The use of ontologies in the HYDRA middleware

Peter Kostelnik\textsuperscript{a}, Ján Hreňo\textsuperscript{a}, Martin Sarnovský\textsuperscript{a}, Matts Ahlsen\textsuperscript{b}, Peter Rosengren\textsuperscript{b}, Peeter Kool\textsuperscript{b} and Mathias Axling\textsuperscript{b}
\textsuperscript{a}Technical University of Košice, Letná 9, Košice 04001, Slovak Republic
\textsuperscript{b}CNet Svenska AB, Svärdvägen 3B, 182 33 Dandaryd, Sweden

Abstract. The goal of this paper is to provide an informative overview of Semantic Web technologies used in the HYDRA project ("HYDRA: Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture") funded within the FP6 IST Programme (IST-2005-034891). The concept of Semantic Model-Driven Architecture is introduced and the main motivations of using the Semantic Web technologies are stated. Basic scenarios of semantic technologies usage are outlined and a brief overview of the designed ontologies is presented.

Keywords: semantic modelling, model driven architecture, usage scenarios

1. Introduction

In the ambient world of the near future, interconnected intelligent devices will be surrounding us at home, at work or while travelling. These devices and their local networks will be connected to the outside world through broadband or wireless networks. Numerous services to support users in their personal life will be provided through these ambient devices. In order to cope with the huge variety of capabilities of these devices, there should be a mechanism providing the necessary adaptations to whatever interfaces or communication protocols these devices offer. To achieve this, the capabilities of the devices must be described in such way that an automatic agent can understand and use them.

The HYDRA project aims to research, develop, and validate middleware for networked embedded systems that allows developers to create cost-effective, high-performance ambient intelligence applications for heterogeneous physical devices. There are three domains for demonstration applications created:
- facility management (smart homes),
- healthcare, and
- agriculture.

One of the major goals of the project is to provide a middleware solution for interconnecting a large number of various heterogeneous devices with different services and capabilities. The HYDRA middleware will be deployable on both new and existing networks of distributed wireless and wired devices, which operate with limited resources in terms of computing power, energy and memory usage. HYDRA aims to develop a 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 applications for ubiquitous networked devices. The SOA and its related standards provide interoperability at a syntactic level. However, HYDRA also aims to provide interoperability at a semantic level. One of the objectives is to extend the syntactic interoperability to the application level in terms of semantic interoperability. This is done by combining the use of ontologies with semantic web services. HYDRA introduces the Semantic Model Driven Architecture (SeMDA) which aims to facilitate application development and to promote semantic interoperability for services and devices. The SeMDA of HYDRA includes a set of ontologies, and provides a set of tools, which can be used both in application design time and in runtime.
2. Motivation

The variety of applications using various devices providing several services is really extensive. It is due to the large set of possible views on how devices have to or should be used, and what the application needs to know about devices to enable the needed functionality. Sometimes it is important to know, which device saves more energy than another; which device has a larger display to play video; which device is capable to measure the temperature in degrees of Celsius or which devices with a certain required functionality are located in the living room. As the middleware serves as a development tool for creating such applications, it has to provide a uniform access to the needed information on the devices, their services, and sometimes also about the application itself.

The main reason for using semantic technologies in general was to enable a flexible and easily scalable representation of the domain describing the applications containing various devices and services. As the application scenarios have been continually evolving during the project life and the functional requirements were changing in a very different ways, it was absolutely not possible to assume, what entities the models should describe, how they should be related or what questions the model should be able to answer. In this sense, the use of ontologies appeared as the best candidate technology promising very flexible modelling and querying capabilities. On the other hand, in the domain of interest, there are already a lot of existing ontologies, so there was a strong assumption that some of them could be easily reused or integrated into the model using annotations.

Furthermore, in the domain of applications using the various devices, many times it is required to retrieve an answer to questions, which cannot be formulated very precisely, as the system does not contain the required information represented explicitly. In such cases, the answer has to be inferred (for example: “What mobile devices owned by Peter are capable of displaying the AVI video?”). This is another good reason to use a technology, which has a built-in efficient inference mechanism.

3. Using the ontologies in HYDRA

SeMDA provides a set of tools helping application developers to use any wireless or wired device in an easy way. All devices in HYDRA application are accessible in a uniform way – as semantic web services. In order to achieve this, a developer has to manually “enable” (which simply speaking means adding some device-related information – see section “HYDRA enabling devices” below) all devices, which are supposed to be used in the application using the SeMDA tools. For each device a semantic description is created, which can be used for device discovery, calling device services satisfying various requirements (such as a suitable quality of service) or context-aware application behaviour. This functionality is ensured by SeMDA, thus the development process is simplified and the underlying implementation is transparent to the developer. The following section will briefly introduce basic scenarios of using Semantic Web technologies at the application design time and also in runtime.

