Integration of Heterogeneous Building Automation Systems using Ontologies

Christian Reinisch, Wolfgang Granzer, Fritz Praus and Wolfgang Kastner
Vienna University of Technology, Institute of Computer Aided Automation
{cr, w, fpraus, k}@auto.tuwien.ac.at

***Abstract*—The challenge of integrating heterogeneous systems in order to combine their functionality is of utmost importance for the further deployment of building automation systems. The goal is to allow comprehensive communication among the systems. This will provide enhanced possibilities thus making way for intelligent buildings. Traditionally, integration is achieved using gateways which require considerable configuration effort. To alleviate this overhead and provide a unified system view, a generic application model is proposed that can accommodate all functionality found in building automation systems. The employment of this model promises several benefits such as a central point for configuration and system access. The method of choice are ontologies, which allow to offer a seminal representation of knowledge, an abstraction of the heterogeneous network infrastructure and automatic reasoning on the stored knowledge.**

I. Introduction

Today's building automation systems (BAS) allow the automatic control of a variety of fields. While originally only the core domains lighting/shading and heating, ventilation and air-conditioning (HVAC) were targeted, nowadays other systems, such as access control and fire alarm systems, are considered for an integration.

The ideal BAS is an all-in-one solution that allows total control of all conceivable scenarios in a building. Although a shift towards higher integrated systems can be observed, this wish still contrasts reality. Current BAS allow to span all application fields only if they are composed of several independent subsystems. However, this integration is very challenging and mostly does not give the desired impression of a homogeneous network to the users.

Still, it is a good idea to build on existing protocols due to compatibility and cost reasons. The main open protocol contenders that can be considered as part of an all-in-one system are BACnet [1], KNX [2], LonWorks [3] and ZigBee [4]. All of them span more than one application domain, but are not completely versatile (e.g., KNX supports lighting/shading and HVAC applications, but has limited features regarding security and safety). Apart from these protocols, also stand alone solutions exist that are dedicated to a specific application domain. Two prominent examples are DALI and MBus, which are used exclusively for lighting and metering, respectively. Furthermore, numerous proprietary systems for various domains exist, which can not be considered suitable for an integration as protocol details are seldom published.

Obviously, it is not sufficient to deploy several systems each covering one domain, but these different systems need to be interconnected to allow an information exchange for the distributed applications. From the communication point of view this interconnection can be achieved using gateways or multi-protocol devices [5], thus allowing to establish the 2-tier model for BAS. In the 2-tier model, different building automation networks are interconnected with the help of a common backbone. This introduces two challenges. First, all devices of each technology employed need to be configured independently, probably with vendor-specific tools. Second, also the gateways require considerable effort for configuration. All data points of interest in a particular subsystem need to be translated by the gateways to allow their use in other systems. This is even complicated due to the fact that most technologies are based on different assumptions and concepts of the outside world and thus represent these concepts differently.

The first step towards a homogeneous view is to define BAS in a general and abstract way. A promising technology for this task are ontologies, which cannot only be used to represent knowledge but also to (automatically) generate new knowledge. In the following section, an introduction into ontologies is given and related work is reviewed. In Section III, the different application models of the four major open BAS protocols are presented and their main differences are pointed out. Ontologies and their benefits in BAS are discussed in Section IV, while the generic application model is presented in Section V. Finally, an example use of our ontology is given in Section VI.

II. Overview of Ontologies

An appropriate definition of ontologies is given in [6]: (An ontology is) "A shared and common understanding of a domain that can be communicated between people and heterogeneous and distributed systems". Derived from this definition an ontology is basically a way to store, organize and represent knowledge. More specific, concepts belonging to a particular domain and their relations can be defined in a generic way, thus forming a model of this domain. Additionally, ontologies allow to reason on the stored data, so that an automatic generation of new information is possible.

Ontologies are commonly designed and developed with tool support and are often based on the *Resource Description Language Schema (RDFS)* [7] and the *Web Ontology Language (OWL)* [8] as description languages. Probably the most prominent tool for this purpose is *Protege*[1] which was developed at Stanford University.

