Masaryk University
Faculty of Informatics

Semantic BMS: Semantics-Driven Middleware Layer for Building Operation Analysis in Large-Scale Environments

Doctoral Thesis

Adam Kučera

Brno, Fall 2017

Masaryk University

Faculty of Informatics

Semantic BMS: Semantics-Driven Middleware Layer for Building Operation Analysis in Large-Scale Environments

Doctoral Thesis

Adam Kučera

Brno, Fall 2017

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Adam Kučera

Advisor: doc. RNDr. Tomáš Pitner, Ph.D.

Acknowledgement

This thesis would have never been finished without collaboration and support of many people who were part of my life during my PhD studies.

I would like to thank my parents for supporting me in my decision to enroll doctoral study. My girlfriend for pushing me towards the finish and listening to my thoughts and complaints. My friends for taking me out occasionally and making me enjoy some fun.

I am grateful for other (both current and former) members of the Lasaris lab that created inspirational environment for me.

I feel great appreciation to my boss at the Department Of Facility Management, Petr Glos, for the long-term support of my study activities and the change to become a part of the Building Management Project at the Masaryk University.

The biggest gratitude goes to my supervisor Tomáš Pitner for the guidance, tolerance, patience, respectful and friendly approach and uncountable amount of precious advice and discussions that were essential for my professional and personal development.

Abstract

Facility benchmarking and evaluation of facility performance are crucial tasks in reaching efficient, economical and sustainable facility operation. Modern buildings are equipped with building automation systems that contain vast numbers of various sensors. In comparison with other data sources, such as financial data, the sensor data are more detailed, less delayed, and more accurate. However, such systems lack convenient tools for data inspection, which limits their use in building operation optimization especially on large sites. The aim of the thesis is to overcome this issue and allow facility managers to use building automation data for facility benchmarking with minimal effort.

The thesis presents an ontology model enriching automation data with additional semantic information. The semantic model is created using the Web Ontology Language. The Semantic Sensor Network (SSN) ontology created by W3C Consortium is adapted and extended for the field of building operation analysis. The model is designed to be automation protocol independent and describes available data in a way that can be utilized during decision support tasks needed for building performance analysis, evaluation, and improvement.

The presented Semantic BMS project focuses not only on the semantic description of the data points but also on querying of the ontology model. Thus, the middleware layer is built on top of the semantic model. The Semantic BMS provides convenient interfaces that allow to gather semantic data about data point or easily select data points that satisfy required criteria without the need to fully understand specifics of the building automation systems or specifics of the ontology languages.

The Semantic BMS focuses on providing tools for user-friendly, flexible, and dynamic querying over the building automation data using criteria that are comprehensible for facility managers. The process of obtaining and inspecting key performance indicator data is significantly simplified. The proposed system allows facility managers to conveniently use BAS data for benchmarking and decision support.

Keywords

facility management, building automation systems, semantic sensor network ontology, decision support, data integration

Contents

1 Introduction

The profession of facility management secures various aspects of organization's operation that are not directly involved in reaching its primary goal (e.g. providing service to a customer or sell its products). One of the important tools of facility management is benchmarking and performance/efficiency assessment. The paper presents novel software system that allows to comfortably use sensor data for efficient facility benchmarking. The novel capability simplifies composition of complex queries over the proposed semantic model that links sensor data with the Building Information Model (BIM) databases.

Modern ("intelligent") buildings are equipped with variety of sensors and controllable devices (e. g. Heating, Ventilation, Air Conditioning –HVAC, security systems). The devices are integrated into the Building Automation System (BAS), also referred to as Building Management System (BMS).[1] The devices incorporated in BMS can be remotely controlled, monitored and queried. Individual information objects (such as current temperature in particular room measured by sensor) accessible in the BMS network are referred to as "data points" further in the thesis.

The building automation systems provide invaluable support in everyday operation of the facility. From the point of view of the facility management, the building operation is perceived in broader context, with a stress to long-term performance and efficiency, as illustrated in the Figure 1.1. Current building automation systems unfortunately are not generally suited for facility managers' needs.

The BMS supports an everyday operation of built environment, or more specifically, secures operation of equipment that controls the facility operation. Therefore, the BMS contains a large amount of

1. The distinction between the two terms is rather unclear and are considered synonyms in many cases. The term "BMS" stresses the user perspective (user manages building operation using the BMS). However, there is a certain ambiguity originating in the use of the term "management" that is not usually associated with electronic and computer systems. The term "BAS" stresses the technology perspective (building operation is automated by BAS). In the rest of the thesis, the term "BMS" will be preferred, as the topic of the research is focused on human interaction with the BMS. However, the term "building automation" is still used in specific contexts where it is more suitable.

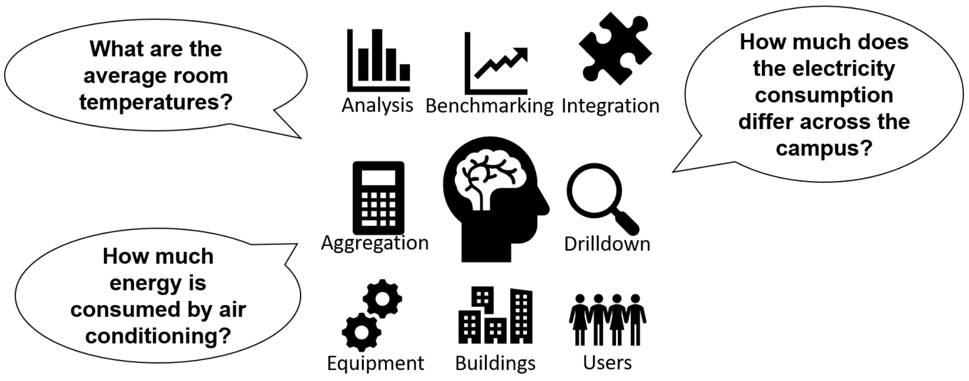

What are the average room temperatures?
Analysis
Benchmarking
Integration
How much does the electricity consumption differ across the campus?
Aggregation
Drilldown
How much energy is consumed by air conditioning?
Equipment
Buildings
Users

Figure 1.1: Facility managers' focus, concerns and tools.

precise, up-to-date and detailed data, which cannot be obtained any other way. The level of detail and precision of acquired data is usually significantly higher when compared to other possible data sources (e.g. financial data or surveys).

Benchmarking methods in management are covered in the EN 15221-7 standard. Requirements placed on Key Performance Indicators (KPIs) are summarized in [1]. Among others, the authors mention flexibility, the quantitative nature of the KPIs and simpleness of use.

BMS data satisfy the two former requirements. The simpleness of use is a downside of current BMS solutions. The aim of presented research is to improve simpleness of use of BMS data, thus rendering them suitable for facility benchmarking and operation analysis.

A facility manager should be able to query the system in similar manner to those examples:

- Show me which rooms on the second floor of A11 building had running AC units during last 8 weekends.
- Tomorrow morning, I want to receive report about electricity consumption in 5 minute intervals for those 4 buildings during this night.
- I want to know which devices influence temperature in office of person XY.

However, the BMS does not provide structured semantic information about the data points. This drawback prevents efficient querying of the data points for analytical purposes, as it is not possible to select and filter the data based on criteria such as the type of a source device, location of measurement or measured quantity kind. The second issue with the BMS data is the ease of retrieval.

This thesis proposes Semantic BMS Ontology and Middleware that aim to provide semantic description of the building automation systems. The middleware layer allows developers of business intelligence applications to effectively query the model and easily gather required building operation data.

The presented novel approach to facility benchmarking facilitates continuous improvement of the facility performance, efficiency and sustainability, as it provides an open and flexible architecture that enables detailed, precise and up-to-date building operation data for use in facility benchmarking. The querying capabilities are designed to support the Deming cycle management method (plan-do-check-act) in facility benchmarking which is naturally needed for meaningful benchmarking and is recommended by the EN 15221-7 facility benchmarking standard.

The thesis is organized as follows: The following Chapter 2 provides brief but complete overview of the research, covering a problem statement, methods, contributions and other essential information about the presented work. Chapter 3 provides an in-depth analysis of the problem solved. Chapter 5 contains an overview of related work. Chapter 4 introduces theoretical background of the research. Chapter 6 describes the proposed solution. In Chapter 7, applications of the resulting software artifacts are presented. Chapter 8 discusses usability of the presented framework and elaborates on its general suitability in the field of facility management. Chapter 9 briefly states direction of future work and final Chapter 10 summarizes and concludes this thesis.

2 Research overview

This chapter aims to provide readers with brief overview of the whole research, covering the essential aspects of research work in shortened form. Following sections of this chapter thus provide problem statement, objectives, methodology overview, discussion of results and contributions and overview of publications authored during the research. The first section is devoted to the motivation for the research – building automation system at the Masaryk University.

2.1 Motivation use case: Masaryk University

The University Campus in Brno-Bohunice (UCB) is an educational, research and development center designed to accommodate 5000 students and 1000 employees [2]. The construction of the UCB started in 2004 [3] and the first part has become operational in 2007. Since then, the UCB has grown to almost 35 buildings. Figure 2.1 displays the main entrance to the UCB.

Figure 2.1: The main entrance of the UCB. (Source: Management of the UCB)

The Masaryk University uses an in-house developed system for storing Building Information Model (BIM – see Section 4.2.1 for further information) data that uses a spatial database as a data storage.

The system is based on the ESRI ArcGIS software. The BIM database completely covers most parts of the UCB.

The project of the Building Management System that spreads over the whole Masaryk University (BMS MU) and its facilities started together with the construction of the first part of the UCB that was completed in 2007. Since then, all the newly built facilities at the UCB have been connected to the common BMS. Additionally, other buildings spread through the City of Brno and belonging to various organization units (Institute of Computer Science, Faculty of Economics, Faculty of Education, University Headquarters, Accommodation and Catering Services) were connected to the system at the occasions of replacing an obsolete building equipment with new units. Other projects under construction are planned to be connected to the BMS of MU in the next few years.

The core of the system is formed by devices and application servers that communicate over the BACnet protocol, which is briefly introduced later in this chapter. BMS of MU today contains over 1700 BACnet-enabled devices from more than 20 industry-leading vendors. The BMS of Masaryk university contains over 150 000 data points, from which over 30 000 are physical sensors and actuators (inputs and outputs of automation devices).

2.2 Problem statement

Data points of the BMS are described by their location in the network topology, not by the role that algorithms, sensors, and actuators fulfill in the building operation. In the case of the BACnet, which is one of the widely used, open, and standardized automation protocols, a data point is described only by a limited set of attributes. A data point representing a temperature sensor is identified by a network address of the device that reads the value, data type of the input (Analog input) and ID of the input within the device. Besides this data point identification, BACnet provides only several free-form string attributes such as Name or Description, which are intended to be easily readable by human operators.

The absence of structured semantic information prevents efficient querying of the data points for analytical purposes, as it is not possible to select and filter the data based on criteria such as the type of a source device, location of measurement or measured quantity kind. If the data from particular data points are required (e.g. electricity consumption for last month for each of the buildings on the site which will be later compared), the operators of the system have to manually gather the data point addresses by inspecting the building plans or user interface of the BMS.