3.1. HYDRA enabling devices

To achieve the semantic interoperability, each device in HYDRA application is exposed as a semantic web service. This approach enables to discover and call the device services in a common way, taking advantage of the semantic description. At the application design time, each device has to be “enabled” to be used in HYDRA. This process is called the HYDRA enabling of the device. The developer can “enable” new HYDRA devices using so-called Device Development Kit (DDK). The new device is annotated to a suitable class in the device taxonomy (e.g. mobile device) and a basic description (for example the device model name and number, manufacturer information, energy consumption profile or device discovery information) is added. As particular devices have different connection and communication capabilities, the service calls have to be transformed into web service calls. For each service, the developer has to add a specific implementation of the service. At the same time, each service is annotated to a suitable service taxonomy class. Generally, each device used in HYDRA application provides two sets of services:

- Common services provided by all HYDRA devices including basic information and functionality services, such as “start WS”, “stop WS” or “get WS endpoint”; the energy consumption related services, such as “current power consumption or remaining battery”; and also the storage and security services.
- and the device specific services.
Devices use various low-level communication protocols, such as ZigBee or BlueTooth. Each device has to be physically discovered by one of the communication protocol specific discovery managers. When a device is physically discovered in the network, each communication protocol provides the set of low-level device information. In order to be able to automatically discover the specific device in the future, the low-level discovery information is stored.

The role of semantics

The whole process of HYDRA enabling devices is guided by the ontology containing basic information on the device, service taxonomies, device model, information on the manufacturer, energy consumption profile, and low-level discovery information. Developer browses through the taxonomies provided by the ontology when selecting the suitable device or service class. Basic information and energy consumption are entered into forms automatically generated from the ontology. Once the device is enabled, the new ontology instances are automatically generated and the ontology is extended by the new device model. The ontology contains one instance for each specific device model. The device ontology representation is used at runtime for device discovery and for retrieving all information required for the service calls. Furthermore, information from related configuration files used at the phase of the device development is also stored in the ontology to be reused in the future in order to make the development of new similar devices easier. In the case of HYDRA enabling devices, the ontology can be seen as a storage element guiding the development process and storing the newly added information. Creating a new device introduces only basic semantic representation of the device, which can be later further elaborated. Each device ontology instance represents a specific device model and serves as the static information template. At runtime, when new device joins the HYDRA network, the best matching template is identified by the semantic discovery process, cloned and tied to the physical device using the HYDRA specific persistent identifier. The values of the runtime instance can change as the device changes its state variables (e.g. measured values of sensor). When a physical device leaves the network, the assigned device runtime instance is removed from the ontology.

3.2. Semantic device discovery

When a new device joins the HYDRA network, it is discovered using one of the low-level discovery managers dedicated to various low-level communication protocols, such as BlueTooth or ZigBee. In most cases, the low-level discovery retrieves only a few items of information, dependent on the particular protocol capabilities. At runtime, this information is used to identify the corresponding semantic device model in the ontology which may contain a full description of the device, its services and other relevant information assigned to device model in the design time. The newly discovered device has to be semantically resolved against all templates in the ontology. The semantic resolution is done by comparing the actual low-level discovery information to discovery information assigned to the device ontology templates in the HYDRA enabling process.

Semantic device resolution finishes with success, when there is just one best match. In this case, the runtime instance of the identified ontology device template is cloned and tied to the physical device using the HYDRA specific persistent identifier. The full device information is retrieved from the ontology and passed to the main HYDRA application component called Device Application Catalogue (DAC). The DAC is responsible for mediating actual information of all devices presented in the HYDRA network and for execution of the services. Using the device ontology representation, the DAC creates a proxy transforming the low-level protocol calls to the web service calls using the device implementation created in HYDRA enabling process. The DAC is aware of the persistent identifier used to tie the physical device with the corresponding ontology instance. When semantic resolution fails, it means that there is no suitable semantic model for a device. In this case, the DAC is populated with the generic device information (meta data) and the physical device has no device specific semantic support.

