Semantic Integration in Building Automation
A Case Study for KNX, oBIX and Semantic Web Applications

Igor Pelesić *, Andreas Fernbach *, Wolfgang Granzer ‡, Wolfgang Kastner *

* Vienna University of Technology

Automation Systems Group

Treitlstraße 1-3, A-1040 Vienna, Austria

{ipelesic, afernbach, k}@auto.tuwien.ac.at

‡ NETxAutomation Software GmbH

Maria Theresia Strasse 41, A-4600 Wels, Austria

Wolfgang.Granzer@NETxAutomation.com

Building Automation Systems (BAS) in cooperation with smart grids show great potential for enhancing the energy efficiency of buildings. Smart buildings should consume the most of their energy when there is enough of renewable energy available and postpone or reduce consumption otherwise. The intelligent bidirectional interaction from devices on the demand side the whole way up to power distributors is hindered by a great heterogeneity of BAS, which is caused by the various devices, protocols and technologies produced by different vendors [1]. The open Building Information eXchange (oBIX) [2] is an open standard which offers a technology independent access to BAS and eases integration on a syntactical level by abstracting from different control protocols and network technologies. Still, the data exchanged does not necessarily bear the same semantics and thus requires further interpretation. In order to reach the goal of a comprehensive integration of different BAS technologies, this semantic incompatibility has to be resolved by enabling the distribution of semantically consistent data by various diversified devices. Overcoming this limitation would make it possible to share data across a distributed BAS that is compatible in terms of semantic aspects and to implement a common knowledge base [3] which in turn will enable the realization of more comprehensive automation scenarios that support the objective of improving the energy efficiency of buildings.

The aim of this work is to provide a semantic interoperability layer on top of an oBIX integration layer having KNX as a BAS representative. Starting from a set of use cases describing typical home and building automation scenarios, an ontology shall be designed which will eventually allow to exchange semantically consistent data across various BAS. The intention is to use the Web Ontology Language (OWL) [4] for this task. An extension or integration of existing ontologies from the building automation sector should be considered. In a second step, the oBIX standard shall be enriched with meta-data annotations, whereby the semantic of these annotations should be taken from the created ontology in order to allow consistent data

across various devices. Further, a transformation from oBIX to OWL needs to be performed. Not only static data like, e.g., the location of a sensor should be considered for the transformation but also run time data like current sensor values. For this task, an XSL transformation (XSLT) [5] is used. In order to evaluate the intended concepts, a prototype is currently developed by the Automation Systems Group of TU Vienna. In addition, an oBIX server interface is developed by NETxAutomation as an extension for their existing NETx BMS Server which integrates KNX and other building automation technologies. To this aim, oBIX contracts following the new KNX Web Service specification [6] need do be defined and implemented. The semantic interoperability prototype, which uses the NETx BMS Server as a data source, will be equipped with oBIX client functionality. It will maintain a common knowledge base as a Resource Description Framework (RDF) [7] triple store. The BAS ontology provides the necessary concepts therefore. In order to synchronise data between the triple store and the NETx BMS Server, a semantic web crawler will be developed which recurrently searches for devices and updates the RDF Triple Store with fresh data transformed from the oBIX interfaces of the underlying layer. The prototype shall also provide means to query and update the data from the RDF triple store via a SPARQL endpoint [8].

1 Introduction

The integration of different BAS technologies allows increasing both the energy efficiency of buildings and inhabitant comfort. Web Services (WS) provide a technology independent way of accessing and representing data and thus represent a promising approach to ease integration of heterogeneous systems and technologies [9] into an Internet of Things (IoT) [10]. The term IoT basically refers to a network of connected *things* such as sensors, consumer electronics, smart phones. The fundamental objective of interconnecting *things* is to empower them to create a preciser model of their context thus allowing them to provide services which react intelligently to dynamic changes within their environment [11].

The smart grid is considered as one of the most important applications of the IoT [12]. The core element of the smart grid is a bidirectional communication between energy producers and consumers. It is expected that smart grids will change the paradigm of energy distribution from a centrally controlled system to a system where the control is distributed to various objects. This should allow a more precise monitoring and controlling of the system which should result in less energy loss and higher efficiency [13].

Still, the interconnection of *things* based on WSs can only be seen as an intermediate step towards the fulfillment of the IoT vision. On this level, heterogeneous devices are enabled to interact via a formally specified application protocol, reasoning about the meaning of the exchanged messages requires a human interaction tough. In order to empower the collaboration of autonomous devices and agents, a common understanding between these *things* has to be established. This requires the utilization of Semantic Web technologies.

Semantic interoperability allows different agents connected to the IoT to interpret the exchanged data unambiguously and further empowers automated communication between *things*. Unambiguous data descriptions that are interpretable by machines and software agents are a major driver for automated information exchange within the IoT. SPARQL queries and semantic

reasoning could be used to solve problems related to knowledge extraction, data abstraction and discovery of resources [11]. Thus, by achieving semantic interoperability, autonomous distributed devices and agents would be enabled to collectively answer questions respectively perform actions like:

- Is every switching actuator of a distinct floor in "off" state?
- How many rooms of a building are occupied?
- Which lamps in a building have exceeded a distinct operating time?
- Turn on hallway lights in a distinct building part!
- Turn on all electrical boilers of a site having a distinct offset between current temperature and setpoint!