Ontologies are already used in various scientific fields, ranging from biology to linguistics. Also technical systems are considered to be augmented with their help. In the industrial automation field, ontologies allow to access fieldbus device information (device descriptions) using Semantic Web technologies and thus easier creation, processing and management of this information becomes possible (cf. [9]). In [10], an ontology called DomoML representing data of household appliances and their environment (i.e., rooms, furniture, etc.) is defined. The main target is human home interaction with the goal to improve pervasiveness and interoperability of domestic devices. A similar approach as DomoML but for industrial and building automation systems is outlined in [11]. The main aim is to establish interoperability among heterogeneous automation networks at the level of web services.

All three examples primarily intend to make systems ”ready” for the Semantic Web, thus exposing and matching their services over the web. However, ontologies hold many more possibilities than merely providing an access point for the Semantic Web. In particular, ontologies allow the representation of application specific knowledge in an abstract and organized way. In BAS, this is especially promising regarding the heterogeneous network infrastructure often found there.

III. Application Models in BAS

Devices that are proven to comply with a particular technology – even if sold by different vendors – are able to communicate with each other. This vendor-independent compliance is called *interoperability*. Interoperability not only has to be guaranteed for communication but also for distributed applications, so that devices of different vendors can be deployed in one system.

Therefore, all technologies define their own application model which states how data is represented (data format, encoding) and how the communication between applications has to be realized (methods to manipulate data). Obviously, each application model is specific for a particular technology, which prevents a direct (i.e., translation/gateway-free) communication across technology borders.

A. BACnet

While BACnet defines the services that are used by applications to communicate with each other, the internal data structures that are used to store application data are left open by the standard. However, to guarantee interoperability, the network-visible representation of the data structures has been defined by the standard. This network-visible part of a single data element is called a *BACnet object*. Each object has a dedicated object type and represents a collection of properties. Each property has a data type that defines the size and the encoding of the particular data element.

In the current BACnet standard, 25 different objects and nearly 200 different property types as well as different object access services are defined. The most important services used to access and manipulate objects are *ReadProperty* (i.e., read the value of a property), and *WriteProperty* (i.e., set a new property value).

For example, BACnet 2004 defines generic binary and analog object types such as `BACnet Binary Output Object Type` and `BACnet Analog Input Object Type`. It is the responsibility of the application program to map the functionality of a simple light to a `BACnet Binary Output Object Type` or a room temperature value to a `BACnet Analog Input Object Type`. However, there are efforts underway to standardize more application-specific object types in BACnet. For example, in Addendum i, basic BACnet objects for lighting have been defined [12].

B. KNX

The KNX interworking model [13] distinguishes between different application domains (e.g., HVAC, lighting). Within these domains, models for the desired applications are defined (application models). The functionality of these mostly distributed applications is spread across various so called functional blocks. Each functional block is implemented by a single device, and each device can host multiple functional blocks. Each functional block is described by a well-known behavior and consists of one or more datapoints. A single datapoint represents a single data of the application. Depending on the type, this may be an input or an output as well as a parameter that influences the behavior of the functional block. To be able to guarantee interoperability, each datapoint has a defined datapoint type. This datapoint type states the format (i.e., the bit length), the encoding, the range (e.g., the upper and lower bound of the value) and the unit (e.g., percent) of the datapoint. Each datapoint can be accessed using a specific KNX communication service as well as a particular address (e.g., group object datapoints use group addresses and are accessible through group communication services).

C. LonWorks

In LonWorks, applications define so called *network variables* (NVs), which are shared by devices over the network. For interoperability reasons, the LonMark Association[2] standardizes the NVs through the definition of *standard network variable types* (SNVT). A SNVT is an exactly defined (e.g., encoding, physical unit), calibrated, filtered and linearized engineering value which allows its doubtless interpretation. Besides the SNVTs, also so called *standard configuration property types* (SCPTs) are defined and used to access configuration functions within a device (e.g., changing parameters).

To satisfy the demands of special domains such as building automation, the LonMark Interoperability Association defines so called standard functional profile templates (SFPTs) for interoperability. These SFPTs are application-specific and include NVs and configuration properties, defaults, and power-up behaviors.

[1]http://protege.stanford.edu/

[2]www.lonmark.org

D. ZigBee

ZigBee applications are implemented by so called *application objects* that are distributed across the ZigBee devices, with one ZigBee device hosting a maximum of 240 application objects. Each application object hereby implements a specific functionality of the distributed application. Within the application object, the functionality is represented by so called *clusters*. A cluster is a collection of commands and attributes. While a single attribute of a cluster represents a single data of the process to be controlled (i.e., the state of a light), commands are used to manipulate these attributes as well as to initiate actions within the device. Therefore, clusters act as interfaces to the application object.

