trusted and dependable information security framework, ubiquitous, working across multiple devices, heterogeneous networks, being organisation independent (inter-operable) and centred around an end-user. SEINIT was exploring new information security models and building the architecture and components to address our nomadic, pervasive, multi-player communicating world. This new solution uses information gathered by ambient intelligence and then deals with the new threats this entails.

2.5 Non-technical Issues of Ambient Intelligence

Safeguards in a World of Ambient Intelligence (SWAMI)
Website: http://www.ist-world.org/ProjectDetails.aspx?ProjectId=212f8a94875941f88cfa0dc321d60a8f&SourceDatabaseId=7cff9226e582440894200b751bab883f
Duration: 2005-2006
Funding: FP6, IST
The main objective of the SWAMI project is (based also on the review of existing AmI projects, studies, scenarios and roadmaps) to identify the social, legal, organizational and ethical implications related to issues such as privacy, anonymity, security and identity in the context of AmI. This has been achieved through the elaboration of 'dark scenarios' as a centre piece of the SWAMI project methodology. The ‘dark scenarios’ depict a realistic future that could emerge from the application of new AmI technologies, but focus on the likely adverse effects which often are overlooked by technology developers and policymakers. The first objective of the dark scenario exercise thus consisted of the identification of potential threats and vulnerabilities that need to be mitigated if AmI is to become a future success story.

Development of Long-term Shared Vision on AmI Technologies for a Networked Agro-Food Sector (AMI@NetFood)
Website: http://www.ami-netfood.com/
Duration: 2005-2006
Funding: FP6, IST
The objective of AMI@Netfood project is to support the implementation of the IST Research Priority and Framework Programme, providing a long-term vision on future trends on scientific and technology research oriented to the development and application of Ambient Intelligence technologies in the agro-food domain.

3. HYDRA Project

The HYDRA project (“Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture”), contract No. 034891, is an Integrated Project (IP) funded by the EC within FP6, Information Society Technologies (IST) Programme. The project started on July 1st, 2006 and its expected duration is 52 months. The project consortium consists of 13 partners from Sweden, Denmark, Germany, Spain, the United Kingdom, Italy, and Slovakia (2 large companies, 5 SMEs, 6 universities and research institutes).

The HYDRA project is addressing the problem, which is frequently faced by producers of devices and components – the need for networking (which is actually becoming a trend) the products available on the market in order to provide higher value-added solutions for their customers. This requirement is implied by citizen centred demands requiring intelligent solutions, where the complexity is hidden behind user-friendly interfaces. It is expected that HYDRA will contribute to the vision of the ambient intelligence world as described in the introductory section.

Overall HYDRA project objectives can be summarised as follows:
1. Development of a middleware based on a Service-oriented Architecture, to which the underlying communication layer is transparent, and consists of:

- Support for distributed as well as centralised ambient intelligent architectures;
- Support for reflective (i.e. self-*) properties of components of the middleware;
- Support for security and trust enabling components.

2. Design of a generic semantic model-based architecture supporting model-driven development of applications.
3. Development of a toolkit for developers to develop applications on the middleware.
4. Design of a business modelling framework for analysing the business sustainability of the developed applications.

From scientific point of view the project is carrying out research as well as application and system integration within the following research areas:

- Embedded and mobile service-oriented architectures for ubiquitous networked devices;
- Semantic Model-Driven Architecture for Ambient Intelligence implementation;
- Ontology-based knowledge modelling;
- Wireless devices and networks with self-* properties (self-diagnosis, self-configuring, self-healing, etc.);
- Distributed security and privacy.

The implemented HYDRA middleware and toolkit is validated in real end-user scenarios in three different user domains: a) Facility management (home automation), b) Healthcare, c) Agriculture.

3.1 HYDRA approach to Semantic Model-Driven Architecture

One of the main goals of HYDRA project is to provide the middleware solution for creating applications interconnecting various heterogeneous devices with different services and capabilities. HYDRA aims to develop middleware based on a Service-oriented Architecture (SOA) providing the interoperable access to data, information and knowledge across heterogeneous platforms, including web services, and support true ambient intelligence for ubiquitous networked devices. The SOA and its related standards provide interoperability at syntactic level. However, HYDRA aims at providing interoperability at semantic level as well. One of the objectives is to extend this syntactic interoperability to the application level, i.e. in the terms of semantic interoperability. This is done by combining the use of ontologies with semantic web services. HYDRA introduces the Semantic Model Driven Architecture (MDA), which aims to facilitate application development and to promote semantic interoperability for services and devices. The semantic MDA of HYDRA includes a set of models (ontologies) and describes how these can be used both in design-time and in run-time.