The above-mentioned problem clearly emerges when operating large BMS system such as the BMS of the Masaryk University.

2.3 Objectives

To solve issues addressed in previous sections, additional software tools are needed. The aim of the research is to design and develop software system that provides complex and structured representation of building automation system installation. The model must be easily "query-able" to facilitate data selection based on enriched semantics. The software also must provide interface for accessing building automation data in feasible way, so they are available for easy use by analytical and decision support applications. The aim is to provide the missing link between the source systems (BMS, Building Information Model – BIM, see see Section 4.2.1 for further information) and end user analytical applications.

The BMS controls and monitors devices that are already described by the BIM, but the relation of the BMS data points and entities represented in the BIM is missing in currently available systems. Therefore, there is an area for great improvement of usability of the BMS data by linking them to the respective elements described by the BIM. The presented research thus provides a method for such linking and adds other necessary semantic annotations to the BMS data points.

When the BMS data are linked to their BIM "counterparts", they can be then used as another data source for the analytical systems that facility managers know and already use for benchmarking and decision support. They also provide an analytical framework for easy definition of various reports required by facility managers.

Development of front end applications lays outside the scope of the research. However, the goal is to support their development. Using proposed software tools, developers of end-user applications will be able to fully focus on front-end features (user interface, query definition wizards, integration with GIS services) and analytical features, and not on the core logic of data integration and retrieval. Advanced user interfaces are essential for the successful BMS data analysis and performance benchmarking, as they will remove the gap between the facility manager's strategy-level (economical) knowledge and operator's technical knowledge needed for gathering the data from the BMS.

2.4 Methods

To reach goals stated above, a middleware layer named Semantic BMS is introduced. The SBMS framework serves a connection between BAS and end-user applications and provides required semantics for the building automation data. This section summaries the most important decisions undertaken during the research, as well as general principles followed during the research process and software development.

Structured semantic model: The semantic information is modeled uses ontology language OWL (Web Ontology Language) that allows for complex description of objects and relations stored in a rigidly defined structure. The OWL provides all necessary devices to model semantics relations in the BAS and other related ICT systems. The OWL is based on the RDF (Resource Definition Framework), which brings support of the SPARQL query language.

Ease of integration: The architecture follows principles of the SOA (Service Oriented architecture) to facilitate integration with other systems. It provides complex, but easy-to-use RESTful interfaces providing BAS data and semantic information. Building on others: During the framework development, existing components were facilitated to the maximal extent. Additionally, the components are required to be Open Source and free software (under GPL, Apache, BSD, or similar license). This approach simplifies maintenance and improves extendibility of resulting software artifacts. Following components, technologies and framework were used:

- RDF – Metadata data model for conceptual description of information;
- OWL – Ontology language used for the semantic model definition. Allows to represent the model in the RDF format;
- SPARQL – Query language used with the RDF semantic model;
- SSN – Semantic Sensor Network ontology serves as a base for the developed ontology model, Provides key concepts related to the sensor measurement domain;
- Java SE 8 – programming language and framework;
- Apache Jena – framework for RDF manipulation and SPARQL engine;
- Jersey framework – reference implementation of RESTful server for Java;
- Bacnet4j – BACnet protocol stack used for use-case implementation of the framework.

Usability: Usability of the resulting framework is key consideration during the development. The model structure and API definitions are driven both by requirements coming from operation of BAS at Masaryk University and by guidelines for facility benchmarking outlined in EN 1552-7 standard. The evaluation of the results was performed both on use cases from the Masaryk University facilities and the EN standard.

Adaptability: One of the resulting artifacts is implementation of the framework for the BACnet protocol (as a part of the use case study). However, the framework provides general and abstract model of BAS and accounts for different users with different setups. It provides interfaces to adapt the framework for use with different automation protocols and archive solutions.

2.5 Contributions

The research presents novel semantic model that is adapted to the domain of building automation and addresses multiple differences between the domain of traditional sensor measurements (e.g. environmental monitoring), for which the SSN ontology was intended, and the field of building automation. The SSN was significantly extended to reach new requirements. The most notable difference is distinction of the source device of the measurement and the device that publishes the data to the network.

The model itself provides structured, rigid description of building automation data, that can be queried and result can be further machine-processed. Such complex model in not currently available.

On top of the semantic model, the middleware layer is built. It provides complex querying capabilities unavailable in current solutions. Flexibility of the APIs brings unprecedented ease of data selection to the domain of building automation.

Conversely, the Semantic BMS project brings building operation data with high level of detail into the field of facility management. Although solutions combining CAFM software systems and BAS data exist, they do not provide such complex semantic description of the data, limiting the analytical capabilities.

2.6 Research process & validation of results

Validation of results is provided by employing developed software artifacts into facility management processes. As a start line for definition of goals and requirements, experience from university campus operation and previous research projects were considered.

At the beginning, certain use cases were selected as templates for developed models and interfaces. Next, semantic model was developed with respect to requirements of such use cases. Development of semantic API followed the model completion. Methods of the API were designed to provide means to implement sample use cases. In more general view, ability to extract all available information about each and every element of the model steered the API development.

In later phases of development, the resulting framework was compared and evaluated against industry standard for benchmarking facility management (EN 15221-7) to ensure its general compatibility with common task in the target domain.

To prove suitability of the framework, sample use cases were selected both from the benchmarking specification and from established processes at the University campus and such processes were (re)implemented using the framework.

2.7 Publications

From the start of the research, several papers were published. In this section, brief commentary on the most significant publications will be provided. Appendix A provides a full list of publications together with the author's contribution and their significance.

All the publications mentioned here are directly related to the presented research, in most cases presenting preliminary or partial results or defining use cases or research objectives. However, aims of the research shifted as certain areas of research emerged as more complex and promising that previously thought by research team.

At the beginning of the research, the focus was on monitoring of building automation systems and complex event processing. Results related to this branch of research were presented in [4]. Then, the attention moved to application of statistical methods on building operation data to provide decision support for facility operation optimization. Results were presented in [5] and [6]. However, information gathered from performed analyses did not prove as sufficient for optimization of building performance.

However, during a preparation of previous papers, inefficiencies of data retrieval workflow in building automation systems were revealed. This finding lead to publication [7] that points out issues of performance analysis based on building automation data and proposes potential future applications of advanced analysis over building operation data.

In the next phase of the research, first iteration of the semantic ontology model was proposed and published in [8] and [9]. Developed semantic model proved its suitability for given problem, however numerous flaws in the design led to development of completely new, enriched version of the semantic model based on the SSN ontology. Resulting ontology was presented in [10].

An extended and differently aimed version of the paper describing the Semantic BMS ontology was presented in [11]. The paper focused stressed on the environmental aspects of the facility benchmarking and presented a broader context of the work. The conference paper will be a part of the ISESS post-conference proceedings once the will become available.

2.8 Summary

This chapter provided brief overview of this thesis. It covered the research motivation, problem statement, methods, main contributions, and evaluation process. Finally, It provided overview of published papers related to the research. In the following chapters, individual aspects of the research will be covered in larger detail.

3 Problem statement: Insufficient semantics in BMS

Traditional facility management information systems (FMIS), also known as Computer-Aided Facility Management systems (CAFM systems) support an every-day operation of a building on the operation level in areas such as maintenance, cleaning or waste management. On the strategic level, they are used for decision support (e.g. space management and planning, asset management) and benchmarking of building operation efficiency and performance (e.g. energy management). For the mentioned task, manually updated data, often of financial nature, are used. Detailed comparison of information systems in facility management is provided in the Section 4.2.1.

The BMS supports an everyday operation of built environment, or more specifically, secures operation of equipment that controls the facility operation. Therefore, the BMS contains a large amount of precise, up-to-date and detailed data, which cannot be obtained any other way. The level of detail and precision of acquired data is usually significantly higher when compared to other possible data sources (e.g. financial data or surveys), making the BMS data more suitable for certain types of benchmarks when compared to the data provided by traditional FMIS/CAFM systems. The Table 3.1 provides a comparison of analytical capabilities of the CAFM systems and BMS, which currently favors the CAFM systems for the decision support and benchmarking. Even though the data used by the CAFM does not provide as high level of detail as sensors and the data are often delayed (e.g. the energy consumption is gathered from invoices), the BMS does not provide analytical tools that could be used for decision support and operation analysis.

Facility managers could greatly profit from employing building operation data coming from the automation systems into their decision support and benchmarking tasks. However, the simplicity of use is a downside of current BMS solutions, as discussed in the rest of the chapter. The aim of presented research is to improve simplicity of use of BMS data, thus rendering them suitable for facility benchmarking and operation analysis.

Table 3.1: Comparison of the CAFM and the BMS capabilities for benchmarking.

	CAFM Systems	**BMS**
Used data	Financial	Sensor
Level of detail	Low	High
Data vividness	Delayed	Up-to-date
Analysis tools	**Complex**	**Simple**
Usability	High	Low

Data points of the BMS are described by their location in the network topology, not by the role that algorithms, sensors, and actuators fulfill in the building operation. In the case of the BACnet, which is one of the widely used, open, and standardized automation protocols, a data point is described only by a limited set of attributes. A data point representing a temperature sensor is identified by a network address of the device that reads the value, data type of the input (Analog input) and ID of the input within the device. Besides this data point identification, BACnet provides only several free-form string attributes such as Name or Description, which are intended to be easily readable by human operators.

The absence of structured semantic information prevents efficient querying of the data points for analytical purposes, as it is not possible to select and filter the data based on criteria such as the type of a source device, location of measurement or measured quantity kind. If the data from particular data points are required (e.g. electricity consumption for last month for each of the buildings on the site which will be later compared), the operators of the system have to manually gather the data point addresses by inspecting the building plans or user interface of the BMS.

In current environments, we can define two types of users – one group of users are facility managers who know which data they need for a building operation analysis, they know the context, but they are unable to get the data from the systems. The other group of users consists of building operators which have capabilities to get the data from the BMS (even if the task includes a large amount of "manual" work), but do not have enough time, knowledge, competence or authority to fully evaluate the building operation and make long-term decisions based on the results.

Current workflow for BMS data analysis is thus too complicated for a flexible data analysis. A responsible staff member (facility manager) asks building operators to get the needed data and agrees with them on the data output format. Building operator gathers the addresses of respective data points according to the request and then extracts the data for each of the data points. Next, a conversion to the defined format for business intelligence application follows. This process is not automated and has to be repeated every time the report is needed.

The above-mentioned problem clearly emerges when operating large BMS system. For the detailed information on the use case of the BMS of Masaryk University, refer to the Section 2.1.

Advanced applications with convenient user interface that will hide low-end aspects of the task (gathering the data point addresses, extracting the data from the database, conversion to the interchange format) will allow facility managers to obtain the data directly from the system without the need of human work of the building operators.

A facility manager should be able to query the system in similar manner to those examples:

- Show me which rooms on the second floor of A11 building had running AC units during last 8 weekends.
- Tomorrow morning, I want to receive report about electricity consumption in 5 minute intervals for those 4 buildings during this night.
- I want to know which devices influence temperature in office of person XY.