The exact structure of the application objects and their associated clusters (including the specification of the clusters' attributes and commands) are not defined by the core specification. However, to enable interoperability between ZigBee devices, so called *application profiles* are defined. An example of such an application profile is the recently published *ZigBee home automation public application profile* [14].

Application profiles are defined for a specific application domain. They contain a set of *logical device descriptions* that define the functionality to be implemented. This functionality is represented by so called clusters, whose implementation can either be mandatory or optional. An application object is thus an implementation of a logical device description (or at least of all its mandatory clusters) within a physical ZigBee device, e.g., an `On/Off Light`.

IV. Ontologies in BAS: Benefits

As mentioned in Section I, an integration of distributed applications that are spread over heterogeneous networks is not a trivial task. In such a case especially the different application models pose a significant problem. These application models do not only use different communication services at the application layer but also different data structures to store application data. Therefore, a mapping between the different application models is necessary. In a concrete installation, this means that the network-visible data structures of the application data, i.e., the datapoints, must be mapped from one protocol to the other (*datapoint mapping*). Likewise, also the services used to access them have to be translated (*application layer service translation*). Additionally, it has to be defined which datapoints shall be used for data exchange and which of them are interoperable with the other technology. This step is referred to as *binding*.

Considering the integration of heterogeneous networks, the employment of an ontology holds major benefits. First, integrated BAS can be configured centrally by accessing and modifying the ontology only. For all protocols that are employed, an (automatic) translation of the (centrally managed) configuration data into the technology-specific form (i.e., the instances representing the used BAS) becomes possible. This central management approach facilitates BAS management and guarantees system consistency. Second, the ontology with its machine-readable data representation also provides a convenient access point for other systems. The accessing systems may comprise all kinds of web services (e.g., system visualization over the web) and also enterprise resource planning (ERP) and facility management tools.

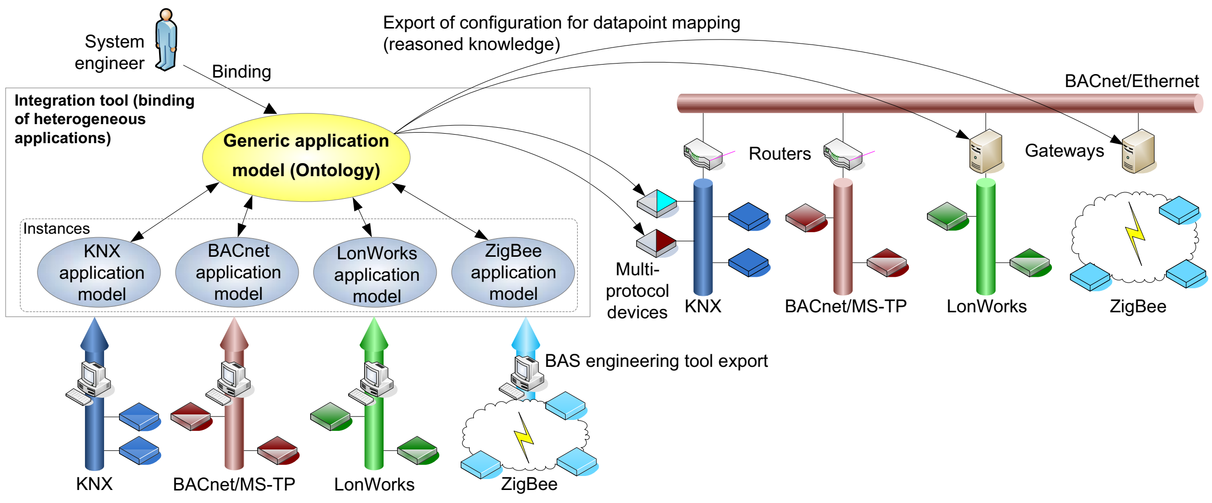
Finally, the ontology alleviates the overhead that is encountered when heterogeneous systems shall be integrated. In particular, even two possibilities exist to realize the integration. On the one hand, it is possible to allow data exchange on the web service level. This shifts the interconnection on top of the application layer. If two or more different technologies shall exchange data, matching web services of each technology have to be found and bound. However, this approach requires the use of web services even if they are not needed for any other purpose. Therefore, it is likely to introduce overhead. On the other hand, the currently more common way of system integration is the use of gateways. These dedicated devices cater for a translation of protocol specifics up to the application layer. A drawback of this approach has always been the effort needed for gateway configuration. As we will show in the remainder of this paper, this effort can be considerably reduced with the help of ontologies.