The basic idea behind the HYDRA Semantic MDA is to differentiate between the physical (real) devices and the application's view of the device. This approach leads to introduction of the concept of Semantic Devices [Kostelnik, 2008]. In the easiest case the semantic device represents a model of a real device and serves as logical unit, which can be semantically discovered and provide information about device capabilities, services. But from these simple semantic devices one can construct more complex semantic devices consisting from several "logical units" (other semantic devices) plus defining a "logics" between these semantic devices. Thus semantic devices can be viewed as logical aggregates of devices and their services.

The services offered by the physical devices are designed independently of particular applications in which the device might be used. A semantic device on the other hand represents what the particular application would like to have. For instance, when designing a lighting system for a building it would be more appropriate to model the application as working with a "logical lighting system" that provides services like "working light", "presentation light", and "comfort light" rather than working with a set of independent lamps that can be turned on/off. These semantic devices may in fact consist of aggregates of physical devices, and use different devices to deliver the service depending on the situation. The service "Working light" might be achieved during daytime by pulling up the blind (if it is down) and during evening by turning on a lamp (blind and lamp being HYDRA devices).

Semantic devices should be seen as a programming concept. The application programmer designs and programs his/her application using semantic devices. The semantic device "Heating System" consist of three physical devices: a pump that circulates the water, a thermometer that delivers the temperature and a light that flashes when something is wrong. The developer will only have to use the services offered by the semantic device "Heating System", for instances "Keep temperature: 20 degrees of Celsius" and "Set warning level: 17 degrees of Celsius", and does not need to know the underlying implementation of this particular heating system. The

Semantic Device concept is flexible and will support both static mappings as well as dynamic mappings to physical devices.

Static mappings can be both one-to-one (mapping of a semantic device to a physical device) or mappings that allow composition. An example of a one-to-one mapping would be a “semantic pump” that is exposed with all its services to the programmer. An example of a composed mapping is a semantic heating system that is mapped to three different underlying devices – a pump, a thermometer, and a digital lamp.

Static mappings will require knowledge, which devices exist in the runtime environment, for instance the heating system mentioned above will require the existence of the three underlying devices – pump, thermometer and lamp – in for instance a building.

Dynamic mapping will allow semantic devices to be instantiated at runtime. Consider the heating system above. We might define it as consisting of the following devices/services:

- a device that can circulate the water and increase its temperature;
- a device that can measure and deliver temperature;
- a device that can generate an alarm/alert signal if temperature is out of range.

When such a device is entered into the runtime environment it will use service discovery to instantiate itself and it will query the physical devices it discovers, as to which can provide the services/functions the semantic device requires. In this example the semantic device most probably starts by finding a circulation pump. Having done that, it might find two different thermometers, which both claim that they can measure temperature. The semantic device could then query, which of the thermometers can deliver the temperature in Celsius, with what resolution and how often. In this case it might be only one of the thermometers that meets these requirements. Finally, the semantic device could search the network if there is a physical device that can be used to generate an alarm if the temperature drops below a threshold or increases too much. By reasoning over the semantic device it can deduct that by flashing the lamp repeatedly it can generate an alarm signal, so the lamp is included as part of the semantic heating system.

The basic idea behind semantic devices is to hide all the underlying complexity of the mapping to, discovery of and access to physical devices. The programmer just uses it as a normal object in his application, focusing on solving the application’s problems rather then the intrinsic of the physical devices. The description of semantic devices are realised as device OWL [McGuiness, 2004] ontology, which will be outlined in the next chapter. The concept of semantic devices supports the semantic MDA approach. The semantic MDA in HYDRA is used in two ways. Firstly, it is relevant at the design-time, and it will support both device developers as well as application developers. Secondly, at the run-time any HYDRA application is driven from the semantic MDA.

