Semantic Devices for Ambient Environment Middleware

Peter Kostelnik
Technical University of Kosice
Letna 9, Kosice
04001, Slovak Republic
Peter.kostelnik@tuke.sk

Martin Sarnovsky
Technical University of Kosice
Letna 9, Kosice
04001, Slovak Republic
Martin.sarnovsky@tuke.sk

Jan Hreno
Technical University of Kosice
Letna 9, Kosice
04001, Slovak Republic
Jan.hreno@tuke.sk

Matts Ahlsen
CNet Svenska AB
Svärdvägen 3B
182 33 Dandaryd, Sweden
Matts.ahlsen@cnet.se

Peter Rosengren
CNet Svenska AB
Svärdvägen 3B
182 33 Dandaryd, Sweden
Peter.rosengren@cnet.se

Peeter Kool
CNet Svenska AB
Svärdvägen 3B
182 33 Dandaryd, Sweden
Peeter.kool@cnet.se

Mathias Axling
CNET Svenska AB
CNet Svenska AB
Svärdvägen 3B
182 33 Dandaryd, Sweden
Mathias.axling@cnet.se

ABSTRACT

The paper describes the approach for modelling and using semantic descriptions of devices in the HYDRA project middleware ("HYDRA: Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture") funded within the FP6 IST Programme). The concept of semantic device is introduced and the main use cases for semantic device descriptions are presented with a focus on the Model-Driven Architecture (MDA) approach in HYDRA. An overview of the ontologies representing the semantic device descriptions is presented.

1.INTRODUCTION

In the ambient world of the near future, interconnected intelligent devices will surround us, at home, work, or while travelling. These devices and their local networks will also be connected to the outside world through broadband and/or wireless networks. Numerous services to support us in our personal life will be provided through these ambient devices and over the connection to the outside world. In order to cope with the huge variety of capabilities of the 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.

One of the goals of the HYDRA project [3] is to provide a middleware solution for interconnecting the various heterogeneous devices with different services and capabilities. 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 for ubiquitous networked devices. The SOA and its related standards provide interoperability at a syntactic level. However, HYDRA also aims at providing interoperability at a semantic level. One of the objectives is to extend the syntactic interoperability to the application level, i.e., 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 (Semantic 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, i.e., 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 devices and the application's view of the device. We introduce the concept of Semantic Devices, which represent the models of the real devices and serve as the logical units which can be semantically discovered and providing the information about device capabilities and services.

2.SEMANTIC DEVICES

The services offered by physical devices are generally designed independently of the particular applications in which the devices might be used. A semantic device on the other hand represents what a particular application would like to have. As an example , when we are designing the 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 logical

devices might 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 of a lamp (blind and lamp being HYDRA devices). We call these logical aggregates of devices and their services for *Semantic Devices.*

Semantic Devices should be seen as a programming concept. The application programmer designs and programs his 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 Celsius" and "Set warning level:17 degrees 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 1-to-1 from a semantic device to a physical device or mappings that allow composition.

- An example of a 1-to-1 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 about which devices exists 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 mappings 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 create 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. But then it might find two different thermometers which both claims they can measure temperature. The semantic device could then query about which of the thermometers can deliver the temperature in Celsius, with what resolution and how often. In this case it might only be one of the thermometers that meet the 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 to much. By some reasoning the semantic device 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 descriptions of semantic devices are based on a device ontology (described below).

The concept of semantic devices supports the semantic MDA approach used. There are two uses of the semantic MDA in HYDRA. Firstly, it is relevant at design-time, supporting both device developers as well as application developers, and secondly, at run-time where HYDRA applications can use knowledge provided be the semantic MDA.

2.1 Semantic MDA at design-time

2.1.1Model-driven code generation for Semantic Devices

The semantic descriptions of devices and their services are used at 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. We are currently making the HYDRA SDK available in an object-oriented language environment, integrating the objects a developer can use to access the devices and services (thru device and service proxies) as well as the properties from the device ontology connected to devices and services. A HYDRA developer can specify a service to be used, and leave the device as generic as possible. 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 a HYDRA application from the selected devices and services. The services could also be used by a service orchestration engine (however, considering that some applications will be standalone and have a fairly small footprint, this may not be suitable for all HYDRA applications).

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 design time as well as applications that always query the device ontology for new types of services that match the descriptions.

2.1.2Model-driven code generation for physical devices

For devices that have sufficient computing capacity to host a web service interface, HYDRA provides a tool to generate skeleton code. Such a tool takes as inputs an interface description and a semantic description of the device on which a web service should run. 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 the available resources of a device, either embedded stubs or 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.