This work is structured as follows. In Section 2, the oBIX standard will be introduced. Section 3 provides a short introduction on the KNX Web Services standard. Next, in Section 4, the Semantic Web and its technologies will be presented. Section 5 describes the proposed Semantic Interoperability Layer for oBIX (SILo), followed by a short description of the proof of concept implementation. Finally, the results of this work are evaluated and some ideas for further work are presented.

2 Open Building Information Exchange (oBIX)

The oBIX specification is released and maintained by the Organization for the Advancement of Structured Information Standards (OASIS). The purpose of oBIX [2] is to empower communication between heterogeneous devices by abstracting from their actual hardware and low-level protocols they use. It provides a common and standardized interface that models a device to a set of datapoints allowing to access these via Web Services. A similar WS based approach is also possible via OPC UA [14] and BACnet/WS [15].

Every accessible oBIX object is identified by its URI [16] and the according information is exchanged in XML or JSON format over HTTP, which makes the information available to every Web browser. Essentially, oBIX provides the means to enable and improve the M2M communication. oBIX complies very well to the REST paradigm. A RESTful service is characterized through a set of principles such as resource-orientation, identification of resources, a uniform interface and stateless requests which are all essential to oBIX as well.

2.1 Object Model

oBIX provides a flexible and extendable object model. The common base primitive of this model is the object abstraction *obix:obj*. An exemplary *obix:obj* implementation is shown in Listing 1.

Every further object type like *real, int, str* is an extension of the base object. Any object type can contain further objects, i.e., the concept of composition (*has-a* relationship) is supported. The extendability of the model is based on the concept of *contracts*, which act as templates for inheritance (*is-a* relationship). Contracts additionally define properties for each object type such as default values and attributes attached to it. Properties supported by all object types are *name, href, is, null, val, ts*.

Further properties, e.g., *min, max, displayName, unit, range* that provide meta-data about objects are named *facets*. The *unit* facet supports most relevant SI-units and the *range, min*

Listing 1: *obix:obj* Example

```
<obj href="/devices/light_switch/" is="obix:Point">
  <bool name="value" href="value" val="false" writable="true"/>
</obj>
```

oBIX Request	HTTP Method
read	GET
write	PUT
invoke	POST
delete	DELETE

Table 1: oBIX-HTTP mapping [17]

and max facets limit the range of values which an object may store. The oBIX standard contains a set of core value objects, each of them storing a different value type.

2.2 Contracts

In order to group different object instances sharing common properties, oBIX introduces the concept of *contracts*, which are comparable to classes in object oriented languages. Contract definitions are templates which are expressed as simple oBIX objects and can be referenced by their URI using the *is* attribute.

Contracts are proper for modeling inheritance relationships in oBIX. Even multiple inheritance is supported. By defining a type, it is possible to assign default values to its instances and agree on the semantics of an object across different vendor systems, e.g., an *obix:Alarm* instance has the same object structure on different vendor systems and implicitly provides information about an alarming condition. The object structure is explicitly defined by a contract, whereas the reasoning about its semantics usually demands human interaction.

The concept of contracts is simple and flexible and allows to introduce new abstractions without inserting new syntax elements to the standard.

2.3 Networking

The communication in oBIX conforms to a client/server paradigm, where a client sends service requests to a server that is handling these requests. The supported service requests are *Read, Write, Invoke and Delete*. There exist different protocol bindings for these atomic operations as REST [17], SOAP [18] and Websockets[19]. The mapping of oBIX service requests to HTTP request methods is shown in Table 1 according to [17]. Different encodings such as XML, JSON and EXI are provided for the oBIX standard.

In order to allow a client to keep track of real time information, e.g. a temperature sensor value, the oBIX *watch* mechanism is presented. The client creates an *obix:Watch* object where it registers all the datapoints it is interested in and gets informed if any of these get updated. Using bindings where its not possible to push events from server to client, e.g. HTTP, the client

has to continuously poll the server for updates, whereas using the Websocket binding allows the server to directly push events to the client in case of changes on registered datapoints [2].

3 KNX Web Services

The KNX specification is about to be extended with a KNX Web Service specification [6] which defines a standardized interface that allows to integrate KNX networks with other IT systems like e.g. the IoT. The interconnection between KNX devices and other IT systems is enabled by the introduction of a *KNX Gateway* as shown in Figure 1. The *KNX Web interface* of the *KNX Gateway* has to support at least one of the following protocols: oBIX, OPC UA, BACnet/WS. It allows Web clients to read and modify data within the KNX network. The *KNX Network access* interface is responsible for the communication to KNX devices. The *KNX information model* specifies the structure of the input model that is used to represent the *KNX Network* within the *KNX Gateway* [6].