3.1.1 Semantic MDA at design-time

Model-driven code generation for physical devices

Within HYDRA project a tool, called *Limbo*, has been developed, which takes as inputs an interface description (“Provide WSDL file”) and a semantic description of the device on which a web service should run (“Provide OWL description”) [Hansen, 2008]. The interface description is assumed to be in the form of a WSDL file and the semantic description is a link to an OWL description of the device (ontologies used are described in the next chapter). The semantic description is used to:

- *Determine the compilation target.* Depending on available resources of a device, either embedded stubs and skeletons are created for the web service (to run on the target device) or proxy stubs and skeletons are created for the web service (to run on an OSGi gateway).
- *Provide support for reporting device status.* Based on a description of the device states at runtime (through a state machine), support code is generated for reporting state changes. This should be also used as the support of self-* properties of HYDRA.

Model-driven code generation for Semantic Devices

The semantic descriptions of services can be used at the design time to find suitable services for the application that the HYDRA developer is working on. The descriptions of these services will be used to generate code to call the service, query the device that implements the service, and manipulate the data that the service operates on. The project team is currently aiming at making the HYDRA SDK available in an object-oriented language environment. Thus, SDK will provide objects a developer can use to access the services (service proxies) as well as objects from the device ontology connected to the service. HYDRA developer can specify a service to be used, and leave the device as generic as possible - any device that is capable of implementing the service. The necessary code will be generated both for the service and the device.

These device objects could be used when creating a semantic device or HYDRA application from the selected devices and services. The services could also be used by a service orchestration engine (however, presumption that some applications will be standalone and have a fairly small footprint may not be suitable for all HYDRA applications).

The way how the application uses the device ontology should be configurable, so that the middleware supports both standalone applications that only use the device ontology at the design time as well as applications that always query the device ontology for new types of services that match the descriptions.

3.1.2 Semantic MDA in run-time

Models for discovery
At the design time, the HYDRA application developer selects the HYDRA devices and services that will be used to implement the application. These devices may be defined at a fairly general level, e.g. the application may be interested only in "HYDRA SMS Service" and any device to be included into the network (or application context) that fits to these general categories will be presented to the application. The application will then work against more general device descriptions. This means that an application should only know of (types of) devices and services selected by the developer when it was defined. This also means that the application could use a device that was designed and built after the application was deployed - as long as the device can be classified through the device ontology as being of a device type or using a service that is known to the application, e.g., a HYDRA application built in 2008 could specify the use of "HYDRA Generic Smart phone" and "HYDRA SMS Service" and thus use also a "Nokia N2010 Smartphone" released two years later.

When a device is discovered, the device type is looked up in the device ontology and if it can be mapped to a specific device model (perhaps it can always be mapped to a most generic type of devices).

A HYDRA Application may present an external interface so that it can be integrated with other applications and devices. It will do this by identifying itself as a HYDRA device with a set of services. This is transparent to other devices, which means that some devices or services used in the application will be composite ones - based on other HYDRA applications that have exposed external interfaces. When such an application is discovered, the applications interested in that type of device and its services will be notified.

The use of models for resolving security requirements
The dynamic and networked execution environment of HYDRA requires strong yet adaptable security mechanisms to be in place. In order to establish the ability to securely connect any application/device to any application/device, HYDRA also uses the semantic MDA to define and enforce security policies [Wahl, 2008]. A basic design objective for the HYDRA security model is to provide a secure information flow with a minimum of pre-determined assumptions, while being able to dynamically resolve security requirements. The security policies of HYDRA can thus be defined and enforced based upon knowledge in the device ontology as well as on knowledge of the context of devices, and also makes use of virtual devices.

The use of models for context-awareness
To support ambient intelligence applications, the models for context-awareness functionality are also provided. The models are created as a combination of OWL ontologies and SWRL [Horrocks, 2004] rules and serve as the basic mechanism supporting the self-* properties [Zhang, 2008]. At present, the context-awareness models are used in the self-management and self-diagnosis task. The ontologies contain the models of devices and the state machines representing the actual status of devices. The ontology containing possible malfunctions, which may occur on devices is also used. The context is modelled using the SWRL rules, which can be defined on the device level (monitoring and reacting to the state of the single device) and the system/application level (monitoring and reacting to the context created as a combination of multiple devices states). The continual execution of rules may lead to generation of several malfunctions containing the description of error and the related remedies, which are provided to application users.