The basic idea of the ontology for BAS is to separate generic information from an installation dependent one through the definition of the abstract model (e.g., the abstract BAS device description) and concrete instances of it (e.g., a BAS device instance representing a particular technology with specific parameters). Additionally, protocol-specific and domain-specific knowledge (e.g., BAS specific vocabulary) are part of the ontology and can be referenced from other ontology parts.

A BAS ontology thus can be used to abstract an existing BAS installation and represent this particular installation in a generic form. All configuration and management tasks can now be performed directly on the abstracted representation and be automatically distributed to the different underlying technologies. This *generic application model* will be described in detail in the next section.

This approach allows two major improvements of integration effort, which are also shown in Figure 1. First, it is no longer necessary to define mapping rules covering all different protocols, but it is sufficient to map each protocol to the abstract representation (the ontology), which acts as a common base for all protocols. In a BAS that uses four different protocols, normally at least six mappings (pairwise for each two protocols) would be needed. Using the ontology, four mappings are sufficient. Furthermore, if a new technology is considered for integration, again just a single mapping must be defined (i.e., from the technology to the ontology), while the other approach would require four additional mappings. Therefore, future extensions are highly facilitated.

Another advantage is that the required bindings can be established directly in the generic application model using a comprehensive integration tool. This binding could for example be accomplished using the ontology design tool itself (e.g., *Protege*), while advanced functionality could be added in form of plugins. On the one hand, a centralized binding approach



System engineer
Binding
Export of configuration for datapoint mapping (reasoned knowledge)
BACnet/Ethernet
Integration tool (binding of heterogeneous applications)
Generic application model (Ontology)
Routers
Gateways
Instances
KNX application model
BACnet application model
LonWorks application model
ZigBee application model
Multi-protocol devices
KNX
BACnet/MS-TP
LonWorks
ZigBee
BAS engineering tool export
KNX
BACnet/MS-TP
LonWorks
ZigBee

Fig. 1. BAS integration and advanced use cases enabled by ontologies

facilitates the identification of interoperable devices (e.g., the abstract representation allows to determine that a KNX light switch can be used with a ZigBee light). On the other hand, all configuration is performed centrally and can thus be easily exported. Changes that are made are automatically converted to meet the respective protocol semantics and committed to these protocols. Additionally, also the gateway configuration can be derived automatically using the reasoning capabilities of ontologies. Consider, for example, a ZigBee light switch that shall be integrated into a KNX lighting system. To achieve this, the engineer now binds the generic datapoints of these two functions. After having finished the binding of all desired functions, it is possible to automatically generate and export this binding in form of configuration data. This configuration data can then be loaded into the gateways or multi-protocol devices respectively.

V. Generic application model

The key challenge of the proposed solution is the definition of the generic application model. This application model shall be expressive enough to be able to model all different types of distributed applications typically found in the building automation domain. A possible solution would be to evaluate the application models of the existing BAS standards, and decide on the most suited one for further development. However, such a process is not likely to yield acceptable results for all targeted systems, as each model has particular advantages and disadvantages and is sometimes focused on a particular domain. Therefore, instead of preferring an existing model, our goal is to define a new and generic application model. This model shall allow an easy mapping between the generic application model and the standard-specific application models of the most popular open BAS standards (cf. Section III). At the same time, our model shall be generic enough to accommodate to the application models of future BAS standards.