The *KNX information model* is based on the *KNX Tag vocabulary* which specifies a set of tags and their relations to each other. It is an extension to the already existing ETS object model and determines the static structure of a KNX network whereas the real-time values are accessed via the KNX network interface. The presented model allows additional sources of information as input to the *KNX information model* which could provide required information missing in the ETS, e.g., semantic annotations. The *KNX information model* allows to specify entities by assigning them tag-value pairs. Tags with a *null* are named *marker* tags and define a is-a relationship whereas the tags ending with '*Ref*' are reference tags pointing to other entities [6].

The specification provides a mapping from the *KNX information model* to each of the supported WS technologies such as oBIX.

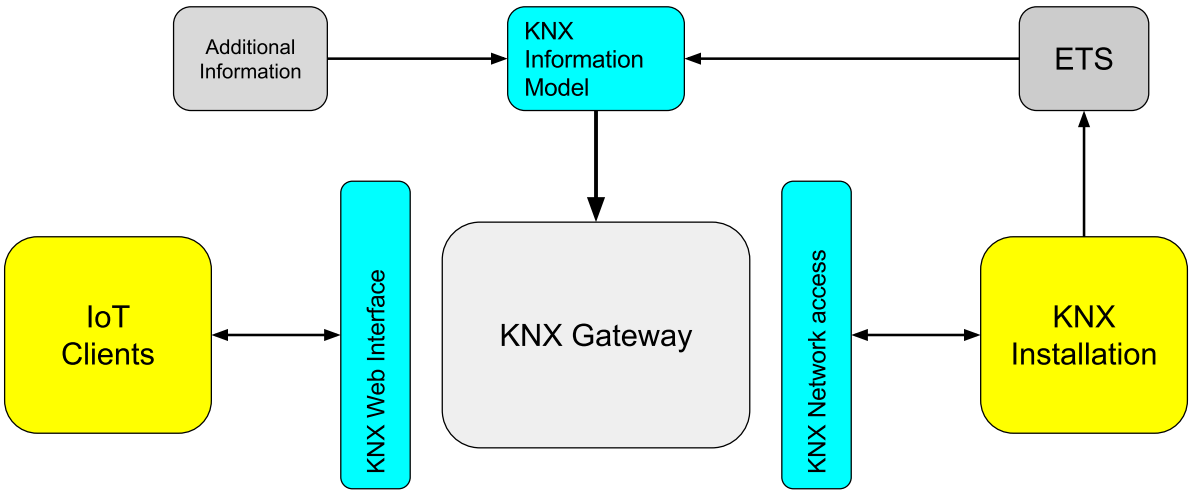

Figure 1: KNX gateway (adapted from [6])

Listing 2: KNX Gateway oBIX response

```
<obj name="example" href="/installation/example/" is="/knx/Installation"
    displayName="Example">
  <list name="views" href="views" of="obix:ref␣/knx/View">
    <ref name="view_heating" href="view_heating" is="/knx/View"/>
    <ref name="view_alarm" href="view_alarm" is="/knx/View"/>
  </list>
</obj>
```

4 Semantic Web

The Semantic Web provides the means to categorize and classify items and to reason about the relationships between these items. The idea of the Semantic Web as presented by Berners-Lee in 2001 [20] is to extend the current Web, where most of the available data is designed for humans, and to improve the structure of the data by giving the provided information a well defined meaning thus empowering advanced H2M and M2M cooperation opportunities. This provided meaning in which items are logically structured and connected enables the interoperability of systems [21].

The W3C has developed a set of technologies and languages to share meaning across the Semantic Web. The usage of IRIs [22] in order to uniquely identify items is essential to the Semantic Web. Basically, an IRI is an extension of a URI as it allows a wider range of UNICODE characters. As IRIs have a global scope, anyone can reference resources associated with them [21]. This allows to semantically reuse or extend existing concepts and to model relationships amongst them.

The Resource Description Framework (RDF) [23] is a framework for expressing information about resources which can be anything like documents, people, abstract concepts. It is designed for scenarios where data should be processed by applications instead of displayed to humans. RDF allows to make statements about resources. The statements or triples consist of a *subject, predicate and object*. *Subjects* and *objects* are resources. The *predicate* or *property* models the relationship between the resources and is always directed from a *subject* to an *object*. Triples can be visualized as a directed graph as shown in Figure 2 and they are usually stored in an RDF store. RDF supports a variety of serialization formats, e.g., Turtle [24], N-3 [25].