The role of semantics

Each low-level communication protocol represents the device discovery information in a very different ways. Sometimes, all available device information includes only the device model name and the number, sometimes various manufacturer data. In case of more sophisticated protocols, such as BlueTooth or UPnP, a list of services or other extending information can be also available. For each low-level discovery information there exists a model in the ontology.
The low-level discovery information is translated into a SPARQL [8] query and the problem of semantic device resolution is transformed into a graph matching problem. In many cases, the execution of a query retrieves more matching candidates, which have to be further investigated by heuristically comparing possible additional information. Possible additional information for each communication protocol is modelled in the ontology, so in the implementation of the comparison there is no need to hard-code the particular comparison cases. As the discovery information provided by communication protocols may vary, it would be quite difficult to design the suitable fixed data structures supporting such a information variability. The flexibility of semantic representation and uniform information access through the graph matching is here a big advantage.

3.3. Extending the device semantic description

Most of sophisticated applications working with various devices would require to search devices satisfying several requirements. For example, an application working with many sensor devices would require to calculate average temperature of thermometers measuring in degrees of Celsius. Another application would need to select a device able to play AVI video having the largest display with best possible resolution. In many cases it is required to execute service with the lowest energy consumption.

Semantic descriptions of the device models created in the HYDRA enabling process represent only the basic information necessary for the device functionality. This information can be further extended using the ontology administration tools included in the HYDRA Integrated Development Environment (IDE), which serves as the ontology and annotation editor. Based on the most frequent application functionality and internal HYDRA requirements, the ontology has been extended with models of hardware, software and energy profiles, quality of service properties and security properties. Device ontology was also extended by properties used to annotate the extended information to device models. As in most cases the requirements were to search for services having several properties, the domain of annotation properties are mostly classes from the service taxonomy. The hardware, software and energy consumption information is modelled as static structures, there can be one hardware, software or energy profile per device. On the other hand, there can be multiple annotations of quality of service and security properties. Using extended semantic descriptions, the devices and services have the full semantic support and are accessible in various ways.

The role of semantics

One of the biggest advantages of semantic modeling is easy integration of new knowledge. In the HYDRA ontologies, there were integrated external quality of service and security properties ontologies. As these ontologies contained only description of separate domains serving only as knowledge extension and they did not have to be mediated with the existing model, integration was really straightforward. The whole integration process included only creation of properties used for annotation of the new knowledge. Another advantage of using technologies was significant simplification of the extension process in the ontology editor. The whole process of knowledge extension is guided by the ontology. Developer may browse related extension information. In case of static information, such as hardware or energy profiles, the input forms are generated automatically. In case of dynamic properties, which could be added using annotations, the developer can browse the structure of device or service and the ontology editor automatically offers the relevant information items, which can be added to the particular ontology entity. For example, for services a set of instances from the taxonomy of possible functionalities (such as plays video, or measures temperature) can be offered, for service inputs relevant units depending on the device type (for example, for output parameter of temperature sensors, the editor would most likely offer the units of Celsius of Fahrenheit) can be offered etc. All depends on the model in the ontology, what are the domains and ranges of annotation properties. If ontologies are extended with the new domain knowledge annotated to the device or service models, the tool automatically offers new possibilities without any further implementation efforts. Dynamic extension of semantic device properties creates the basic for using the semantic web services in HYDRA. The fact that several application components, or in some cases different HYDRA applications themselves, use the same domain vocabulary, implies semantic interoperability at the semantic level.

3.4. Application context awareness

When designing the application, in many cases it is required to create the application domain model. For example, in the case of the home automation ap-
plication, it is helpful to specify, what locations (e.g. rooms) will the application have, which persons use the application, which devices belong to specific locations or are owned by concrete persons. This way it is possible to describe the application domain, but also to create the background knowledge for the reasoning in the context. When working with the application domain model, sometimes it is required to specify, which devices have to be used for certain operations (e.g. switch on the lamp besides Peter's bed in the bedroom), so it is required to address the device directly using the HYDRA persistent device identifier without any need for reasoning. In other cases, it is required to infer, which devices belong to the specific context (e.g. get average temperature of thermometers in the living room).

The role of semantics

In HYDRA, it is actually assumed, that the application domain models to be used are integrated into ontologies including the properties for annotating devices to the context entities. When developer creates the application, devices to be used for context computations can be selected. These devices can be annotated using the ontology editor to the relevant context entities. Then, in application logic implementation, it is possible to call pre-implemented and parametrized ontology search services. The parameters are specified as a comma separated set of property sequences starting from device or service instance to the target entity. Decision, from which type of instance the sequence starts, is driven by two keywords: device or service. All other properties and target values used must follow the ontology vocabulary. The parameters are of two types (for simplicity, the examples contain only resource names without ontology uri prefixes):

- Parameters, which device or service must satisfy. In this case it is required to specify for all required properties also the target value (usually the name of the instance), e.g. “device isLocated livingRoom, service hasCapability playsVideo”.
- Parameters, which have to be retrieved for further processing in application logic. In this case, only the sequence of properties without target value is used. Target value is expected to be retrieved, e.g. “device hasHardware hasDisplay hasResolution”.