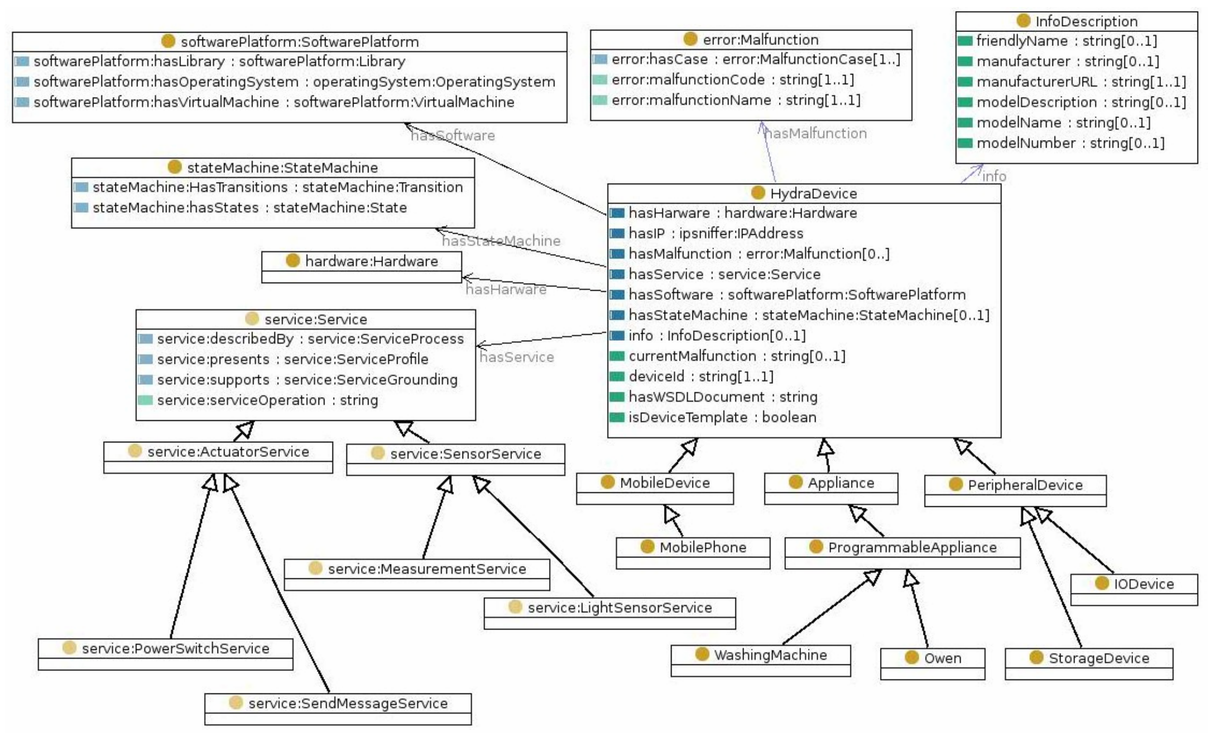

softwarePlatform:SoftwarePlatform
softwarePlatform:hasLibrary : softwarePlatform:Library
softwarePlatform:hasOperatingSystem : operatingSystem:OperatingSystem
softwarePlatform:hasVirtualMachine : softwarePlatform:VirtualMachine
error:Malfunction
error:hasCase : error:MalfunctionCase[1..]
error:malfunctionCode : string[1..1]
error:malfunctionName : string[1..1]
InfoDescription
friendlyName : string[0..1]
manufacturer : string[0..1]
manufacturerURL : string[1..1]
modelDescription : string[0..1]
modelName : string[0..1]
modelNumber : string[0..1]
hasSoftware
hasMalfunction
info
stateMachine:StateMachine
stateMachine:HasTransitions : stateMachine:Transition
stateMachine:hasStates : stateMachine:State
HydraDevice
hasHarware : hardware:Hardware
hasIP : ipsniffer:IPAddress
hasMalfunction : error:Malfunction[0..]
hasService : service:Service
hasSoftware : softwarePlatform:SoftwarePlatform
hasStateMachine : stateMachine:StateMachine[0..1]
info : InfoDescription[0..1]
currentMalfunction : string[0..1]
deviceId : string[1..1]
hasWSDLDocument : string
isDeviceTemplate : boolean
hasStateMachine
hardware:Hardware
hasHarware
service:Service
service:describedBy : service:ServiceProcess
service:presents : service:ServiceProfile
service:supports : service:ServiceGrounding
service:serviceOperation : string
hasService
service:ActuatorService
service:SensorService
MobileDevice
Appliance
PeripheralDevice
MobilePhone
ProgrammableAppliance
IODevice
service:MeasurementService
service:LightSensorService
service:PowerSwitchService
WashingMachine
Owen
StorageDevice
service:SendMessageService

Figure 1. Part of the HYDRA Device Ontology.