RDF makes statements about resources, but it does not allow to make any statements about the nature of the resources or what they stand for. In order to enable the classification of resources, the RDF Schema [26] language was presented. The RDF Schema introduces the notion of *class* expressed through the type *property*, which allows to build hierarchies of classes, subclasses, properties, subproperties. Restrictions on types are expressed through the *domain* and *range* properties. RDF Schema is a primitive ontology language [27]. Ontologies are collections of information which formally define relations among terms [20]. As both RDF and RDF Schema are rather restricted in their expressiveness, the Web Ontology Language (OWL) [28] was introduced. It is designed to model knowledge about things, groups of things and relations between them, which can be exploited by computer programs. OWL ontologies consist of classes, properties, individuals and data values. Classes define a concept while the prop-

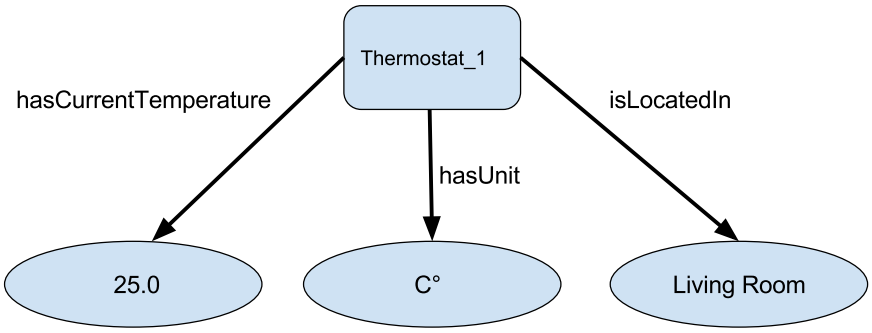

Figure 2: RDF graph visualization

erties model the relations between these concepts. Individuals are the instantiations of these concepts. The main exchange syntax for OWL is RDF/XML. Additionally to the features already supported by RDF Schema, it provides the means to express equality/inequality of classes, to specifically exclude membership to classes (disjoint classes), to apply cardinality restrictions on properties etc. OWL distinguishes between datatype properties and object properties. The range of datatype properties is always a datatype such as string, date. Object properties always point to another resource object. An important aspect of the Semantic Web is the possibility to search for semantic data. Therefore, W3C has provided the RDF Query Language (SPARQL) and SPARQL protocol [29]. This allows to query and even update RDF stores. SPARQL provides an HTTP and SOAP binding which allow to remotely execute SPARQL queries. In many aspects, SPARQL is comparable to SQL used with relational databases. The basic primitives of SPARQL queries are the *triple patterns* which are similar to normal RDF triples, but instead of the *subject, predicate* or *object* they contain a variable placeholder prefixed with the character '?'. Combining multiple *triple patterns* which all have to be satisfied is called a *group pattern*. Group patterns are enclosed by curly brackets [30]. As already mentioned, it is also possible to update RDF stores with a SPARQL update [31]. Updates contain a delete and insert clause specifying which triples to delete and insert respectively.

5 Semantic Interoperability Layer for oBIX (SILo)

In order to enable semantic interoperability that allows to exchange semantically consistent data unambiguously between different machines on basis of oBIX, a transformation of oBIX resources is required. The data presented at the oBIX interface is syntactically formalized but the meaning of the representation is not implicit. Semantic Web technologies provide the means to model the meaning implicitly. Therefore, a transformation of the oBIX representation to an OWL ontology is proposed. This ensures that the static structure of the BAS as provided at the oBIX interface is semantically well defined. This provides the means to use the resulting ontology as a common vocabulary for autonomous agents, execute SPARQL queries and gain new insights through semantic reasoning.

Nevertheless, the actual run time data of BASs like sensor and actuator values require an additional consideration. The transformed ontology depicts the state of a BAS at the moment

of the transformation. An energy efficient operation of BASs which satisfies the requirements related to comfort necessitates an access to run time data. Therefore, a synchronization mechanism needs to be implemented which ensures that the data presented by the OWL ontology are always up-to-date.

The proposed Semantic Interoperability Layer for oBIX (SILo) as shown in Figure 3 is tailored to the oBIX REST binding [17] syntactically encoded in XML. It provides a SPARQL binding that allows to query and update the values of the BAS in control. The binding to oBIX is based on HTTP. It provides the means to read the actual data at the oBIX interface and further to write values of the BAS by exchanging REST calls between the oBIX server and the SILo implementation.

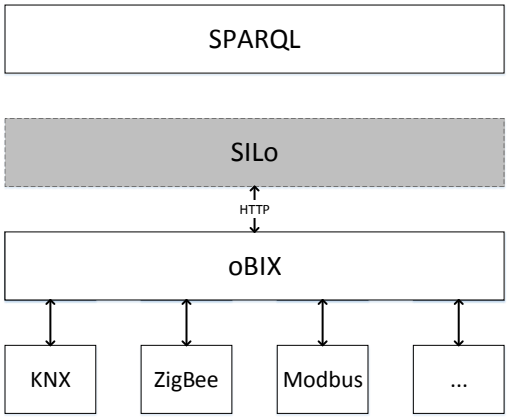

Figure 3: SILo stack

As both the oBIX source and the resulting ontology can be encoded in XML, the transformation is based on XSLT. Due to the flexible object model of oBIX, it can be concluded that an approach as presented in [32] including a mapping of the XML Schema Definition Language (XSD) [33] schema to an ontology is not useful as it would allow only to model the formal syntactical model of oBIX. This concept does not provide the means to model server specific oBIX *contracts* which provide explicit semantic information extremely useful to the transformation. It further can be argued that every specific oBIX server implementation, as long as there is no standardized oBIX interface, will require a customized transformation implementation regardless of the used ontology.