The basic structure of the generic application model is shown in Figure 2. In this model, the nomenclature is based

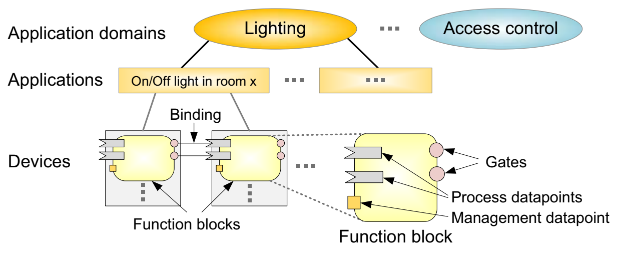


Fig. 2. Basic structure of the generic application model

on the one used in IEC 61499 [15], but modifications were made with respect to the building automation domain vocabulary. The functionality of BAS is realized by *distributed applications* which are spread across different devices. Each application belongs to a different *application domain* (e.g., the application "On/Off light in room x" is a member of the application domain "lighting"). In a distributed BAS, devices host one or more so called *function blocks*. Each of these function blocks implements a particular part of the application functionality. While a device can host multiple function blocks (different or even of the same type), we assume that one function block is always dedicated to exactly one device.

Each function block consists of one or more *process datapoints* (PDP), one or more *management datapoints* (MDP) and so called *gates*. A single PDP represents a single data of the application process (e.g., the current state of a light switch or light) that can be read and (optionally) be written. The structure of a process datapoint used in our model is similar to the one defined in [16]. Each process datapoint has a present value and some associated meta-data. This meta-data further characterizes the datapoint by describing, for example, the datapoint type, valid range and unit of the present value.

A MDP is comparable to a PDP. It also consists of an actual value and has meta-data associated. Additionally, it can be declared as read-only or writable. The main difference to the

PDP is that a MDP does not represent any application data. Rather, it is used for configuration and/or maintenance (e.g., to change a setpoint or access logging information).

Finally, function blocks also contain one or more gates, which do not represent any data. Gates are used to read and/or write PDPs and therefore they can invoke particular actions within a function block. The necessary association between a gate and its corresponding datapoint is established via a binding. Since multiple gates can be bound to multiple PDPs, m:n relations are possible. The binding of gates to MDPs is not foreseen in this model. The reason is that MDPs are only accessed by dedicated management devices (e.g., a logging station) which are not considered as part of the actual application process. Thus, a binding is not required. Figure 4 shows an example of a lighting application where a gate belonging to the function block `On/Off_Switch` is bound to the function block `On/Off_Light`.

The generic application model is defined through an ontology using the web ontology language OWL. The ontology consists of multiple OWL classes that are derived from the root OWL class `owl:Thing`. Figure 3 partially shows the main classes without their properties. These classes are explained in the rest of this section.

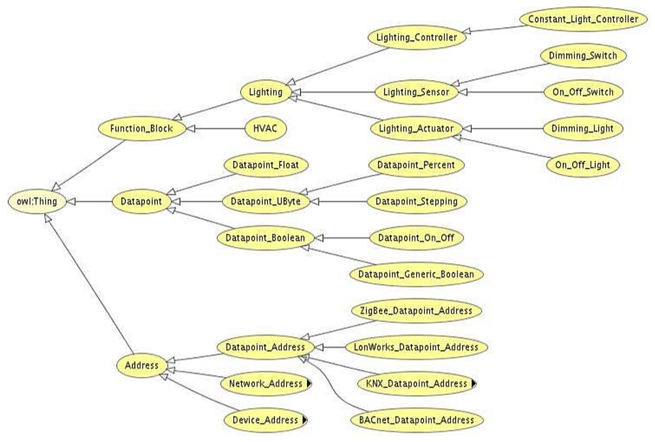


Fig. 3. Excerpt of the BAS ontology

A. OWL class `Function_Block`

This class contains different so called *application domain subclasses* that group the function blocks of each application domain (e.g., lighting or HVAC). Each application domain class has three subclasses. One contains the classes that implement actuator functionality, one for implementing sensor functionality and one for implementing controller functionality. These, in turn, contain the different classes that represent the specific function block types.

Since each function block has different PDPs, MDPs and gates, each function block class has a list of references to other datapoints[3]. In OWL, these references are represented as `OWL object properties`. Gates are represented as object properties to PDPs (i.e., one PDP can be referenced by multiple gates). PDPs and MDPs are represented as `OWL inverse functional object properties`. Therefore they only reference to exactly one instance of a datapoint (i.e., a PDP or MDP can only be referenced by exactly one function block).

[3]The term datapoints is used as a collective term for process and management datapoints.