Currently, applications allowing to answer such complex questions are not available. Development of such applications is very demanding. The system has to cover each of the aspects of the task:

- Data retrieval;
- Definition of data semantics;
- Analysis;

- User interface and experience.

To accomplish the above mentioned tasks, experts from several fields are needed. The problem comes mainly with the two first points. They require expertise in the fields of building automation protocols and building technologies (e.g. HVAC devices) itself, which is not common among IT experts. The vendors of building automation systems focus mostly on development of a software that can be used for management, programming of the system and for an every-day operation of the building technologies, rather than on an analytical software.

Development of such complex systems is probably commercially unprofitable, because large sites that would profit from such systems are relatively rare today. At smaller sites, a sufficient data analysis can be performed by involving simpler approaches such as a manual report definition, an export of raw data (an analysis is then performed ad-hoc by end users) or by using purely financial data (i. e. invoices) for operation analysis.

We believe that large sites equipped with the BMS will emerge more and more frequently. The trend towards deeper and broader interconnection of building technologies is evident and will gain an importance in conjunction with other modern trends such as *Internet of Things* and *Smart Grids* or *Smart Homes*. However, a situation in the field of a "traditional" building automation and facility management is different when compared to mentioned new trends. Corporations and other large organizations are rather conservative when comes to equipping newly built facilities with modern technologies. Buildings are constructed for expected lifespan of multiple decades and investors lean to use of established and well known technologies, which on the other hand do not offer modern features.

As stated above, we identify two main issues that prevent developers from creating flexible and user friendly applications for a building operation analysis:

- Accessing the data in the BMS;
- Enriching the BMS data with a semantic information.

The following sections elaborates on possible approaches to overcome these issues.

3.1 Data Accessibility

Generally speaking, each application that accesses the BMS data needs to implement the building automation protocol stack (this is not necessarily true in a case of accessing only an archival data, as discussed further in the text).

The need of a specialized protocol stack differs the domain of conventional building automation from other fields, such as modern smart homes technologies or solutions for environment monitoring used in data centers (IP based temperature and humidity sensors). As an example of the different approach, today's popular electronics platforms such as *Arduino*, *.NET Gadgeteer* and solutions based on *Raspbery Pi* or *BeagleBone* boards, are usually equipped with an operating system (typically Linux) or are at least capable of communicating using the HTTP protocol, XML-based data exchange formats or other ways as required for the particular situation. Similar situation applies for IP based environmental sensors that often offer SNMP capabilities in addition to previously mentioned communication methods. On the contrary, many building automation devices are limited only to a communication using a specialized protocol (e.g. BACnet).

To facilitate application development, multiple protocol stacks are available. In the case of the BACnet protocol, both Open Source (e.g. *BACnet stack* [12], or *BACnet4j* [13]) and commercial (e.g. *SCADA Engine* [14]) for various programming languages can be used. Additionally, the BACnet network can also be accessed using other methods. Typical example of such approach is the Delta Controls ODBC driver (now discontinued product) that allows to query and control building automation devices using SQL.

Protocol stacks aim to cover a complete functionality of the protocol. However, the complexity of the protocol stack quickly becomes unnecessary for basic tasks such as a data reading, which is the core functionality needed for front-end analytical applications. This use case would benefit from an abstraction layer that would hide networking-related aspects of the protocol and provide a limited number of simple data access methods.

Furthermore, using the protocol stack in an application requires to deploy the application with direct access to the network of the BMS. Since the BMS devices usually communicate in dedicated and separated network due to the security and management reasons, this necessity puts additional requirements on the used hardware (e.g. need of multiple network interfaces) and the system configuration. To overcome this issue, some form of "proxy" server can be placed into the BMS network and connected to the internet as well. The proxy server then forwards a communication between the BMS and the outside world. In case of the BACnet protocol, the specification includes definition of the BACnet Web Services that aim to provide message format definition using well established Web Services technology [15]. However, the introduction of the *BACnet Web Services* does not solve the problem of the protocol complexity.

In the case the application does not require access the data directly and aims to the processing of historical data instead, the problem becomes significantly simpler as historical data can be stored in a relational database. For the BACnet protocol, database-based storage solutions for archival data are available at the market. At the Masaryk University, Delta Controls Historian application server [16] is used together with the Microsoft SQL Server database.

3.2 Missing Semantics

Currently available solutions for the BMS do not contain mechanisms for describing data points in the means of the semantics of the data or relation to the environment. Each data point is identified only by its network address according to the BACnet protocol specification. Name of the data point is designed to be understandable by human operators and is thus unsuitable for machine processing. In the BMS, the semantic information is maintained only on the user interface level.

The user interface of a typical BMS consists of a number of visualization views that display certain parts of the system (temperatures on one floor, scheme of air conditioning unit. The views contain graphic, text and dynamic data from the BMS (on-line values are presented on a static background). The building operator is able to gather semantic information either from the unstructured name of the data point, or from the relations between different components in the user interface. In case of a temperature sensor measuring room temperature in par-

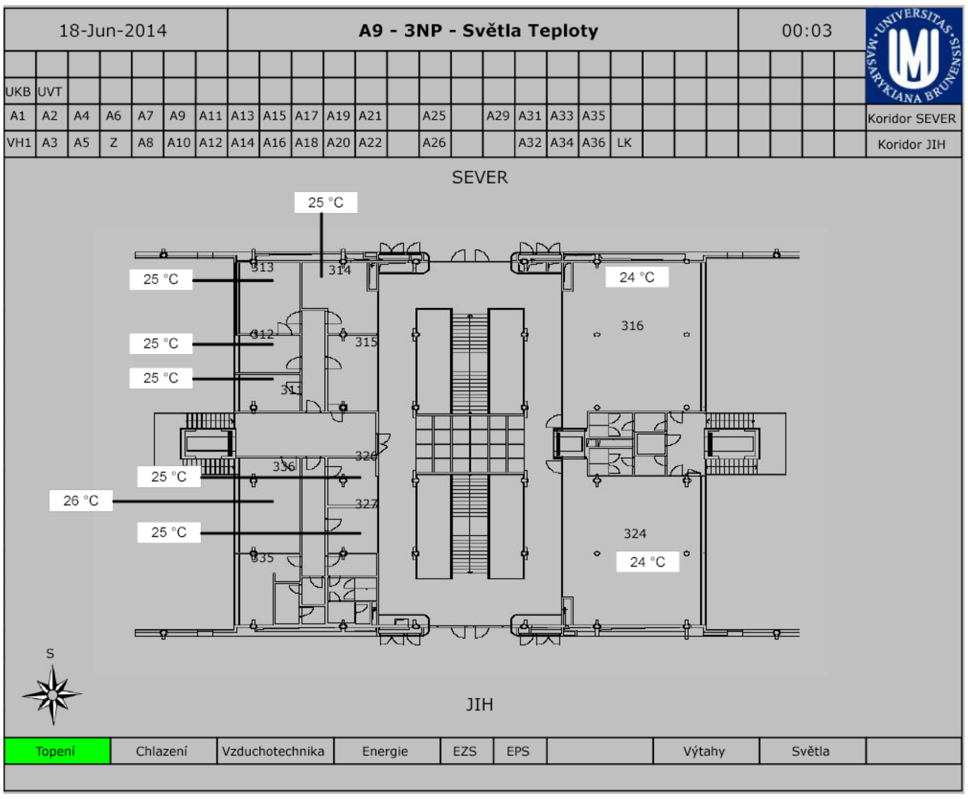


Figure 3.1: BMS visualization user interface

ticular room (see Figure 3.1 – in Czech), the meaning of the data is gathered from following observations:

- *Meaning of the data* – The heading of the view states that this particular screen presents a temperature and lighting data;
- *Building* – The heading of the view contains building name (“A9”);
- *Floor* – The heading of the view contains floor name (“3NP”);
- *Room* – Each temperature is connected with the respective room either by a line or by placing the value inside the room boundaries in the floor plan;
- *Type of the source device* – Not available in the user interface.

Gathering a semantic information from visualization views is obviously ineffective if not even impossible. Some information is not present in the views and the view definition usually cannot be queried in a meaningful and effective way. Considering that the semantic information often needs to be extracted in a real time during the query execution, semantics on the level of a user interface is unsuitable for machine processing.

Thus, semantic information such as a location of the sensor or a measured quantity must be added manually for the purposes of analysis. This drawback makes a data analysis inflexible and prolongs a data analysis workflow. An ad-hoc approach to system integration is sufficient for experimental purposes such as testing new methods of data analysis and provides valuable results concerning evaluation of different approaches, but prevents a deployment of proposed methods for a routine operation. The complexity of the system and a lack of semantic information about gathered data in fact prevents facility management staff to perform a data analysis routinely and on a regular basis.

The need of an ad-hoc approach in order to link data from various sources also hides complex relations between data points, devices, environmental variables and other factors that influence a building operation. To fully understand a behavior of building systems, we need to utilize a large number of indicators. Some of them are not measured at all, some of them are not stored in an archival database or they are available in an information system that is not freely accessible. On the other hand, some additional data about a building operation are gathered and stored by BMS. Size and complexity of the system although hides certain relations between an operation data from various sources which limits the human operators in meaningful usage of such data.

A typical example of such hidden relation is a link between state of pumps, motors and valves of a central heating unit placed in a utility room in a basement of a building. A heating unit is monitored and controlled by a different part of the BMS than a local air conditioning unit in the lecture room. The local air conditioning controller has ability to control the valve on a heating radiator in the room. Actions of the local AC controller are thus influenced by a current setup of the central heating unit (if the central heating is off, controlling of the valve on the radiator in the room does not influence the room temperature at all – there is no hot water in the radiator). The relation of central heating unit setup and an operation of the local AC unit can't be observed from the archive data itself. Some information about complex relations can be derived by examining regulation algorithms. Other relations are not described in the BMS itself at all because they are determined by physical installation of various pieces of equipment in the building. In the case of multiple central heating units, the BMS does not contain information concerning physical plumbing - we can't tell which radiator valve is connected to which central heating unit. However, this information is stored in the BIM database. Thus, integration of the BMS and the BIM will allow to examine such relations.

3.3 Summary

This chapter provided in depth analysis of the problems that prevent development of analytical applications that employ data from building automation systems. The main issues are missing (suitable) semantics of the automation data and difficulties encountered by the developers when accessing the automation data. The following chapter presents broad overview of the whole research domain.

4 Theory: Fundamental concepts and principles

The presented research topic lays on an intersection of multiple disciplines. It aims to utilize devices and methods of computer science in the discipline of facility management. Furthermore, it encompasses use of cyber-physical building automation systems.

This section further elaborates on key concepts and principles used to achieve the research goal. These serve as founding stones of the research, allowing to reach desired goals. The concepts are described in detail in the rest of this chapter.