As the oBIX standard does not provide the means to model semantic information implicitly, it is up to server developers to communicate such. Therefore, it cannot be generally expected for such information to be provided by the oBIX server. If required semantic information is missing, a transformation is only possible when the semantic information is obtained from a different source. Commonly, providing semantic information requires a human interaction and is considered as a necessary part of the transformation engineering process which needs to be adapted to different oBIX server implementations. The higher the gap in the level of abstraction between the oBIX server implementation and the targeted ontology the more semantic information needs to be provided in order to allow a transformation, i.e., if the oBIX server delivers information on the level of datapoints and the targeted ontology models the BAS at the same level, no or little additional semantic annotations will be required. On the other hand, if the ontology assigns

datapoints to devices an information will be required as to define which datapoints should be grouped together to model a device.

As a result of the REST paradigm which is a core concept in oBIX and the supported reference feature, it might be necessary to traverse the complete oBIX object tree in the course of the transformation process in order to obtain all the required information. A specific WS call might return only the relevant data to this specific resource whereas information about its child objects might be provided as references. These child references have to be resolved by separate WS calls.

5.1 Model Transformation

On the grounds of the provided arguments, a generic transformation concept as depicted in Figure 4 is proposed. According to that, the transformation is performed in a three step process which is described in the following:

1. *Traverse and Combine* - In this step, the oBIX server object tree is traversed via multiple WS requests and the responses are combined into a *complete oBIX document*, which provides a complete view on the oBIX representation of the BAS.
2. *Annotate* - This optional step allows to introduce semantic data annotations to the *complete oBIX document*. This can be omitted if the oBIX server of interest is already providing the required information. The result of this step is a *complete semantic oBIX document*. The semantic annotations should be attached to required resources via the oBIX contract mechanism.
3. *Transform* - In a final step, an XSL transformation is performed resulting in the desired OWL ontology. It is recommended to preserve oBIX URIs as unique identifiers for OWL individuals during XSL transformation.

5.2 Data Synchronisation

The SILo implementation has to be designed in a way that the data that is provided at its SPARQL interface is up-to-date in order to ensure an adequate and reliable access to the BAS. It is assumed that the data available from the oBIX server represents a current state of the underlying BAS. Therefore, the task of the synchronisation mechanism is to merge differences between the data provided by the oBIX server and the data presented by the SPARQL interface within a reasonable time frame.

If data values are changed at the SILo SPARQL interface, the internal representation, i.e., the OWL ontology has to be updated. Additionally, the affected values at the oBIX server have to be updated in a timely manner. A single SPARQL update might require several HTTP requests to be transmitted.

One possibility to keep the OWL ontology up-to-date is to poll the data provided by the oBIX server recurrently. However, this approach does not scale well as a possibly large number of sensor and actuator values would produce a high network load even if the values have not changed. Because of that, the utilization of the oBIX *watch* mechanism is proposed. This approach still requires polling due to the fact that HTTP does not allow the server to contact the client, but reduces the network load substantially as only updated data values are transmitted. The oBIX Websocket binding [19] would obviate the usage of the watch service as it allows