3.2 Semantic Description of Devices

To achieve the vision of a Semantic Model Driven Architecture we have decided to base our approach on ontologies and related semantic technologies. HYDRA device ontology presents basic high level concepts describing basic device related information [Kostelnik, 2008]. The semantic device model can be viewed from two perspectives:

- *Design/development phase*: 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 the new device type is needed, the adequate concept in the device classification module can be further sub-classed by more specialized concepts and the new properties can be added. Specific device models are created as the instances of device ontology concepts and filled with real data.
- *Run-time phase*: Each instance created in the development phase represents a specific device model and serves as the template for run-time instances of real devices discovered and connected into HYDRA. In the device discovery process, the discovery information is resolved in ontology and the most suitable template is identified. The identified template is cloned and new unique run-time instance representing the specific device is created. Each real device instance has an assigned unique HYDRA Id. Using this Id, it is possible to retrieve and update all the relevant information related to the general description of the device and its actual run-time properties.

The ontology structure was designed to support the maintainability and future extensions of used concepts. The ontologies have been developed using the OWL language. Figure 1 illustrates a part of HYDRA device ontology.

The Hydra Device Ontology presents the basic high level concepts describing basic device related information, which are used in both the development and run-time process. The core device ontology contains taxonomy of device types and basic device and manufacturer information. The description of the device properties and capabilities is divided into four interconnected modules:

- Device malfunctions ontology represents possible errors that may occur on devices. The malfunctions are divided in the taxonomy of possible error types according to the severity, such as error, warning or fatal. Each error model contains the human readable description and set of cause-remedy pairs. The connection of malfunction model and device state machine is used also for diagnostic purposes. Various faults related to specific ontology states can be, for example, used to predict or avoid fatal error states of device or to invoke related call-back events to handle the error states that may occur in the run-time.

- Device capabilities represent the extended device information mainly used for purposes of generation of embedded device services code and self-* properties. The device capabilities are divided into three modules:
    - Hardware module includes the hardware related device properties such as connection and communication protocols (e.g. Bluetooth or various network bearers, etc.), description of hardware interfaces (such as camera, display, etc.). Device hardware capabilities are mainly used for generation of embedded device services code.
    - Software module includes various software platforms, operating systems, etc. Software description is used by the Limbo compiler used for generating the codes for embedded device services.
    - The special case of capability is the state machine model representing the concepts of states and transitions, which are updated in the run-time and represent the device/service actual status. The state machine ontology is also used by Limbo to generate state and transition related code.
- Device services ontology presents the semantic description of device services on a higher, technology independent level. The HYDRA service model enables the interoperability between devices and services, employing the service capabilities and input/output parameters. The semantic service specification is based on the OWL-S [Martin, 2004] standard, which is currently the most complete description of semantic mark-ups for services following the web service architecture. The OWL-S approach was taken as the starting point for HYDRA service model. The models of services may be automatically created using the SAWSDL annotations [Farrell, 2007] enabling automatic classification of device and its services into ontology. Service models serve as the semantic representation supporting the search of devices providing required functionality and also as the grounding information for service execution.
- Security capabilities ontology represents the security properties of devices and the services, such as protocols, policies, mechanisms or objectives. The NRL ontology [NRL, 2007] was selected as a starting point for this model and has been modified and extended to match HYDRA's requirements. The NRL ontology is a set of various security related models covering the representation of credentials, algorithms, assurances, but also the service security aspects directly supporting the SOA approach.

3.3 HYDRA end-user applications

The HYDRA middleware serves as a tool supporting efficient development of AmI applications in various end-user domains. System developers are thus provided with tools for easy and secure integration of heterogeneous physical devices into interoperable distributed systems.

The middleware includes support for distributed as well as centralised architectures, cognition and context awareness, security and trust and will be deployable on both new and existing networks of distributed wireless and wired devices that typically are resource constrained in terms of computing power, energy and memory. The middleware is validated in three application domains: Building automation, healthcare and agriculture.

The development is realised in an iterative way - four iterations are planned within HYDRA for the middleware development. After each cycle the middleware is progressively becoming more advanced – while learning more about developer as well as end-user requirements. Evaluation of experiences from the previous cycles is taken into account. At the end of each cycle a demonstrator is produced with a specific purpose to illustrate:

- Concept Demonstrator;
- SDK Demonstrator;
- DDK (Device Development Kit) Demonstrator;
- IDE Demonstrator.