4.1 Facility management

The facility management as a term describing a profession ensuring building operation emerged in the 1970's. As stated in the introduction, The *International Facility Management Association* defines facility management as "a profession that encompasses multiple disciplines to ensure functionality of the built environment by integrating people, place, process and technology." [17] Particular aspects of the facility management are covered by international standards, such as EN 15 221 (Parts 1 to 7) [18] or ANSI/BOMA Z65 [19] (see Section 4.2 for further details).

4.2 Efficiency evaluation in facility management

One of the main concerns of the facility management is evaluation of an organization's operation performance and efficiency. Since the aim of the thesis is to provide tools for an evaluation of a building operation, the following section provides an overview of benchmarking methods and approaches in the facility management.

The facility management is undergoing a long-term process of a standardization. In the context of the European Union, the domain is covered by the *EN15221 – Facility Management* [18] standard issued by European Committee for Standardization.

For the field of the space management, the EN15221-6 provides guidelines for a measurement of dimensions, a categorization and an evaluation of a "building performance" (e.g. a ratio of an overall area of a building and an area that can be leased). In the USA, The *Building Owners and Managers Association* (BOMA) provides a different set of standards known as the *ANSI/BOMA Z65 – Standard Methods of Measurement* [19], which focuses on the area of space management and specifically on the measuring methods and an efficiency evaluation.

Benchmarking is a subject matter of the last part of the European standard listed as EN15221-7. The focus is put mostly on processes or services which can be easily outsourced, such as a cleaning or maintenance. Evaluation of *Key Performance Indicators* (KPIs) then becomes essential for enforcing Service License Agreements witch service providers. The document also covers a benchmarking in other areas, including energy management.

An energy efficiency of facilities gains a significant interest from research groups, authorities and an administration. In the European Union, the *Directive 2002/91/EC* on the energy performance of buildings [20] introduced the Energy Performance Certificate (EPC) that provides an A to G scale for a rating of an energy efficiency of a facility.[21][22] However, the rating is based solely on an evaluation of used materials, an equipment, and a design – it do not reflect an actual energy consumption during a facility operation. The EPC is thus criticized for its inaccuracy (e.g. [23], [24], [25]).

Considering an energy consumption evaluation during a facility operation, different approaches (sets of KPIs) are proposed [26], [27]. In [28], *Complex Event Processing* tools are used for an energy consumption analysis and decision support for an industrial environment (i.e. a factory). Analysis of possible usage of energy consumption benchmarking in the environment of Masaryk University is provided in [29] (in Czech).

Typically, the data needed for the evaluation are gathered from energy costs or other financial indicators. If there were tools for a BMS data extraction available, evaluation of the KPIs would be simplified, more precise (BMS provides greater level of the detail than an invoice for the whole building or site) and even completely new KPIs could be defined, taking into account other aspects such as temperature oscillations in the facility.

4.2.1 Information systems in facility management

The technology part of the facility management have experienced great change as information and communication technologies (ICT) are widely used in different aspects of a facility operation.

We can distinguish several systems and/or data sources that can be utilized in order to support and simplify tasks of facility management staff.

The *Computer aided facility management* (CAFM) software covers wide range of activities that are not part of an organization's core business, but are necessary for its operation – space management, hoteling and reservations, preventive and an on demand maintenance, service desk, portfolio management, and accounting. Computer Aided Facility Management (CAFM) systems facilitate tasks such as assigning employees to rooms, logging maintenance plans, requests and tasks, or energy consumption data.

In addition to support of an everyday facility operation, the CAFM provides tools for an analysis and an evaluation of the facility operation performance in a long-term perspective. CAFM systems offer advanced analytical tools for efficiency and performance evaluation of organization's operation based on financial (energy consumption), spatial (occupancy planning), and technical data (maintenance). The CAFM systems are considered the "core" Facility Management Information Systems (FMIS). However, there are at least two other information systems that are closely related to the facility management.

The *Building Information Model* (BIM) is designed as a database containing all the available and necessary information regarding the whole lifespan of a facility (building) from its construction to its demolition, covering spatial aspects of the facility (e.g. dimensions, floors, walls), physical aspects (e. g. used materials), as well as installed equipment (e.g. sensors, heaters, air conditioning devices, plumbing). During the construction, the BIM provides an environment for information interchange between all participating parties (e. g. a customer, contractors, and authorities). As a data interchange format and a data model, the *Industry Foundation Classes* standard (ISO 16739:2013 [30]) can be used. During the building operation, the BIM database should be updated to reflect all modifications of the facility. If properly maintained, the BIM serves as an source of a precise information that can be used for the building operation efficiency analysis, operation cost estimates (e.g. costs of cleanup services or equipment revisions), as well for planning future modifications of the facility.

There is a close relation between the BIM and the CAFM systems described above. The CAFM naturally needs to be connected and synchronized with the BIM database to ensure a consistency of data. The CAFM offers its analytical capabilities for the data stored in the BIM and also provides an interface for updating the BIM database.

The next field of a facility operation where the ICT is largely involved is the building automation. In the field of building construction, the term *intelligent building* is used mostly for facilities equipped with remotely controlled and monitored interoperating devices. Other definitions of the term are discussed in [31]. A wide variety of electronic systems is usually installed in modern buildings – e. g. a regulation and an automation of heating, ventilation and air condition (HVAC), a security system, an access control system, a closed-circuit television (CCTV), a fire alarm system, a lighting control. The building systems can be integrated into an integrated environment which provides a common user interface for all the systems, as well as other services such as an alarming or a data archiving. The ability to control and to monitor the systems remotely from a common user interface reduces a number of a staff needed for ensuring an operation as well as demands on an expert knowledge of system operators. A common interface for all the systems also allows them to cooperate.

The integration of building systems is usually accomplished by establishing a common communication bus that uses one of the integration protocols, e.g. *BACnet* (ISO 16484-5, [32]), *LonWorks* (ANSI/CEA-709.1-B, [33]), *KNX* (ISO/IEC 14543, [34]), or *MODBUS*. The bus interconnects workstations of operators and application servers (web user interface, archive database servers, monitoring systems) with the backend devices such as *programmable logic controllers* (PLCs), central units of security systems or with protocol gateways and translators. The described infrastructure (workstations, servers, gateways, common bus) is referred to as a *Building Automation System* (BAS) or *Building Management System* (BMS) in further text.

Table 4.1 provides a comparison of BIM, CAFM and BMS systems with respect to the data they provide and tasks they facilitate. The BIM provides mostly static data describing the built environment and installed devices. Generally, the data in the BIM are changed in case of "uncommon" occasions (e.g. during the building's reconstruction). The CAFM system supports an every-day operation of the organization. The CAFM systems facilitate rather dynamic data, including information about human resources, organization structure or cash flow (e.g. invoices from energy vendors). The BMS provides volatile sensor data and actuator settings that vary during the day in order to ensure correct operation of the building.

Table 4.1: Comparison of information systems in the field of Facility Management.

	BIM	**CAFM System**	**BMS**
Meaning	Building Information Model	Computer-Aided FM System	Building Management System
Scope	Built environment Locations Devices	Space management Furniture Maintenance Energy management	Building automation Remote monitoring Remote control
Data volatility	Low: Generally static data	Moderate: Continuous updates by human users	High: Automatically collected sensor data
Typical usage	Reference documentation	Analysis and reporting	Online controlling and monitoring

4.3 Cyber-physical systems

Traditional fields of information technology and computer science (such as algorithms, networking, databases, data mining, software engineering) focus on processing data – pieces of information that are virtual with no direct representation in the physical world. Manipulating such data does not immediately affect processes happening in the physical world surrounding a computational unit.

However, as information technologies pervade into other fields of engineering, the assumption of non-influence ceases to be true in many cases. So-called cyber-physical systems is a class of information technology systems which operations directly, immediately, and intentionally influence processes in the physical world and vice versa

– the system allows for sensing phenomena occurring in a physical environment.

Computing units of cyber-physical systems are equipped with sensors and actuators that ensure interaction with an environment by transferring digital signals used in data processing to electric signals on output interfaces where they can be used for controlling e.g. relays or other electric components. Electric signals from sensors (e.g. thermistors) are converted to digital signals using A/D converters in a similar manner as in opposite direction.

Digital data can be naturally processed by an algorithm. A typical use of a cyber-physical system is a scenario where a device periodically scans its sensors, processes received data and sets actuators to desired values, thus regulating and controlling assigned process in physical world – air conditioning function, assembly line operation, etc.

Cyber-physical systems find applications in all types of process automation, in environmental sensing and measurements, transportation industries (e.g. automotive, aerospace), energetic industry and various other fields.

One of the instances of cyber-physical systems are building automation systems described further in this chapter.

Many cyber-physical systems are considered so-called *critical infrastructures*, which are introduced in the following section.

4.4 Critical infrastructure

The term *critical infrastructure* is not defined by any technical similarity of systems it covers. Instead, it defines such systems that ensure "fluent" functioning of a society. Critical infrastructure is defined by national governments based on their importance to the state executive.

Typical examples of critical infrastructure system are electricity generation and distribution, gas distribution, water supply, transportation systems, public health, or telecommunications.

In the European union, general guidelines for identification, designation, and protection have been issued in the (European) Council directive 2008/114/EC ([35]). On the national level, the directive was implemented as the Act No. 430/2010 Sb ([36]). that amended the Act No. 240/2000 Sb. (Emergency Management Act).

The council directive defines critical infrastructure in point (a) of the Article 2 in [35] as follows: "Critical infrastructure means an asset, system or part thereof located in Member States which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions."

The EU directive does not define specific critical assets (see recital 5 in the preamble of [35]). However, the national (Czech) legislation names multiple key critical assets in the First part, Head I, § 2, subparagraph m:

- Energy industry;
- Water resource management;
- Food industry and Agriculture;
- Health care;
- Transport industry;
- Information and communication systems (ICT);
- Finance and currency;
- Emergency management services;
- Public administration.

Gradually, many parts of critical infrastructure have become heavily dependent on ICT systems (both "traditional" and cyber-physical). That makes them vulnerable to an entire class of new threats that can be described as cyber-attacks. In the case of cyber-attack, an attack vector goes through an ICT system instead of direct attack that harms a physical infrastructure.

A well-known example of successful cyber-attack on a cyber-physical system is the application of the Stuxnet computer virus, harming Iranian nuclear program (see [37], [38]). The Stuxnet attacked the PLCs (programmable logic controllers) that issued faulty signals (changed rotation speed of a connected motor), resulting in damage of controlled uranium-enriching centrifuges. Although the nuclear program is not a typical part of critical infrastructure, this use case illustrates the vulnerability of ICT-aided infrastructures.

Vulnerabilities of critical infrastructures were addressed in *Framework for Improving Critical Infrastructure Cybersecurity* (see [39]) issued by U.S. administration. The framework proposes methods for securing such systems.

The next section describes building automation systems which are not usually considered to be part of critical infrastructure due to its relatively small scope – installations are local and separated, there is no country-wide automation network. However, they are vulnerable to similar attacks as critical infrastructure part that facilitate cyber-physical systems.

Furthermore, elaborate attacks on automation system can gain attacker either confidential information, access to other ICT systems, or even physical access to restricted areas of facility. All mentioned attack goals can lead to direct threat to critical infrastructure.