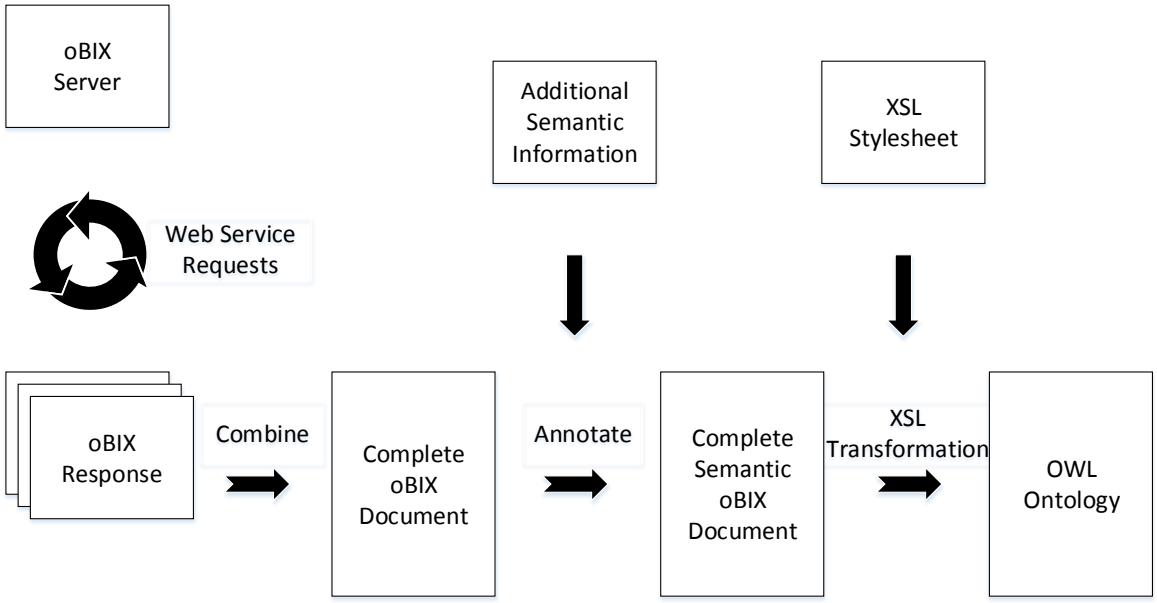

Figure 4: Transformation concept overview

the server to contact the client on demand. As soon as an updated value is provided by the oBIX watch service, the internal representation needs to be updated accordingly. The polling interval between successive *pollChanges* calls should be adjustable as to allow different control scenarios to be realized.

6 Implementation

The prototype implementation of the SILo is intended as a proof of concept. The prototype basically consists of two artefacts, an XSL document which allows to transform the oBIX model to an OWL ontology and a Java based Semantic Web crawler. The ThinkHome [34] [35] Energy and Resources ontology was chosen as target ontology as it allows to depict relevant parts of a BAS precisely. Some additional concepts were introduced to the target ontology in order to ease the implementation of the prototype. The knowledge base and the SPARQL interface of the crawler are based on Apache Jena [36], a Java framework for the Semantic Web.

Within the prototype implementation, the NETx BMS Server from NETxAutomation acts a gateway that is used to exchange data and information between the devices of the BAS. As shown in Figure 5 the NETx BMS Server provides different interfaces to a various number of systems and technologies. In addition to open standards like KNX, BACnet and Modbus, application specific interfaces to proprietary systems (e.g. hotel management, access control systems, ...) are supported too. Especially this diversity of available interfaces provides the

opportunity to realize complex and enhanced uses cases which cannot be done using a single technology. The datapoints as well as the available meta-data are represented within the NETx BMS Server in a transparent view that is independent of the underlying technologies. Via the newly developed oBIX interface, the Semantic Web crawler can access this view. In addition to the standard oBIX services (e.g. read and write requests), oBIX watches for push notifications are supported by the oBIX interface too.

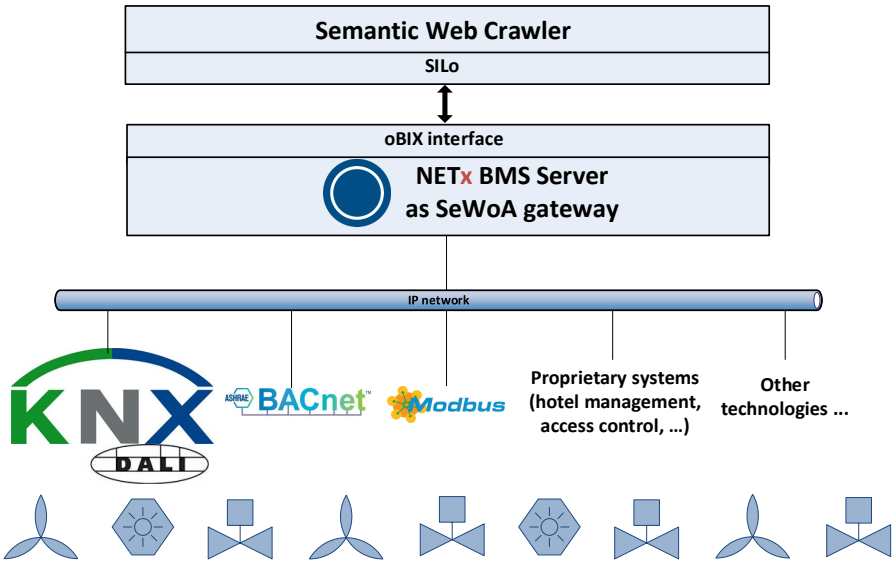

Figure 5: NETx BMS Server as SeWoA gateway

The architecture of the Semantic Web crawler is depicted in Figure 6. The crawler traverses the oBIX tree, resolves all found oBIX references and generates a complete oBIX document. As the oBIX interface of the NETx BMS Server adheres to the KNX Web Services specification which provides a model that is an extension of the ETS export model, it is possible to semantically annotate the BAS directly in the ETS by assigning meaningful names to KNX groups. Therefore, an additional enrichment of the data provided at the oBIX interface of the NETx BMS Server is not required. Nevertheless, the crawler provides the means to annotate the complete oBIX document with semantical meta-data to be used during transformation. The resulting complete semantic oBIX document is eventually transformed to the desired ontology by utilizing XSLT. The transformed ontology is stored inside the Apache Jena RDF triple store.