B. OWL class `Datapoint`

The OWL datapoint class contains different subclasses where each subclass represents a specific, generic datapoint type (e.g., boolean or unsigned integer of length 8 bits). Each generic datapoint class has again several subclasses that are used to further restrict the range of the datapoint type or to specify a different encoding scheme (e.g., a subclass of a 8 bit unsigned integer may represent a percentage or an enumeration type that specifies some configuration modes).

Each datapoint class has an OWL datatype property `is_of_type` that specifies whether the datapoint is a MDP or PDP. Additionally, a class has an OWL datatype property `value` that represents the actual value as well as additional OWL datatype properties that are used to store the datapoint meta information.

Furthermore, gates shall have access to both PDPs and MDPs. Therefore, each datapoint has an `OWL object reference` to an instance of the class `Datapoint_Address` (cf. Section V-C).

C. OWL class `Address`

This class and its subclasses are used to store address information. The three different subclasses `Datapoint_Address`, `Network_Address` and `Device_Address` are differentiated. Each of them has again multiple (one for each supported BAS standard) subclasses to represent technology specifics (e.g., `BACnet_Datapoint_Address`, `KNX_Device_Address`).

All `Address` subclasses and the instances of `Datapoint_Address` are used to store the addressing information and to allow this information to be referenced by PDPs and MDPs. Therefore, the address information is used by the gates and management devices to access the corresponding datapoints.

VI. An Implementation for Lighting

The following paragraphs shall give an impression how the generic application model is used in practice. Clearly, not all features can be outlined in full detail, but an example use shall be shown. Therefore, an ontology that represents the application domain `lighting` has been defined using the OWL engineering tool *Protege*. The goal is to support a mapping of the application models of all four major BAS standards into the generic model. Figures 4 and 5 show four different instances of function blocks:

- `On/Off_Switch_A` is an instance of the OWL class `On/Off_Switch` which is a subclass of `Lighting_Sensor`. The function blocks implements the functionality of a basic light switch that can be used to switch a light "on" and "off" via the gate `on/off_switch_state_change`.

- `On/Off_Light_A` is an instance of the OWL class `On/Off_Light` which is a subclass of `Lighting_Actuator`. It represents the corresponding actuator that can be used in combination with a `On/Off_Switch` sensor via the PDP `on/off_light_state`.
- `Dimming_Switch_B` is an instance of the OWL class `Dimming_Switch` which is a subclass of `Lighting_Sensor`. It implements the functionality of a switch that can be used to dim a light, for instance, via its corresponding gate `relative_dimming_change`.
- `Dimming_Light_B` is an instance of the OWL class `Dimming_Light` which is a subclass of `Lighting_Actuator`. The light is dimmed when a binding to PDP `relative_dimming` is established.

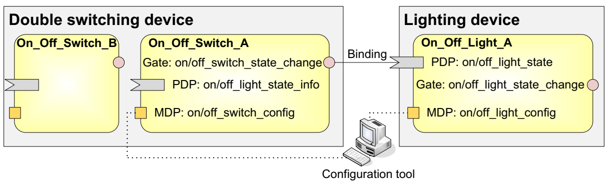


Fig. 4. Basic lighting function blocks including a (possible) binding

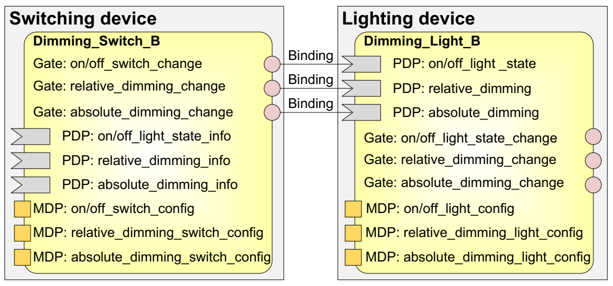

Switching device
Dimming_Switch_B
Gate: on/off_switch_change
Gate: relative_dimming_change
Gate: absolute_dimming_change
PDP: on/off_light_state_info
PDP: relative_dimming_info
PDP: absolute_dimming_info
MDP: on/off_switch_config
MDP: relative_dimming_switch_config
MDP: absolute_dimming_switch_config
Binding
Binding
Binding
Lighting device
Dimming_Light_B
PDP: on/off_light _state
PDP: relative_dimming
PDP: absolute_dimming
Gate: on/off_light_state_change
Gate: relative_dimming_change
Gate: absolute_dimming_change
MDP: on/off_light_config
MDP: relative_dimming_light_config
MDP: absolute_dimming_light_config

Fig. 5. Dimming function blocks including (possible) bindings

VII. Conclusion

In this work it was shown how ontologies can be used to facilitate an integration of heterogeneous building automation networks. A generic application model is proposed that abstracts technology-specific information and provides a generic view of the BAS. Therefore, this representation acts as single access point for configuration and maintenance tasks. Additionally, the reasoning capabilities of ontologies are exploited to allow the automatic calculation of gateway configuration data.