In conclusion, critical infrastructures provide both reason to secure building automation systems and methods to securing them.

4.5 Building automation & management systems

Building automation and management systems (BAS/BMS) are one of the several ICT systems used in facility management. However, they significantly differ from other facility management systems mentioned earlier in this chapter. The building automation installation is usually cyber-physical distributed system. Consequences of the cyber-physical nature are discussed in section 4.3. Description of the BAS/BMS architecture follows.

Figure 4.1 presents the simplified scheme of the BMS infrastructure as implemented at the Masaryk University. However, the topology is identical or at least very similar for all deployments of the BMS using the BACnet or even other automation protocols. The lowest level of the system consists of specialized programmable logic controllers (PLCs) that directly control and monitor "subordinate" systems (mostly HVAC devices). Lower level PLCs are connected to a serial bus (usually RS-485). The middle level consists of integration PLCs that connect the low level bus to the standard computer network using the Ethernet and the TCP/IP. On an application layer, devices communicate using a common automation protocol, such as the BACnet in the case of the BMS of Masaryk University. Various protocol translators and gateways are also connected to the BMS, translating communication between other protocols and the BACnet. The top level of the system consists of front end application servers providing a user interface to the BMS, archiving services for the operational data, or surveillance.

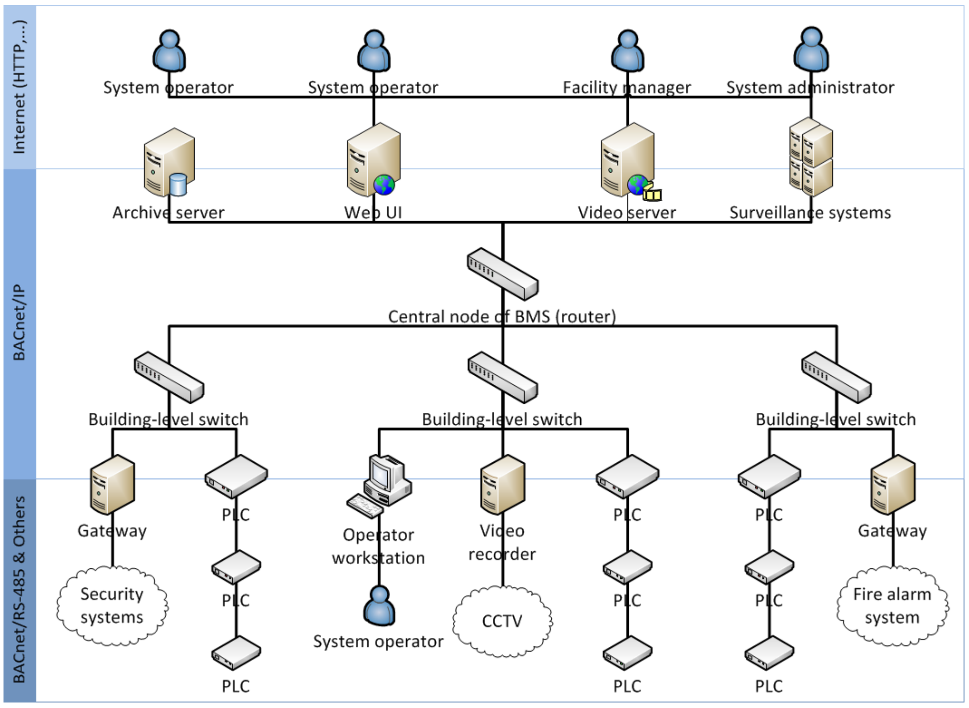


Figure 4.1: BMS structure overview

Alternative approach to a distributed infrastructure with a common communication protocol is based on a central gateway that ensures translation between different protocols. Such gateway usually employs the *OLE for Process Technology* (OPC) as standardized way to integrate systems. However, among other drawbacks of OPC, a centralized approach limits a scalability of the BMS. [40]

Devices connected to the BMS publish various data to the network. The values represent observations (measurements), control signals for subordinate systems, inner states of the devices, or other information about the system state and configuration. Each published value is identified by its network address. In the further text, the network address will be referred to as a *data point*.

4.5.1 BACnet protocol

The BACnet protocol was proposed and is currently maintained and developed by the *American Society of Heating, Refrigerating and Air-Conditioning Engineers* (ASHRAE). The development started in the 1987 and the first version of the BACnet protocol was introduced in 1995 [41]. Updated versions of the protocol are periodically introduced, last revision being 135-2012 (ver. 1, rev, 14) published in 2013 [42]. The protocol specification was accepted as the *ISO 16484-5* standard [32] in 2003. In 2014, the ISO specification was updated to reflect the changes in the latest ASHRAE revision.

The protocol specification defines services, that describe messages interchanged between devices in the network (e.g. reading or writing values, notifications, network discovery services), and object types that define a data representation and its organization in the devices. Each object consists of properties that store data of a particular data type. Values of the properties identify an object instance and describe its state. The concept of the object representation of the data largely corresponds to the concept of classes and objects as known from the Object Oriented Programming.

Besides the protocol specification, additional standard (*ANSI / ASHRAE 135.1*, *ISO 16484-6* [43]) under the name *Method of Test for Conformance to BACnet* was established. The standard describes the device testing procedure which results in the *Protocol Implementation Conformance Statement* (PICS). Since only the core services and objects of the BACnet protocol are required for the successful operation in the network, the PICS document issued for each device type defines the exact subset of the protocol specification that is implemented in the specific device. This allows a fast and cost effective development of simple single-purpose devices. Standardized testing procedure and conformance statement allows customers to evaluate device capabilities before purchase.

On the other hand, the protocol specification is also open for vendor-specific extensions to the protocol. The vendors are able to define their own object types and properties using data types defined in the protocol specification. Although the extensions can be used only for communication between products of the same vendor, this capability allows to use implementation of the BACnet protocol stack for transmission of the data that are not described in the protocol specification (i.e. various configuration and settings that vary for each type of device and thus they are not standardized).

Standardization of the protocol, together with the above described properties (a limited amount of mandatory functions and a possibility of vendor specific extensions) leads to the wide use of the protocol among building automation manufacturers. The ASHRAE registers 769 vendors from all over the world in their database [44]. Although the Vendor IDs are provided cost free and we can expect that not all the registered vendors actually develop BACnet devices, the BACnet protocol is well established and implemented in devices of both small companies and industry leading corporations (Honeywell, Siemens, ABB, or Johnson Controls).

Because of the wide spread, we see the BACnet protocol as an ideal platform for development of the middleware prototype that can be at least partially used in different environments without modifications.

4.6 Modeling semantic information using ontologies

For modeling of semantic information, various techniques can be used, including "traditional" normalized relational databases. For the OLAP processing, methodologies such as the *Dimensional Fact Model* (see [45]) have been developed. However, specialized frameworks used for describing relations in a real world have been proposed, designed, and implemented. This section provides overview of approaches that are related to methods of the presented research – specifically, modeling using ontologies. However, this section covers the topic starting from underlying trends and more general technologies.

4.6.1 Linked data and semantic web

The term *linked data* is coined by Tim Berners-Lee, head of the World Wide Web Consortium (W3C). It denotes publishing data in a format that is suitable also for machine processing. It facilitates the Resource Definition Format (RDF, see subsection 4.6.2 below) format as a data model representation.

Although the presented research does not follow linked data principles, as nature of building automation data differs from those available publicly on the internet, the linked data approach is presented here, as it serves as driving force to development of other technologies that are directly used in the SBMS framework.

The linked data requires that each described object is annotated using RDF and identified by unique Uniform Resource Identifier (URI). Additionally, according to the linked data principle, all the URIs should be using the HTTP protocol and they should be possible to dereference (point to an existing resource). Additional relations to other resources should be provided at the URI destination. The data at the URI destination should also be provided in structured format (i.e. RDF), creating a large graph of public machine-readable information recursively.

Similar term, the open linked data, adds condition that the data are published under an open license. Application of the (open) linked data principle builds the semantic web.

The semantic web is an approach presented by W3C as the next evolution step of the World Wide Web, moving from sharing of documents to sharing of structured data. It is a result of world-wide application of the linked data approach.

The semantic web utilizes number of technologies and standards that are needed to achieve the desired result of a large mesh of structured information and which were defined and developed by W3C during their efforts towards the semantic web.

4.6.2 Resource Description Framework

Resource Description Framework (RDF) is a World Wide Web Consortium (W3C) standard (see [46]) for metadata modeling. The framework provides a method for sharing, publishing, merging, and relating of structured (or semi-structured) data regardless of underlying data model.

The information in the RDF model consist of *triples* representing relations between objects. The triple obviously consists of three following elements: subject, relation type, target object. The objects and subjects are denoted as *resources* and they are assigned an Uniform Resource Identifier (URI). The model is then set of triples that form a graph, resources acting as nodes and relations forming edges of the graph.

A RDF model can be stored either in serialized form of RDF/XML or other human-readable formats (e.g. Turtle), or in a form of triples in so-called triple store. The triple store serves as dedicated ontology repository. A repository can be implemented as a noSQL database (considered a native triple store) or in traditional relational database.

An RDF model can be queried using the *Simple Protocol and RDF Query Language* (SPARQL). The RDF modeling is well supported by available APIs, triple stores and query engines, such as *Apache Jena* (see [47]) providing tools for RDF manipulation and storing.

4.6.3 Taxonomies and ontologies

The terms *taxonomy* and *ontology* (also, a *vocabulary* can be added to the list) describe similar approaches of modeling semantic information, differing in level of complexity and rigidness of resulting models. However, the distinction is not strictly defined and can be perceived differently depending on the context[1].

The *taxonomy* is usually understood as a hierarchical structure assertions. It defines a category tree of objects, providing one relation – denoted as *is-a*, *parent-child*, *class-subclass*, or *generalization-specialization* – which can represent several slightly different types of relations, depending on a context. That makes taxonomies less formal, less rigid, less complex, and less complicated than ontologies.

The term *ontology* denotes the model of relations between elements in a domain.

In general, an ontology defines concepts (classes of objects), individuals belonging to concepts, and properties modeling the relations between different individuals. An ontology allows to restrict the domain (object) and range (subject) of the property (relation). This additional layer of relations between objects of different classes distinguishes ontologies from simpler taxonomies. As an example of an ontology language is the RDF Schema (RDFS). Ontologies defined using the RDFS are also called RDF vocabularies.

1. As discussed in `http://www.ideaeng.com/taxonomies-ontologies-0602` and `http://broadcast.oreilly.com/2010/02/what-is-the-difference-between.html`

A typical use case for a taxonomy is a search engine (commonly document search), where taxonomies aid human users to broaden or limit results using the *class-subclass* relation. Ontologies, on the other hand, aim to be machine-readable in much greater extent, providing specialized types of relations, so the knowledge can be derived by traversing the relation graph (this process is called *inferencing* or *reasoning*).

As ontologies are usually built using the RDF, URI identifies each individual. This feature allows for interconnection of separate ontologies based on the individual's URIs. Ontology hierarchy, dependency and inheritance is also possible and widely used (see subsection 4.6.5).