The crawler handles the data synchronisation between the oBIX interface and the internal knowledge base during runtime. In order to allow a scalable implementation, the oBIX watch service is utilized to register all devices of interest as well as their datapoints and to poll for changes. Potential changes of data on the oBIX interface are merged into the RDF triple store. In case of a SPARQL update, one or more REST calls are sent to the oBIX server in order to perform the desired action.

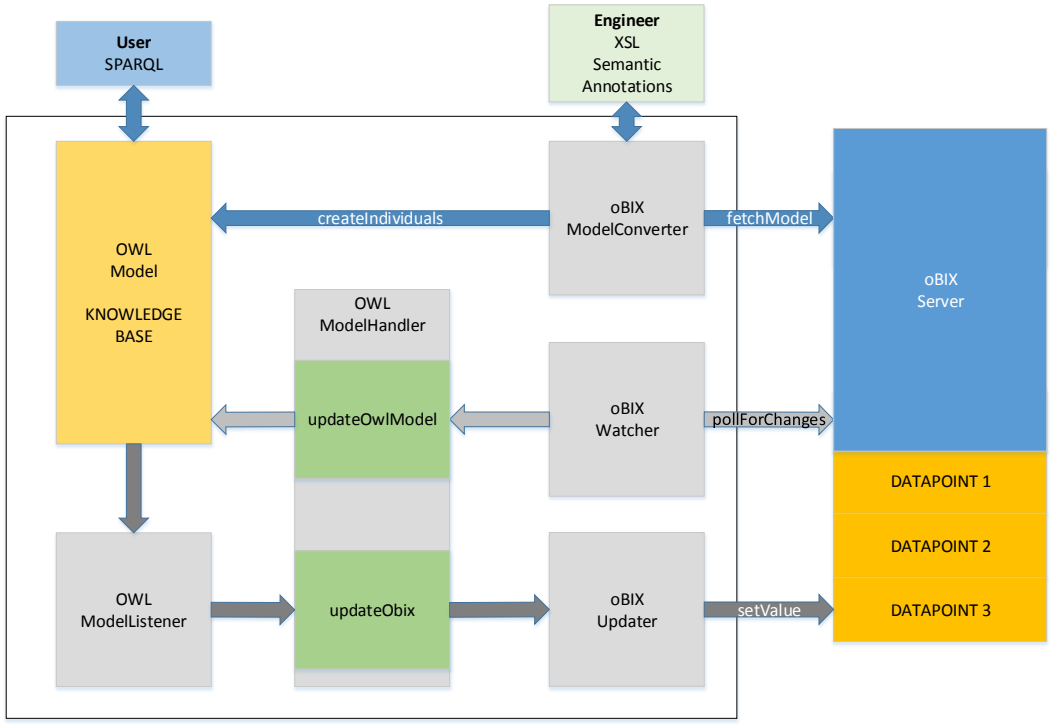

Figure 6: Semantic Web crawler architecture

7 Conclusion

The proposed SILo provides the possibility to dynamically map an arbitrary oBIX server representation to an OWL ontology, which can be used as a common vocabulary for distributed autonomous agents. This allows to increase the context-awareness of involved agents and thus contributes to a more energy efficient control and operation of buildings. The presented transformation process cannot fully be automated due to the flexibility of the oBIX standard, as distinct oBIX implementations will require a customized transformation. However, the KNX Web Services standard proposal allows to implement a reusable transformation for oBIX servers adhering to this specification, like the NETx BMS Server. An engineering effort might still be required due to a lack of semantical information.

Additional bindings for SILo like RDF/XML or Java besides the presented SPARQL binding would increase the interaction possibilities and ease further integration of devices. The transaction management of SILo requires further investigation as this is seen as essential to a distributed control mechanism. Security and safety aspects, as well as the error handling, also deserve further examination.

Acknowledgement

This work was funded by FFG (Austrian Research Promotion Agency) under the projects “Secure and Semantic Web of Automation” (FFG IKT der Zukunft, P840206) and “Kognitive Regelstrategieoptimierung zur Energieeffizienzsteigerung in Gebäuden” (FFG Energieforschungsprogramm, 848805).

References

[1] W. Kastner, L. Krammer, and A. Fernbach, “State of the art in smart homes and buildings,” 2014.

[2] “OBIX Version 1.1. OASIS Committee Specification.”

[3] W. Kastner, A. Fernbach, W. Granzer, and M. Jung, “KNX and the Semantic Web of Automation,” in *Proceedings of the KNX Scientific Conference*, Nov. 2014.

[4] “OWL2 Web Ontology Primer (Second Edition),” W3C Recommendation, 2012. [Online]. Available: http://www.w3.org/TR/2012/REC-owl2-primer20121211

[5] “XSL Transformations (XSLT),” W3C Recommendation, 1999. [Online]. Available: https://www.w3.org/TR/xslt

[6] “KNX System Specifications - Web Services,” KNX Association, 2016, Draft.