Search methods then retrieve all devices matching parameters to be satisfied. The result for each device contain the set of the properties to be retrieved. In more complicated cases, developer can directly formulate a SPARQL query. In both approaches the developer has to be aware of the ontology structure and vocabulary. Anyway, with the support of inference mechanism, it is possible to achieve reasonably smart application behaviour, while using really simple context models.

3.5. Semantic devices

Each physical HYDRA device provides a set of specific services, which can be directly used by the application developer. For example, the thermometer may provide device specific services, such as “get temperature” or “set temperature”. The idea behind the semantic devices is to enhance the application development by providing the application specific services. For example if there are more thermometers in the room, an application may provide e.g. “get average room temperature” or “hold temperature at specified level” services. The concept of semantic devices brings the idea of specifying the application specific behaviour achieved as the composition of several HYDRA devices organized into complex units, i.e. aggregates of other HYDRA devices. Semantic devices can include physical, but also other semantic devices. For example, the heating system semantic device may include embedded thermometer devices and provide semantic services – the behaviour composed of all embedded services. Each semantic device is defined by a set of semantic services. Each semantic service is composed by a set of requirements in terms of preconditions, which have to be satisfied. There are two kinds of preconditions:

- Static preconditions define the list of persistent identifiers of concrete devices, which will appear in the application and will be used in semantic service execution in runtime.
- Dynamic preconditions used in the runtime to generate the candidate devices matching the specified requirements (e.g. all thermometers measuring in degrees of Celsius located in the hall). The requirements are specified in the same way, as was described in section on application context.

For each precondition a minimal cardinality attribute can be also specified, which represents the minimal number of devices, that have to be present in the application to be able to execute the services of the semantic device.

At the design time, developer has to define and implement semantic device services using the DDK tool. In this case, the preconditions defined for each
service are used to automatically generate the class of proxy implementation using the configuration attached to the semantic models of used devices. At the runtime, each time a new device joins the application, the semantic devices are rediscovered and the required devices satisfying defined preconditions are automatically tied with the semantic devices.

The implementation of semantic device can be realized as a combination of statically defined devices and the orchestration of services based on the dynamic selection of devices. The static definition is used only in the case, when semantic service has to work exactly with some specific devices. But this specification does not entail any limitation for using also orchestrated devices. For example, the developer may decide to create a specific temperature alert device using just some selected thermometers in the room, which have to be specified (thermometers are specified as the concrete devices – static mapping).

When the temperature measured by selected thermometers decreases below some level, the semantic device may perform a “low temperature alert”, e.g. by sending a SMS to some of the mobile devices owned by the resident (the device is specified only by ownership or by device type – dynamic mapping).

The more complex semantic devices may be also used as the decision units providing a specific functionality in terms of effectiveness by some specified criteria. For example, application may use two semantic devices capable of controlling light in the room. One semantic device controls the lamps, another controls the blinds. These two devices may be composed into a more complex semantic device, which would be capable for example to save energy. Using the specific information, the device will be able to decide, how to perform the light control. At a summer day time it may use the blinds controlling semantic device to control light. In the evening or in winter it can prefer to use the semantic device controlling the lamps. Using more information about devices, e.g. various kinds of energy profiles, semantic devices can be used as standalone units implemented to perform the operations while satisfying the specified goals (e.g. energy saving). The application development can be radically simplified by reuse of existing HYDRA (enabled) devices as semantic devices adapted to the specific environment.

Fig. 1: Part of semantic description of device.
The role of semantics

The developer creates a new semantic device in the DDK tool, with a picture of the target device functionality. The ontology support is useful when selecting the proper component devices or services, which should be embedded in the semantic services. The ontology driven design simplifies the whole development process, as the ontology editor provides a wide set of specific listings and views filtering the devices by the specified requirements (e.g. quality of service). For all devices having the suitable functionality, there is immediately accessible a list of full signatures of available services. When the devices are selected, the configuration information attached to device models is used to automatically generate the proxies and code in the DDK tool. Thus, the role of developer does not have to deal with the low-level details of devices, but can focus on the selection of proper devices and the implementation of pure logic of semantic services.