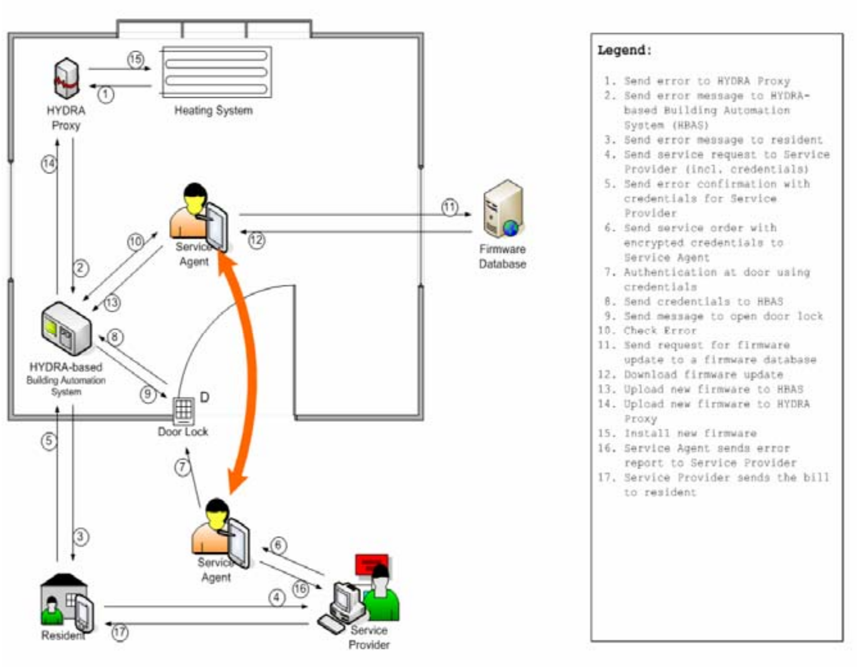
For each cycle more and more of the end-user applications are implemented, so they become more sophisticated as the project progresses. For each cycle there is one user domain in focus but scenarios from the other two is also considered - in cycle (1) Building Automation is the main emphasis, in cycle (2) Healthcare and in cycle (3) Agriculture. Finally in the fourth cycle all domains will be fully demonstrated.

An example of the building automation scenario used for the demonstration of the proof of the concept can be outlined as follows (see Figure 2 for illustration):
The resident is living in a new flat in the "Krøyers Plads" housing complex in Copenhagen. In addition to the usual set of automatic lamps, computer and wireless network, the flat is equipped also with an automatic heating system.

While the resident is at his office, he receives an alert from his "Hydra Building Automation System" (HBAS) that the heating system has broken down. Since the temperature has reached sub zero level, HBAS categorized it as an emergency situation and tries to contact the resident until he replied to the alert.

Since the resident is having a contract with the service provider of the heating system to send out a service agent to repair the system in case of a break down, the resident sends a repair order to the service provider. The service provider sends out a service agent to the flat. The service provider has transferred the appropriate credentials to enter the house and to repair the heating system to the service agent's PDA.

When the service agent arrives at the complex, he authenticates himself to the door and is given access to the resident's flat after the validation was successful. The service agent checks the logs present in the heating system to identify the errors and uses the online help of the service provider to fix the problem. After finishing his work, the service agent leaves the flat and the HBAS system informs the resident that the heating system is working again and the service agent has left the flat.



Legend:

1. Send error to HYDRA Proxy
2. Send error message to HYDRA-based Building Automation System (HBAS)
3. Send error message to resident
4. Send service request to Service Provider (incl. credentials)
5. Send error confirmation with credentials for Service Provider
6. Send service order with encrypted credentials to Service Agent
7. Authentication at door using credentials
8. Send credentials to HBAS
9. Send message to open door lock
10. Check Error
11. Send request for firmware update to a firmware database
12. Download firmware update
13. Upload new firmware to HBAS
14. Upload new firmware to HYDRA Proxy
15. Install new firmware
16. Service Agent sends error report to Service Provider
17. Service Provider sends the bill to resident

Figure 2. Example of home automation scenario.