2.2Semantic MDA in run-time

2.2.1Models for discovery

At 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 only be interested e.g. in "HYDRA SMS Service" and any device entering the network (or application context) that fits in these general categories will be presented to the application. The application will then work against the more general device descriptions. This means that an application should only know of the (types of) devices and services selected by the developer when it was defined. This still 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 be mapped to a specific device.

A HYDRA application may present an external interface so that it can be integrated with other applications and devices. It will do this by announcing 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.

2.2.2Use 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 other application/device, HYDRA also uses the semantic MDA to define and enforce security policies. 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.

3.Semantic description of devices

The HYDRA device ontology presents the basic high level concepts describing device related information. 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 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. Specific device models are created as the instances of device ontology concepts are 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 device discovery process, the discovery information is resolved using the ontology and the most suitable template is identified. The identified template is cloned and a 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 the all 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 [4]. The different parts of the model are contained in separate ontology files. The references between more general and specific ontologies are realized using the OWL import mechanism. The core of the device ontology consists of the basic device information. The basic *HydraDevice* concept is sub classed by the taxonomy of device types and serves as the root ontology concept. The important property of the *HydraDevice* concept is the *deviceId*, which represents the unique device URI. Device URI is the unique identifier of the device template or the HYDRA Id assigned to specific run-time device instance. In order to create aggregate devices, there can be a recursive property *hasEmbeddedDevice* of the concept, which enables to create more sophisticated models composed of multiple devices. The basic device information is contained in the concept *InfoDescription*, which contains the device name, manufacturer information etc.

Part of the core device taxonomy and basic information is illustrated in Figure 1.

The structure of the semantic device description is divided into four modules connected to the core ontology concepts:

- device capabilities (hardware, software properties and state machines)
- device services
- device malfunctions
- device security properties

The initial device ontology structure was extended from the FIPA device ontology specification [2]. The initial device taxonomy was extended from AMIGO project vocabularies for device descriptions [1].

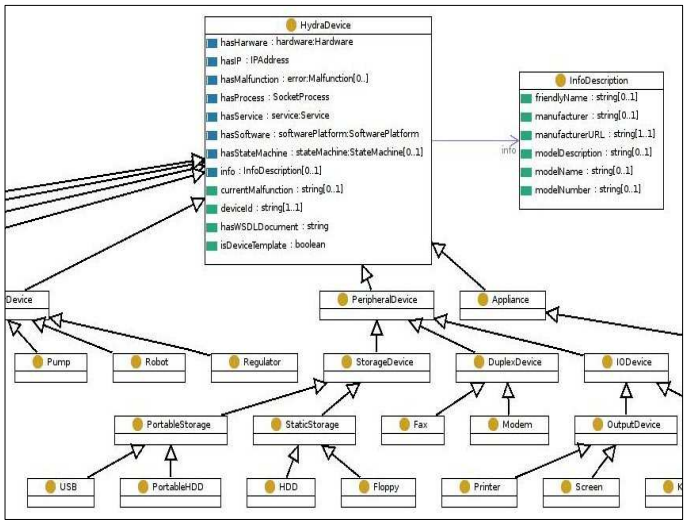

Figure 1. Device taxonomy

3.1Device capabilities

Semantic description of device capabilities represents the extended device information.). The device capabilities are divided into three modules directly referred by *HydraDevice* concept:

- the hardware model is based on the hardware description part from W3C *deliveryContext* ontology [8]. It 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.). The model was refined with concepts related to power-awareness and additional network connections. The root of the hardware model is the Hardware concept referred by *HydraDevice* concept. All hardware capabilities are represented as the subclasses of Hardware concept. Device hardware capabilities are used for generation of embedded device services code.
- the software platform related model is important for generating the code for embedded device services. It provides descriptions, such as software platform classification, rigorous specification of platform dependencies and resource consumption relationships. The software model is composed of several parts, including a software platform, operating system, Java and .Net models.
- a 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 to generate state and transition related code.

3.2Device services

The device services ontology component presents the semantic description of device services on the 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 [6] standard, which is currently the most complete description of semantic markup for services following the web service architecture. The OWL-S approach was taken as the starting point for HYDRA service model.