Further, more complex ontology languages support restrictions on properties (e.g. cardinality) that allow more precise definitions of rules in a model. One of them is the *Web Ontology Language*.

4.6.4 Web Ontology Language

From a large number of existing ontology definition languages, *Web Ontology Language* (OWL) gains importance and popularity as it is a W3C standard [48] intended to be used for an implementation of the *Semantic Web* and the *Linked Data* approach. The OWL extends the *RDF Schema*.

One of the key features of the OWL is the fact that its semantic expressiveness is formally grounded in a particular description logic (the specific description logic type differs among different versions of the OWL – see below). The formal basis of the OWL allows for *inference* or *reasoning* over the asserted facts. The inference engine uses the formal logic reasoning to derives new statements (triples) about the system modeled by the ontology. Brief overview of the formal background behind ontology reasoning is provided in [49].

Several versions of the OWL exist. First, there are two main iterations – OWL 1.1 from 2004 and OWL 2 from 2009. Next, different variations (*profiles*) of the OWL exist, differing in complexity of relation definitions. Lesser complexity of the model is trade-off for lesser computational complexity when reasoning over the model.

The OWL is widely used both by researchers and commercial vendors, as it is supported by the W3C and it extends other established standards – RDF and RDF schema.

4.6.5 Upper level ontologies

Ontologies are meant for interconnecting data from different sources and different domains, ultimately creating one large graph of all the available knowledge available in machine-readable format (Semantic Web). That said, methods for ensuring data consistency and concept alignment. In simple words, there is a need to be able to declare that a certain term (usually class or relation) has the exact same meaning as the same term in another data source. The RDF allows for that by identifying all the resources by URIs. If authors of an ontology want to declare alignment with another ontology – e.g. to say that classes are synonyms, that the individual belongs to class defined elsewhere, or to define subclass or sub-property of existing class – they can do so simply by referring the URI from the source ontology.

Upper level ontologies are such ontologies, that do not target any specific domain, but rather provide general concepts that are meant to be reused by authors of domain-specific ontologies. The general concepts from the upper ontology used in domain ontologies then allow for mutual alignment – the common concept occurring in them allows for assumption that the meaning is completely the same.

Furthermore, the upper level ontologies unify modeling approaches (conceptualizations) of space and time and positioning of individuals in space and time. Rich discussion on purpose and features of upper level ontologies provides [50].

One of the main upper level ontologies is Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). One of its derivatives developed under the name DOLCE+DnS Ultralite (DUL) was used as a base ontology for the Semantic Sensor Network ontology by W3C. Thus, DOLCE and DUL conceptualizations define modeling approach of the whole SBMS framework, as it is based on the Semantic Sensor Network ontology.

4.6.6 Semantic Sensor Network ontology

Sensor network is a viable abstraction for certain aspects of building automation installation that are in the scope of the SBMS framework. An ontological representation of a sensor network was proposed by the W3C as Semantic Sensor Network ontology (SSN) [51]. The SSN itself derives its model from previously existing projects described below.

The Open Geospatial Consortium (OGC) provides Sensor Web Enablement (SWE) suite of standards, ensuring syntactic model of sensor networks (SensorML) as well as interfaces and protocols for data exchange. OGC's Observations & Measurements (O&M) provides a limited semantic description of a sensor network. The SWE however does not cover the domain specific semantics (e.g. it does not provide registers of measured qualities).

The O&M was identified as a suitable model of a sensor network for purposes of the W3C. Translation of O&M to Web Ontology Language (OWL) is provided in [52]. An overview of other semantic sensor projects is provided in [53].

Shortcomings of existing semantic sensor projects led to the creation of the Semantic Sensor Network Ontology (SSN) developed and maintained by the W3C Consortium[54]. The SSN adopts the scheme of the Observations & Measurements Model. However, as proved in [55], adjustments were needed to align O&M concepts with the upper-level ontology DOLCE. For the latest version of the SSN, the DUL (DOLCE+DnS-Ultralite) was used as upper-level ontology. The SSN facilitate the Stimulus-Sensor-Observation ontology design pattern introduced in [56] that has its origins in the O&M framework.

The Semantic BMS project facilitates the SSN ontology. The SSN is used as a base for the semantic model of a building automation system.

4.7 Systems integration methods

As systems integration and code re-usability simplify software development (and lower the costs), they are naturally important concerns for software engineers. Cooperation of multiple software products can be achieved by many different ways, providing different services to developers. The following text omits a traditional approach represented by static or dynamic libraries, instead it focuses on solutions that allows interaction between rather loosely coupled components.

On Microsoft Windows platform, the *Component Object Model* (COM) [57] and its later upgrades COM+ and DCOM are still widely used nowadays, even though they are continuously being replaced by .NET framework. The COM provides a language neutral standard for defining interfaces for inter-process communication, later versions add support for distributed computing. In the field of building automation, COM serves as a base for *OLE for Process Control* (also known as *Open Platform Communications*) standard (see Section 4.5), as OLE (Object Linking and Embedding) is based on COM+.

The Common *Object Request Broker Architecture* (CORBA) [58] is a middleware layer providing tools for cooperation in distributed computing environments. The CORBA is multi-platform and supports a wide variety of programming languages.

Additionally, many other language dependent solutions for the *Remote Procedure Call* (RPC) exist, among the most common being *Java Remote Method Invocation* (RMI) [59] and *.NET Remoting* [60]. For both Java and .NET, complex middleware layers were later developed in order to facilitate whole lifecycle of an application, resulting in *Java EE* (specifically *Enterprise JavaBeans* [61] and Java EE containers), *OSGi* platform [62] for Java or *Windows Communication Foundation* [63] on Microsoft Windows. All mentioned platforms allow fast development of flexible distributed applications or components.

During the spread of Internet (specifically the HTTP protocol), binary-based distributed platforms have been substituted by text based solutions, often based on the XML, most notable example is the Web Services Architecture by *World Wide Web Consortium* (W3C), defining an interface using *WSDL* [64] and communicating using *SOAP* [65] messages.

Probably due to the complexity of WSDL and SOAP languages, alternatives are heavily used in recent years; most notably *Representational state transfer* (REST) together with the *JavaScript Object Notation* (JSON) format [66] that originated as a serialization method in *Asynchronous JavaScript And XML* (AJAX) applications.

4.7.1 Web technologies: REST, JSON, & AJAX

The REST is used to implement *RESTful Web services and APIs*. REST facilitates HTTP protocol and its verbs (GET, POST, PUT, DELETE and others) together with hierarchy of URLs to provide CRUD operations to resources managed by the API. The RESTful APIs are easy to understand by developers and interaction with them can be easily implemented in virtually any environment or programming language, as they rely on standardized HTTP. Moreover, implementation of RESTful API is facilitated by a number of existing frameworks, such as Windows Communication Foundation for .NET or Jersey for Java.

The JSON format is a human-readable text format used for serialization of data transferred over the network. Technically, it represents JavaScript object in form of a hash-table – pairs of key (attribute) and value. However, the format itself is language-independent. As with the REST, JSON is supported by wide variety of programming languages and other tools, rendering it highly suitable for scenarios requiring high interoperability.

AJAX applications are web pages that facilitate JavaScript capabilities to obtain data from remote data sources. As a result, the page does not have to be refreshed as a whole in order to update displayed information. Instead, JavaScript method calls are used to dynamically update Document Object Model (DOM) of the page with data retrieved from remote services using asynchronous HTTP requests. AJAX techniques are used in virtually every interactive web application.

AJAX is heavily used in *mockup* applications that combine data from multiple sources. Typical use case are arbitrary data in custom layer placed over maps retrieved from web service such as Google Maps.

Mockup (or in general AJAX) applications are supposed to be one of the significant types of consumers of the APIs provided by the SBMS framework. They are also utilizing *Service Oriented Architecture* (SOA) which is presented in the following section.

4.7.2 Service oriented architecture

Growth of distributed computing middleware (Web Services, CORBA and others) lead to the formulation of a concept of *Service Oriented Architecture* (SOA) as an design based on distinct loosely coupled elements (services) that are interconnected through a network, providing functionality to other components. [67] [68] The SOA is platform and protocol independent and can be implemented using any protocol or even combination of multiple protocols. The SOA supports reusability of services, scaling of the system, redundancy of components and other concepts facilitating software development and management. The principles of SOA were summoned in SOA Manifesto [69], published in 2009.

Service oriented architecture is not rigidly defined, however there are certain generally accepted characteristic. As a source of such commonly perceived features of SOA, Wikipedia.org page on SOA [70] was used. Following principles are the most notable from the perspective of the presented research and lead to favoring the SOA over other approaches:

- Abstraction – Services are considered "black boxes" with hidden inner logic. Behavior is defined by service contract.
- Location transparency – A service can be called remotely from the whole network where the service is available.
- Loose coupling – Services are not dependent on each other from the design perspective. However, they depend on each other on the run-time level, as they exchange data. However, they are aware only of their mutual existence and service contracts, not inner logic.
- Service autonomy and independence – Services fully control the functionality they provide.
- Statelessness – Services are stateless in order to minimize resource use and simplify interaction.
- Composability – Services are meant to be used by other services and thus composing new, more complex services.
- Reusability – Single service can be used by a number of different applications.

The SOA becomes a natural choice to systems architecture as cloud computing, Software as a Service (SaaS), mash-up applications and other distributed computing methods gain spread.

In an attempt to unify the communication of different services within an organization, the concept of *Enterprise Service Bus* (ESB)

[71] was defined as a middleware layer that receives messages from producers, translates them to the format used by the bus, routes them to the receivers and translates them to the desired output format. The ESB provides an additional improvement to the SOA in terms of ease of management and even looser coupling of components, as they don't have to use the same communication protocol.

However, the less robust concept of "pure" SOA seems sufficient for most of the real deployments and situations.

4.8 Data extraction & analysis tools

As analytical and reporting features are usually requested and expected by users of virtually every information system, the field receives large amount of attention both from commercial subjects and research groups. The following section provides overview of different techniques used for extracting valuable information from large amounts of input data.

The nature of queries used for a data analysis significantly differs from queries used in other scenarios, such as inserting data into the database. Each type of use lays down different set of requirements on the database engine. Two types of usage can be specified:

- *On-Line Transaction Processing (OLTP)* – Large number of transactions and high ratio of data modification queries. The database is optimized for a fast data modification and immediate responses. Contains detailed and up-to-date information and the scheme is normalized.

- *On-Line Analytical Processing (OLAP)* – Analytical queries used for generating reports that often return large amounts of data. The data are not required to be completely up-to-date. The database is optimized for joining the data, aggregating and grouping. The database scheme is not completely normalized in some cases.

The differences between OLTP and OLAP operations leads to separating them and creating dedicated database store for purposes of the data analysis. In the field of Business Intelligence applications, the

OLAP database is usually referred to as data warehouse. Since mid-1990's, the two best recognized approaches to data warehouse design and implementation have been established as de-facto industry standards. The approaches are known under the names of their authors as Inmon [72] and Kimball [73]. The data are conceptually represented as a multidimensional array (data cube). By making "slices" in the cube, we select the proper data of interest. Fundamentals of the OLAP and other related technologies are provided in [74] and in updated, newer article of the same authors [75].