At the runtime, the presence of devices in the HYDRA network may change. When devices join or leave the HYDRA network, the ontology is continuously queried and all affected semantic devices are re-discovered. Each change may cause, that some of available semantic devices are disabled, whereas some may be enabled for usage. For more, semantic devices have to ensure the real-time orchestration of embedded devices. Each time when semantic services with dynamic mappings are executed, the ontology has to infer the required devices, actually presented in the HYDRA network, matching the specified preconditions.

4. Semantic description of devices

The most of models used in HYDRA are created as OWL-DL [6] ontologies. With respect to characteristics of the domain, a careful modelling strategy was used. Development of ontologies was strictly following the user and application requirements to keep them simple. The ontologies in HYDRA are used for both static information storage and also complex query answering purposes. This section will briefly introduce the core models used.

The HYDRA device ontology represents the concepts describing device related information, which can be used in both design and runtime. The basic ontology is composed of several partial models representing specific device information. The initial device ontology structure was extended from the FIPA device ontology specification [4] and the initial device taxonomy was adopted and extended from AMIGO project vocabularies for device descriptions [1]. Part of device semantic description is illustrated on Figure 1. The ontologies used in HYDRA can be outlined as follows:

- The core device ontology contains a taxonomy of various device types and the basic device description including model and manufacturer information.
- Device services are modelled in the terms of operation names, inputs and outputs. Services are also organized into the taxonomy. Services are the basic executable and functional units in HYDRA. To enrich the service description, the model of service can be annotated with additional properties, such as various capabilities, quality of service or security properties. The model of services used in HYDRA was inspired by OWL-S ontology [5]. As the OWL-S was considered too extensive for the project objectives, a more suitable approach was to create simplified, customized ontology for service description based on OWL-S.
- Device capabilities represent the hardware properties, software description and energy profiles. The mentioned information profiles are modelled as static structures, where only one profile of each type can be attached to the device.
- Discovery models contain the models of all discovery information provided by low-level communication protocols. The discovery model is mandatory and is attached to each device. The purpose of device discovery information is the ability to semantically resolve the suitable device semantic model when new device enters the HYDRA network and is initially described only by low-level discovery information depending on communication protocol used.
- The semantic device model represent logical aggregates of composed devices to provide more advanced application related functionality. Semantic devices are modelled as a set of semantic services specified by preconditions, which have to be satisfied for semantic device to be executable. The preconditions specify static or dynamic requirements for devices embedded in the semantic device.
- Application models contain a set of ontologies dedicated to various application domains. Each application model specifies the domain entities
and relations in order to achieve a context aware application behaviour.

- Quality of Service (QoS) model contains descriptions of various aspects of service quality. High level properties, such as taxonomy of service functional capabilities (e.g. plays video or measures temperature) are modelled. The QoS ontology contains also the specification of the lower level service properties, such as response time, availability or reliability. The QoS ontology also contains the taxonomy of various units (such as temperature, time, pressure, the currency).

- Device malfunctions represent various types of errors and failures, which may occur when using the device at runtime. For each malfunction there is a set of possible remedies in the form of text description.

- Security properties specify various security properties, such as protocols, algorithms or objectives, which may be attached to devices or services. To describe the security properties, third party NRL ontology [7] was integrated and annotated to the device ontology.

- Configuration model supports a device creation using the DDK tools. For each created device, the information about the configuration and implementation files used by particular IDE are stored. These files serve as templates of code or IDE project files and can be reused when new similar devices are created. Another purpose of the configuration models is the support for automatic device code generation (e.g. selecting suitable device implementations) for the device development.

5. Conclusions and lessons learned

The main advantages, from which the implementation of Semantic Web technologies in the HYDRA project profited, are: semantic model driven development, where the design process is guided by the ontology; easily extendible knowledge base, even in the case of integrating the external ontologies; and flexibility of semantic search including the inference support.

As the ontologies were designed to be simple, in most cases the RDFS [2] language would be satisfactory. The more advanced application functionality requirements, mostly in the areas related to context awareness already required the usage of OWL-Lite or OWL-DL languages.

In the process of ontology design, the most important problem, which has arisen, was the proper selection of existing ontologies. For example, when models for services were designed, OWL-S and WSMO [3] technologies were investigated. But, for the project purposes, both of them were considered overly complicated. So, the better trade-off was to design a customized ontology for service representation instead of using the existing one. On the other hand, integration of external knowledge bases, such as models for QoS or security properties was really easy and straightforward.

In general, the experiences in using the Semantic Web technologies in HYDRA has led to the practical results, which has net many, though not all, of the expectations.

Acknowledgements

The work presented in the paper is co-funded by the European Commission within the FP6 IST-2005-034891 Project “HYDRA – Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture”

References