[7] “Resource Description Framework (RDF),” W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/rdf-primer/

[8] “SPARQL Query Language for RDF,” W3C Recommendation, 2008. [Online]. Available: http://www.w3.org/TR72008/REC-rdf-sparql-query-20080115

[9] W. Granzer, W. Kastner, and P. Furtak, “KNX and OPC UA,” in *Konnex Scientific Conference*, Nov. 2010.

[10] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[11] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, “Semantics for the internet of things: Early progress and back to the future,” *Int. J. Semant. Web Inf. Syst.*, vol. 8, no. 1, pp. 1–21, Jan. 2012. [Online]. Available: http://dx.doi.org/10.4018/jswis.2012010101

[12] O. Hersent, “KNX,” in *The Internet of Things: Key Applications and Protocols*. Wiley Publishing, 2011.

[13] N. Bui, A. P. Castellani, P. Casari, and M. Zorzi, “The internet of energy: a web-enabled smart grid system,” *IEEE Network*, vol. 26, no. 4, pp. 39–45, July 2012.

[14] OPC Foundation, “OPC Unified Architecture (UA),” 2008, [Last accessed 26-March-2015]. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua

[15] ASHRAE, “Proposed Addendum C to Standard 135-2004, BACnet,” 2004, [Last accessed 26-September-2016]. [Online]. Available: http://www.bacnet.org/Addenda/Add-2004-135c-PR1.pdf

[16] “Internet Engineering Task Force. RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax, January 2005.” http://www.ietf.org/rfc/rfc3986.txt, [Last accessed 26-September-2016].

[17] “Bindings for OBIX: REST Bindings Version 1.0. Edited by Craig Gemmill and Markus Jung. 14 September 2015. OASIS Committee Specification 01.” [Last accessed 26-September-2016]. [Online]. Available: http://docs.oasis-open.org/obix/obix-rest/v1.0/obix-rest-v1.0.htm

[18] “Bindings for OBIX: SOAP Bindings Version 1.0. Edited by Markus Jung. 14 September 2015. OASIS Committee Specification 01.” [Last accessed 26-September-2016]. [Online]. Available: http://docs.oasis-open.org/obix/obix-soap/v1.0/obix-soap-v1.0.html

[19] “Bindings for OBIX: WebSocket Bindings Version 1.0. Edited by Matthias Hub. 14 September 2015. OASIS Committee Specification 01.” [Last accessed 26-September-2016]. [Online]. Available: http://docs.oasis-open.org/obix/obix-websocket/v1.0/obix-websocket-v1.0.html

[20] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[21] N. Shadbolt, T. Berners-Lee, and W. Hall, “The semantic web revisited,” *IEEE intelligent systems*, vol. 21, no. 3, pp. 96–101, 2006.

[22] “Internationalized Resource Identifiers (IRIs). January 2005. RFC.” http://www.ietf.org/rfc/rfc3987.txt, [Last accessed 26-September-2016].

[23] Resource Description Framework (RDF), “W3C Recommendation,” 2014, [Last accessed 26-March-2015]. [Online]. Available: http://www.w3.org/TR/rdf-primer

[24] “RDF 1.1 Turtle. W3C Recommendation. 25 February 2014,” https://www.w3.org/TR/turtle/, [Last accessed 26-September-2016].

[25] “RDF 1.1 N-Triples. W3C Recommendation. 25 February 2014,” http://www.w3.org/TR/n-triples/, [Last accessed 26-September-2016].

[26] “RDF Vocabulary Description Language 1.0 RDF Schema,” W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[27] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2004.

[28] OWL 2 Web Ontology Language Primer (Second Edition), “W3C Recommendation,” 2012, [Last accessed 26-March-2015]. [Online]. Available: http://www.w3.org/TR/owl2-primer

[29] SPARQL. Query Language for RDF, “W3C Recommendation,” 2008, [Last accessed 26-March-2015]. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query

[30] D. Allemang and J. Hendler, *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[31] “SPARQL 1.1 Update. W3C Recommendation. 21 March 2013.” http://www.w3.org/TR/sparql11-update/, 2015, [Last accessed 1-October-2016]. [Online]. Available: http://www.w3.org/TR/sparql11-update/

[32] H. Bohring and S. Auer, “Mapping XML to OWL Ontologies.” *Leipziger Informatik-Tage*, vol. 72, pp. 147–156, 2005.

[33] “XML Schema Definition Language (XSD) 1.1. W3C Recommendation. 5. April 2012,” https://www.w3.org/TR/xmlschema11-1/, [Last accessed 04-October-2016].

[34] C. Reinisch, M. J. Kofler, and W. Kastner, “Thinkhome: A smart home as digital ecosystem.” Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies, 2010, pp. 256–261.

[35] M. Kofler, “An ontology as shared vocabulary for distributed intelligence in smart homes,” Ph.D. dissertation, Vienna University of Technology, 2013.

[36] Apache, “Jena,” https://jena.apache.org, 2016, [Last accessed 26-March-2016]. [Online]. Available: https://jena.apache.org