Placing the data into the OLAP data store does not ensure successful knowledge extraction by itself. In the rest of this section, two related problems are covered – importing data into the warehouse and data analysis methods.

In the context of data warehousing, the process of filling the OLAP database with data is referred to as *Extract, Transform, Load* (ETL). The process comprises of gathering the data in source format (Extract phase), modifying the data to fit the target structure (selecting needed attributes, changing the structure, data types or values, aggregating data - Transform phase) and importing them into the final destination (Load phase).

An analysis of the data is often performed using standard SQL queries, which is sufficient for most of the cases, but reaches its limits when the amount of data or attributes is too high. Therefore, advanced techniques are subject of extensive research. Examples of such methods are *Machine Learning* and *Data Mining* techniques and *Complex Event Processing*.

Fundamentals of the *Data Mining* are covered [76] and in [77]. The technique uses methods of the machine learning (self-learning algorithms) for extracting knowledge from data that cannot be processed by humans due to complexity or volume. Examples of possible methods for data mining are clustering or searching for outliers.

The *Complex Event Processing* (CEP) is a method of data processing proposed by David Luckham in [78]. The CEP engine receives vast amounts of events from the system. Based on defined rules, the engine aggregates them and forms the complex events with high information gain, thus allowing users to extract significant information from the processed data.

For the environment of BMS at MU, prototype of data warehouse and ETL process was proposed in [79] (in Czech). Possible use of data mining methods was examined in [80]. The prototype of Complex Event Processing engine for the BMS was designed and implemented in [81] (in Czech).

4.9 Data access policies

There are harmful scenarios that exploit building automation data in some form. Generally, building automation data are considered as sensitive and confidential in many cases. As building automation systems are cyber-physical systems, there is one more notable thread when comparing building automation data to data in other information systems. The data often represent current state of the facility systems. Form such data, presence of an occupant can be derived. Data form security systems (motion sensors, security zone states), air conditioning (occupancy sensor, state of the local air-conditioning unit), or lighting (motion sensor, state of the lighting source) can be used to gather information about the occupants in the facility. Another typical type of data monitors state of the windows (open/closed) to be able to switch the AC unit off when the window is open. Such data can be used by the intruders, e.g. to plan their movement through the building. Thus, security of the facility data is an important aspect of the data-providing tools such as Semantic BMS.

This section deals with certain specifics of security policies in facility management with accent on building automation. It does not focus on implementation of security policies, which is out of scope of this work. The section also does not aim to provide extensive and exhausting overview of all available or used approaches. Rather, it informally discusses advantages and disadvantages of selected practices observed in several facility information systems that were examined during the research. At the beginning, basic overview of several established security policy mechanisms is provided. Extensive overview of traditional access control mechanisms can be found in [82].

4.9.1 General approaches to data access policies

One of the classic approaches to the resource access policies is implemented in UNIX-like systems for file system access control. The User-Group-Other and Read-Write-Execute distinction of privilege levels (access modifiers) is not widely used in information systems due to its inflexibility, as it is impossible to fine-tune the access rights (e.g. file created by the user from the group Admin cannot be assigned write privileges to the members of the Managers group and read privileges for members of the Operators group, as they both fall to the Others category).

Another widely used approach for the filesystem access control is Access Control List (ACL), implemented in MS Windows and many Linux distributions and filesystems. When using the ACL, every resource has a list of access rights attached. Security principals (usually users or user groups) contained in the list have their own access privilege specification, usually distinguishing rights to Read, Write and Execute. Vast number of information systems adapts the ACL approach. There are usually some modifications needed, however the basic principles do not change, as the ACL approach is suitable for heterogenous types of resources.

In Relational Database Management Systems (RDBMS), the access rights are usually not granted on the lowest possible level (usually table), but on a set of tables organized in the common database schema, or on the level of the whole databases (Fine-grained user privileges are then handled on the user level). Then, when tables are made visible to the user, allowed actions are specified for the user. There are basically three different types of tasks that the database users might perform – Schema modification, Data modification, and Data reading.

There is a substantial difference between the filesystem and the database access privileges. For the filesystem, the access rights are attached to the resource. In the databases, the access privileges are attached (at least at the conceptual level) to the principal.

In various domain-specific information systems, RDBMS is usually used as the data storage. The access privileges are then stored in a table that generally contains three attributes - Resource, Principal, and Right. Usually, the authorization is performed immediately before the access to the resource is granted, suing a lookup query that checks if the table contains record meeting given criteria: `SELECT COUNT(Principal) From Privileges WHERE Principal = 'P' AND Resource = 'R' AND Right = 'V'`.

The general approach to security is thus very similar across domains. What's different is the concept of resource that gets various content in different domains and different applications.

An application is a typical resource that has access rights assigned. The application provides its user interface to the user, as well as business logic and data. This defines privileges of the user (i.e. which actions are allowed).

In the field of facility management, typical applications are Space planning management, Energy management, Maintenance planning tools, Maintenance work request update, Maintenance supervision, Helpdesk, etc. In project management tools, the applications are usually similar to Issues, Tasks, Code repository, Milestones, Planner, Wiki and so on. Each of the applications is a suitable for different group of workers, depending on their roles in the processes supported by the information system.

4.9.2 Data access policies in facility management IS

In this section, different examples of data access policies are presented. The software systems described in this section were selected because they are used at the Masaryk University and the author has an experience with their usage and administration.

Use case: CAFM system

Archibus is a Computer-Aided Facility Management System widely used in many organizations. It defines three different dimensions of authorization in the terms of the data access. First, it allows to assign Applications to users. Second, it provides Security groups. Finally, the system allows to create Virtual Private Archibus Restrictions. Considering the data access, the Application defines which tables are visible to the user. Security group defines which columns (attributes) are visible to the user. Restrictions define which rows are visible to the user. This multi-dimensional system of user privileges allows to precisely define user privileges depending on:

- User profession – space manager and maintenance manager will have different Applications available
- Level in the management hierarchy – Director will have less restrictive Security group and will see more attributes
- Used facility – Facility managers responsible for the facility (e.g. site) will not see data from other facilities (e.g. another sites) due to the applied Restrictions.

This is very powerful tool for precise privilege management that satisfies vast majority of relevant use cases. According to author's experiences, in most use cases not all the three privilege dimensions mentioned above are facilitated.

Use case: BMS

Delta Controls OrcaWEB and *enteliWEB* are a web-based Building Management Systems that provide native BACnet support. The OrcaWeb is used at the Masaryk University at the moment and the enteliWEB is a newer product replacing the OrcaWeb. However, the security principles remain very similar.

The difference between the Archibus system and the OrcaWeb is the nature of presented data. The Archibus basically presents data that are stored in its relational database. The Orcaweb and enteliWEB, in contrary, present data downloaded from the automation devices on-the-fly. The Archibus thus handles data from single source and the BMS handles data from distributed networked system.

There are several basic Applications defined in the BMS – Dashboards, Navigator (i.e. network view), Graphics (i.e. prepared user interfaces, such as interactive floor plans and technology schemes), Alarms, Reports, and Logs (i.e. historical data). In the case of Graphics, only relevant parts of the user interface are usually made available to the user.

Additionally, the access to the automation network is restricted. The user is separately granted access to:

- Devices – PLCs and other automation devices;

- Data types – e.g. Analog input, Binary output, Schedule, Calendar, Trend;

- Services – most notably Read and Write.

As in the case of the Archibus system, multiple dimensions of security settings allow to define access privileges based on the team role and facility usage.

However, such complex security policies bring undesirable downside. Certain combinations of setting can result into unwanted behavior of the system. The configuration of complex access conditions is highly error prone. Two problem can occur – either the users can view more data than they should, which is both risky for the administrator and uncomfortable for the user, or the restrictions are set too strictly and some application is unusable because of the insufficient rights. Multiple dimensions of access controls are clearly needed, however complex validity checks are not implemented in the tested systems.

Use case: BIM-like system

Compass application (*Kompas* in Czech) is the BIM-like system of Masaryk University that visualizes data from the building and technology passport (see REF), the access control is defined on the level of Location and Technology. The general concept behind the security policy is that the user is able to view all devices of allowed type that are located at allowed location. Devices of different types are not shown, as well as devices of granted type outside the granted location. The user role is not distinguished, as the data are read-only in the Compass application. The Location allows to grant access to certain sub-trees of the location hierarchy (Sites – Buildings – Floor – Room). It is possible to grant access to the whole site by one assertion in the privileges database, there is no need to list all the subordinate buildings, floors, and rooms. The allowed device types are selected from the hierarchical structure of the Technology that forms a tree of generalized device types. The user can be granted access to the certain sub-tree of the hierarchy, e.g. to be able to view all the devices related to the building automation system or the HVAC system.

4.9.3 Guidelines for the Semantic BMS design

It is worth noting that in all the discussed access privilege assignment methods hierarchical structures pay important role. This patter is observed in the filesystem security (privileges for the directory are applied for all the children nodes), CAFM system (hierarchy of locations – site/building/floor/room), in BMS (subnets of the automation network) and also in the BIM-like systems (hierarchy of location, hierarchy of device types).

From above mentioned examples, conclusions can be drawn for design of the access control mechanism in the Semantic BMS:

- The hierarchical approach without the need to explicitly define access control for each of the nodes of the tree is obviously the best practice.

- The Semantic BMS combines data from the BAS and from the BIM systems and should not interfere with the data access policies of the source systems. The best option is to directly adopt the policies form the source systems. However, the problem of errors originating from combining multiple access control dimensions becomes even more crucial when combining data from multiple information systems.

- The Semantic BMS does not use relational database for storing its data. Instead, it uses ontology as a data model and triple store as a storage. The automation data are loaded directly from the automation network and they are not stored in the middleware. Traditional approach thus cannot be used.

- Since the BIM and BAS systems are prerequisites of the Semantic BMS deployment, API that allows the user to provide custom authorization mechanism should be sufficient - the authorization itself can be performed by using the BAS or BIM policies stored in another data store.

-

4.10 Summary

This chapter provided overview of fundamental concepts and principles that are relevant to the aims and methods of the presented research. The chapter started with an overview of the general fields the thesis is related to: Facility management, cyber-physical systems, critical infrastructures, and building automation and management systems. In the later parts, it focused on technologies, approaches and methods that are facilitated in the Semantic BMS project. Namely semantic modeling, systems integration, data analysis, and data access policies. Presented topics define the context for the Semantic BMS project that is introduced in the next chapter.

5 Related work: Semantic descriptions of BAS

This chapter provides overview of the related work in the field of semantic descriptions of building automation systems, as this issue was identified as the most important to overcome for the thesis aims.

5.1 BAS a part of a facility

Building automation system is a part of a facility. First, this chapter provides overview of semantic descriptions of a facility, where BAS is only a part of the whole model.