Each service is represented by the Service concept, which serves as the root concept for subclasses creating the service taxonomy. The Service taxonomy represents the service categorization. As a service may belong to multiple categories, the instance of specific service may be of more rdf:types representing several categories (e.g. light switch device may be of type *ScheduledService* and/or *PowerSwitchService*). In the actual model, each service represents one WSDL operation, thus an important property of Service concept is the *serviceOperation*, which contains the WSDL operation name and serves as the identifier of the service. Each *HydraDevice* concept may have as many services, as needed (depending of which services are provided by the specific device). The Service concept references three components:

- ServiceProfile concept presents the basic service description used mainly for service discovery process. Service profile describes the general information, such as human readable service name and description, service capabilities and service inputs and outputs. Capability concept is used to describe the specific service capabilities related to service functional properties, such as ability to handle various media formats or to handle required device states. *ServiceInput* and *ServiceOutput* parameters are specific subclasses of general *ServiceParameter* class and should be annotated to semantic model describing various input and output types in the syntactic (for example, string, number) and semantic way (for example, address, user name, etc.). Capabilities and input/output descriptions can be used for suitable service discovery or service composition, but also for semi-automatic or fully automatic generation of self-descriptive service user interfaces.

- The ServiceProcess concept aims to describe the service process model, which defines if the service represents the immediately invocable atomic process or work-flow of composite processes. In the actual implementation, the *ServiceProcess* concept is empty and each service is treated as the atomic process.
- ServiceGrounding concept aims to specify the details, how to access the service and physically realize the service invocation. In the actual implementation, the *ServiceGrounding* concept is empty. For grounding information, *HydraDevice* concept has the (SA) WSDL document reference (using *hasWSDLDocument* property), which contains the operations modeled by Service instances of device. For more, in specific cases, the service models can be generated from (SA) WSDL document.

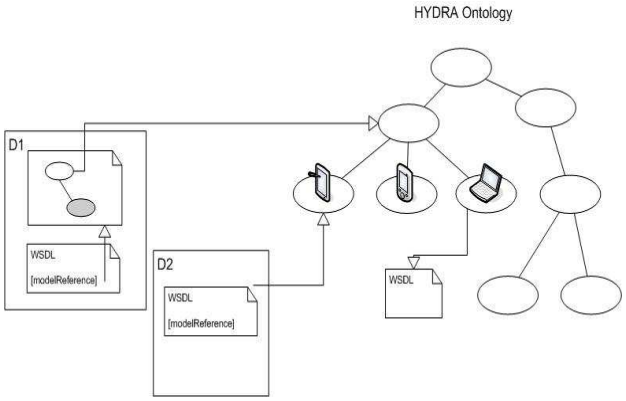

Figure 2. Illustration of SAWSDL references to ontology

In general, devices in HYDRA are provided with semantic descriptions by combining the device ontology with the SAWSDL standard for annotating device WSDL files (see Figure 2).

3.3Device malfunctions

The semantic model of device malfunctions represents possible errors that may occur on devices. Each malfunction, represented by the Malfunction concept, is described by the error code and the human readable name information. The model contains the one-to-many relation of malfunction to cases, represented by *MalfunctionCase* concepts, which describe the possible cases and remedies for each fault.

In order to have a flexible model of malfunctions, the *Malfunction* concept can be further sub classed to several malfunction levels or severity, such as error, fatal, warning, info, etc. Possible severity levels can be further extended by the hierarchy of specific faults.

Connecting the device taxonomy to the malfunction taxonomy creates a flexible representation of fault states, which may occur on various device types and the possibilities of their solutions.

3.4Security capabilities

Security capabilities ontology represents the security properties of devices and the services, such as protocols, policies, mechanisms or objectives. The main concept of security ontology, the *SecurityConcept*, is referred by *HydraDevice* and Service concepts using the *hasSecurityProperty* connection The NRL ontology [5] 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. The NRL ontology was designed to describe the security concepts related to any resource type, to cover the information on the various levels of detail and to be easily extendible. Information contained in the ontology is designed with main focus to the functional aspects of capability, content and parameters.

4.CONCLUSIONS

In this paper, we have given a brief overview of the semantic device concept used in the HYDRA middleware, with main focus to middleware MDA design. The first part of paper introduced the semantic devices modeled in ontologies and widely used to support the HYDRA MDA in both design and run-time phase. The usage cases of semantic device were briefly presented in the tasks of model-driven code generation for physical and semantic devices, device discovery and security requirements resolvence. The second part of paper has provided the brief description of ontologies used to represent the semantic devices properties, services and capabilities.

5.ACKNOWLEDGEMENTS

The work presented in the paper is supported by the EC within the FP6 IST-2005-034891 Project "HYDRA – Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture"

6.REFERENCES

[1] Amigo middleware core: Prototype implementation and documentation, deliverable 3.2. Technical report, Amigo Project, IST-2004-004182, 2006.

[2] FIPA Device Ontology Specification, Foundation for intelligent physical agents, 2002.

[3] HYDRA: Networked Embedded System middleware for Heterogeneous physical devices in a distributed architecture", Project Proposal, September 2005.

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

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

[6] OWL-S: Semantic Markup for Web Services, 2004.

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

[8] Delivery Context Ontology. W3C Working Draft, 2007.