Future development intends to extend the generic application model and thus the ontology to span even more parts of the building automation domain. On the one hand this concerns other application domains than lighting, the integration of building automation specific vocabulary as middle ontology and further building related knowledge (e.g., more global concepts such as floors, rooms, doors, etc.). A valuable source for the latter can be the Industry Foundation Classes (IFCs) [17]. Although not actually organized in an ontology (yet), the IFCs already provide a data model structure of buildings and building elements.

On the other hand, the ontology shall be extended to cover other scenarios. A conceivable one is the definition of a whole BAS first only in the ontology and in an abstract way. From this representation, all configuration data, an automatic choice of appropriate devices, and best gateway placement can be derived and distributed to the different networks. An ontology can also help to improve security. Through a data capturing component attached, the regular behavior of a BAS (abstracted of technology specific details) can be logged. This is then a potential input for intrusion detection systems that monitor any derivation from standard system behavior.

Acknowledgment

The work presented in this paper was funded by FWF (Austrian Science Foundation) under the project P19673.

References

[1] "BACnet – a data communication protocol for building automation and control networks," ANSI/ASHRAE 135, 2004.
[2] "KNX specification," Version 1.1, 2004.
[3] "Control Network Protocol Specification," ANSI/EIA/CEA 709.1, 1999.
[4] *ZigBee Specification 2007*, ZigBee Alliance, San Ramon, 2007.
[5] C. Reinisch, W. Granzer, and W. Kastner, "Secure Vertical Integration for Building Automation Networks," in *Accepted at 7th IEEE Int. Workshop on Factory Communication Systems (WFCS '08)*, May 2008.
[6] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Dominque, *Enabling Semantic Web Services*, 1st ed. Springer, 2007, ch. 3.
[7] D. Brickley and R. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," Feb. 2004, W3C Recommendation 10 February 2004.
[8] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," Feb. 2004, W3C Recommendation 10 February 2004.
[9] S. Hegler and M. Wollschlaeger, "The Semantic Web in action: semantically enabled Device Descriptions," *5th IEEE Int. Conf. on Industrial Informatics*, vol. 2, pp. 1013–1018, June 2007.
[10] L. Sommaruga, A. Perri, and F. Furfari, "DomoML-env: an ontology for Human Home Interaction," in *SWAP 2005: Proc. of the 2nd Italian Semantic Web Workshop*, P. Bouquet and G. Tummarello, Eds., vol. 166, Dec. 2005. [Online]. Available: http://ceur-ws.org/Vol-166/34.pdf
[11] K. J. Charatsis, A. P. Kalogeras, M. Georgoudakis, and G. Papadopoulos, "Integration of Semantic Web Services and Ontologies into the Industrial and Building Automation Layer," *EUROCON, 2007. Int. Conf. on "Computer as a Tool"*, pp. 478–483, Sept. 2007.
[12] "BSR/ASHRAE Add. i to ANSI/ASHRAE Standard 135-2004," March 2008, second Public Review Draft.
[13] *KNX Specification - Part 7: Interworking Model*, Version 1.0, Konnex Association, 2002.
[14] *ZigBee: Home Automation Public Application Profile*, ZigBee Alliance, 2008.
[15] *IEC 61499-1: Function Blocks - Part 1: Architecture*, 1st ed., International Electrotechnical Commission, Jan. 2005, International Standard.
[16] W. Burgstaller, S. Soucek, and P. Palensky, "Current Challenges in Abstraction Data Points," in *IFAC Int. Conf. on Fieldbus Systems and their Applications*, 2005, pp. 40–47.
[17] "Industry Foundation Classes Specification - IFC2x Edition 3," International Alliance for Interoperability, Feb. 2006.