4. Conclusions

Semantic technologies represent a very promising approach to solving many problems in AmI applications. Coupling of these technologies with the MDA method generates many possibilities for incorporating semantics into the design of applications as well as into process of running AmI applications. Enabling semantics on the level of the description of AmI technological elements (mainly services and/or devices) results in enriched possibilities for matching operation (e.g. service to service, service to goal, device to service, device to device, etc.) not forced to rely only on syntactic similarity any more. Such matching can be found in the background of such essential tasks as discovery, composition, and mediation.

Regarding the future development within the HYDRA project, since the semantic MDA is built on ontologies we foresee a need for tools and methods for design and managing these ontologies. Therefore for the next development iterations the problem how the design and management of the three Hydra ontologies can be carried out efficiently will be investigated. The ontology management addresses mainly the following issues:

- Ontology design process: The initial HYDRA ontology design process is manual, performed by ontology engineering experts. The tools enabling the ontology design and refinement also for not expert users should be provided.

- Ontology extensions: To support the efficient development of applications in various end-user domains, ontologies should be extended also by descriptions representing the end-user domains. Such descriptions should lead to increased capability of semantic interopearability between devices, but also between applications.
- Automatic modification of ontologies: HYDRA should provide more tools for automatic extension of ontologies, mainly in the terms of automatic classification of the new devices into ontology, where the description of new device may be provided in various form, such as MS Word, PDF, HTML or XML. This process would require the development of transformation tools enabling the parsing and processing of this description forms.
- Mediation, aligning and merging of ontologies: A developer should be able to import an external (device) ontology and be provided with tools for its adaptation and use in application development.

There are also several issues to be further investigated regarding the management of the discovery process and management of semantic devices at the run-time. Actually, the whole service composition occurs at the design time. In the following iterations, the Hydra middleware may have to resolve this at the run-time when a set of devices and services that are present in the network constitute a composite device. The semi or fully automatic service composition may improve the system robustness mainly in the case of run-time device failures.

There will be also required various visualisation mechanisms supporting several development tasks, such as searching and retrieving the devices or services by functionality requirements or required capabilities. This functionality may be used in the tasks of semi or fully automatic creation of composite semantic devices and will require, as the addition to the ontology reasoning, also planning capabilities.

Acknowledgements

The work presented in the paper is partly funded by the European Commission through the IST Project HYDRA (Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture), contract No. 034891 as well as by the Slovak Grant Agency of the Ministry of Education and Academy of Science of the Slovak Republic within the 1/4074/07 Project “Methods for annotation, search, creation, and accessing knowledge employing metadata for semantic description of knowledge”.

References:

[HYDRA D2.2, 2007] D2.2 Initial technology Watch report. Hydra Project Deliverable, IST project No. 034891, 2008.

[HYDRA D13.9, 2008] D13.9 Report on Projects Connected to HYDRA. Hydra Project Deliverable, IST project No. 034891, 2008.

[Farrell, 2007] J. Farrell, et.al. Semantic Annotations for WSDL and XML Schema, W3C Recommendation, 2007.

[Hansen, 2008] K.M. Hansen, W. Zhang, G. Soares, Ontology-Enabled Generation of Embedded Web Services. The 20th International Conference on Software Engineering and Knowledge Engineering (SEKE '2008), 2008.

[Horrocks, 2004] I. Horrocks, et.al. SWRL: A Semantic Web Rule Language. W3C Member Submission, 2004.

[Kostelnik, 2008] P. Kostelnik, M. Sarnovsky, J. Hreňo, M. Ahlsen, P. Rosengren, P. Kool, M. Axling. Semantic Devices for Ambien Environment Middleware. EURO TrustAMI, Internet of Things and Services Workshop, 2008.

[Martin, 2004] D. Martin, et. al. OWL-S: Semantic Markup for Web Services, W3C Member Submission, 2004.

[McGuinness, 2004] D.L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview, W3C Recommendation, 2004.

[NRL, 2007] Naval Research Lab. Nrl security ontology. http://chacs.nrl.navy.mil/projects/4SEA/ontology.html, 2007.

[Wahl, 2008] T. Wahl, J. Schütte, P. Kostelnik, Security Mechanisms for an Ambient Environment Middleware. ServiceWave, 2008.

[Zhang, 2008] W. Zhang, K.M. Hansen, Towards Self-managed Pervasive Middleware using OWL/SWRL ontologies. Fifth International Workshop Modeling and Reasoning in Context (MRC 2008), 2008.