In architecture, engineering and construction industry, Industry Foundation Classes standard (IFC, in current version 4 – ISO 16739:2013) aims to provide a common data standard for BIM software solutions, significantly increasing interoperability in the field of BIM systems.

The object-based data model together with the provided file formats ensure data exchange between BIM systems and can be viewed as a source of semantic information that can be used for analysis of building performance, as shown in [83].

However, the IFC does not aim to model complex semantic information concerning data points available in the building automation systems. In [84] and [85], direct mapping of data points to IFC entities is used for building operation analysis. As noted by the authors of [85], the IFC itself does not provide built-in capabilities for describing features of interest and properties observed by sensors.

The IFC was translated to the OWL language, resulting in the ifcOWL ontology presented in [86]. The ifcOWL is a direct and exact translation of the IFC standard as it is modelled in the EXPRESS language. The paper provides both the ontology definition and description of the conversion process. The authors argue that the translation enables systems integration and development of knowledge-based systems that are both understandable by humans and processable by machines.

For many uses, this ontology is unnecessarily complex. Therefore, there is a demand for simplified representations of the building model that could be used in multiple scenarios where only a part of the BIM is used or where certain relations between the model components are not desired. One approach to development of such complex ontologies is to generate them from the complex ones. A method providing simplified models derived from the ifcOWL is presented in [87].

There are also ontologies developed independently, containing only the most significant elements. The W3C Linked Building Data Community Group is working on a set of smaller ontologies describing building data that are suitable for the linked data domain. One of such ontologies is the Building Topology Ontology (BOT)[1] that represents basic elements of a built environment (building, storey, space, element). An element concept expresses equipment installed in the building including building automation devices. The BOT does not provide additional semantic information besides installation location, however it aims to serve as an basis that will be extended for specific use cases.

An extensive overview of the semantic web technologies in the architecture, engineering, and construction industry is provided in [49]. The authors identify three main uses for the semantic technologies: interoperability (i.e. strict mapping of different representations of an identical object), linking across domains (e.g. use cases utilizing heterogeneous data sources to derive new knowledge) and logical reasoning and proofs. The paper presents vast number of different approaches and efforts that pursue one or more of the general goals. The most relevant mentioned projects are included also in this chapter.

5.2 BAS as a standardized protocol

Building automation systems are basically a communication networks that use certain communication protocol (or multiple protocols) for communication. However, besides the communication, they perform control (regulation) of a building operation. Some automation protocol focus almost purely on information exchange (e.g. *BACnet*, *MODBUS*), other describe also a way algorithms are constructed (e.g. function blocks in *KNX*). The following section provides overview of the semantic information that can be derived from the protocol specifications.

Standardized building automation protocols such as BACnet (ISO 16484-5), LonWorks (ISO 14908-1), KNX (ISO ISO 14543), or ZigBee generally cover an operation of building automation devices, provid-

1. Available from `http://www.student.dtu.dk/~mhoras/bot/index-en.html`.

ing specifications for physical communication layer, data link and networking layer and application layer on the highest level. Automation protocols focus on communication interfaces and do not provide tools for a complex and structured description of the semantics of the data points.

Some automation protocols (LonWorks, KNX, ZigBee) introduce the concept of function blocks. Similar functionality is also available in products implementing other automation protocols (e.g. BACnet-enabled BMS web server entelliWeb by Delta Controls company). Functional blocks are generally collections of input, outputs, parameters and algorithms that cooperate on performing defined task, thus provide additional semantic information. Although they are protocol or vendor specific, IEC 61499 provides a standardized model of function blocks. The information provided is limited when compared to ontologies such as SSN, namely because of the absence of the core concepts of "Observation", "Feature of interest" and "Observed property".

Ontological representations of protocol specifications such as BACOWL[2] exist, but they provide an exact translation of protocol specifications to ontology language (e. g. OWL) and do not enrich the descriptive power of the protocol in the means of additional semantic information.

5.3 BAS as a complex system

Advanced semantic representations model the BAS as a complex system with multiple aspects. While the approaches mentioned in the first section perceived the BAS a "just another part of a facility" and the approaches in the second section modeled the BAS as a rather isolated system, works in this section are aware both of the internal structure of the BAS and the surrounding systems and environment. That said, presented works pursue large spectrum of different goals. The following list shows just a few examples of different purposes:

- integrate different automation systems;
- describe the processes controlled by the BAS;

2. Available from `http://bacowl.sourceforge.net/`

- improve the control capabilities of the BAS;
- simplify BAS design and deployment;
- use BAS as a source of data for operation analysis.

From the point of view of computer science, building automation systems can be at least partially viewed as a sensor network. The semantic description of sensor networks is a subject of ongoing research, as it combines two significant trends in computer and information science – Semantic Web (Open Linked Data) and Internet of Things. Furthermore, the trend towards "Smart Cities" stimulates research on the integration of automation systems and building automation data analysis. The most relevant projects from the field of Semantic Web is the Semantic Sensor Network Ontology presented in [54]. For further information on the SSN, see section 4.6.6. The SSN serves as basis of the Semantic BMS project presented in this work. Some of the below-mentioned works use the SSN as their foundation, some choose different approaches.

The MOST project presented in [88] provides a framework for building operation analysis. The MOST framework uses a relational database for storing basic semantic information about operation data. Even though the technologies used are not related to the scope of the research, the general focus and goal are very similar, as the MOST framework aims to annotate building sensors with semantic meta-data describing the measurements coming from the sensor.

In [89], authors define several basic "ontology modules", addressing different aspects of automation systems' semantic description. The authors further argue for common modelling paradigm for automation system ontologies.

The Smart Appliances REFerence ontology (SAREF) presented in [90] aims to semantically describe the domain of home automation, where individual installations are of limited size and also the range of observed properties and employed devices is limited. The SAREF is able to describe various properties of home appliances including price, it provides a basic framework for a description of sensor measurements, and aims to completely describe relation and processes occurring in the home automation system.

The SESAME-S project presented in [91] aimed mostly on households, providing both hardware and software for home automation. It provided several ontologies related to the automation system operation, (Automation ontology, Meter Data ontology and Pricing ontology), aiming to optimize energy consumption of a household.

An extension to the SSN is presented in [92] and [93]. The extension adds a model of physical processes occurring in the building (e.g. adjacent rooms exchanging energy) in order to provide a tool for building operation diagnosis and anomaly detection.

The linked Open Data approach for building automation data streams is facilitated by EDWH Ontology proposed in [94]. The aim of the EDWH ontology is to provide a bridge between the SSN ontology and the W3C RDF Data Cube vocabulary, as the data are meant to be analyzed by On-Line Analytical Processing (OLAP) data cube techniques.

In [95], [96] and [97] the author use SSN-based ontology for energy management based on sensor data using OLAP and Complex Event Processing. The semantically described BMS data help to establish situation awareness on the strategy level, allow multi-level evaluation of energy consumption (from organization level to the level of individual appliances).

In [98], an approach to automatically associate sensor measurements with descriptive tags from a standard set (i.e. project Haystack) is presented. The method uses supervised learning and classification algorithms facilitating the statistical distribution of individual sensor data.

The Device Description Ontology (DDO) proposed in [99] aims to allow for an automated design of building automation networks. The DDO is extended by the BASont project [100]. The BASont is middleware layer aiming to facilitate system design and commissioning, thus focused on the physical structure of BAS.

A method for automatic design of a BAS network is proposed in [101]. A set of ontologies covering each of the engineering steps is defined. The process starts at the requirements phase which is used for a development of an abstract design describing functions of components. A detailed design of the BAS network is then derived as abstract function definitions are replaced by device profiles and function blocks defined in protocol specifications.

In a follow-up paper [102], an ontology is proposed to describe different types of automation devices. The presented method aims to allow automatic design of the BAS by selecting cost optimal devices. The proposed tool provides API and GUI for querying the ontology.

In [103], slightly different point of view on requirements engineering is proposed by the same authors as above. Ontologies are used for requirements definition and for description of room templates. Room templates represent types of rooms occurring in the facility and describe functions of the BAS needed in them. The IFC standard is used for gathering the data about the building structure. Different approach to requirements engineering - uses ontology for description of requirements, uses room templates (several), uses IFC to gather data about building structure.

The concept of Building as a Service (BaaS) is introduced in [104], aiming to simplify development and maintenance of building automation installations. The BaaS contains an abstract semantic model (vendor or platform independent) of building automation systems. The BaaS reflects the difference between a sensor and a data point providing the BAS with data.

An approach that facilitates the BaaS and brings the SOA approach in building automation is presented in [105]. The BAS is viewed as a network of nodes that provide services to each other. The nodes are semantically described and information discovery is performed using SPARQL query.

In [106], an automation system that follows SOA principles is proposed. The system aims to be configured automatically and uses an ontology for description of the services provided by the devices installed in the building.

The SOA approach is facilitated in [107]. In this case, however, ontologies are not used. Semantic description of services are used within SOA to integrate BAS systems with the smart city environment.

An extension to the KNX/EIB automation protocol is proposed in [108], using an ontology and inference to enable service and device discovery in the network. Inference is also used for selection of the best available services to satisfy user needs.

In [109] and [110], discovery services for smart building are proposed using enriched SSN ontology. They, however, lack structures needed for complex querying (e.g. hierarchy of locations). Instead, they aim to facilitate a development of self-adapting control algorithms.

In [111], an ontology representing components of the BAS is proposed. The ontology serves as an integration platform for multiple automation protocols. The ontology aims to be placed between the BMS (user interface) and the BAS (automation technology) in order to create a common knowledge base that can be facilitated during the facility control processes.

Integration of different automation systems facilitated by ontological generic application model is proposed in [112]. The ontology models function blocks according to IEC 61499. The generic application model enables deployment of platform-independent system configuration to physical devices implementing different automation protocols.

In [113], ontology for integration of different automation systems is proposed. The ontology describes platform-independent "parameters" (data points) which can be observed or controlled. A different approach to automation systems integration is an ontology-driven OSGi gateway for systems integration proposed in [114], which aims at residential buildings.

The concept of "Semantic Agents" securing different aspects of building operation (Energy management, Safety, Security, Comfort) is proposed in [115]. Semantic agents are complex applications facilitating semantically described automation data.

In [116], the authors propose an ontology that aims to support effective building operation via various tasks such as predictive building systems control or smart load balancing. The ontology defines several basic categories, such as inhabitants, environmental conditions, or control systems. The ontology interconnects the data from multiple categories and allows to gather data from different domains related to a particular sensor measurement.

In [117] and [118], the OntoFM ontology is proposed. The authors address the suitability of building automation data in the facility management decision support and propose the integration of the IFC data, sensor networks, ontology languages and software agents for real-time operation monitoring, aiming specifically at the employment of wireless ZigBee sensors.

In [119], the 2eA-FB project is introduced. The 2eA-B extends the function blocks model by adding the intelligent software agent and semantic knowledge that aims to bridge the gap between the building management and building automation (BAS), the main goal being efficient energy consumption of the system. The agent is able to evaluate validity of certain actions that are to be performed by function blocks and cancel them if the actions are not meaningful in the given moment.

In [120], a negotiation process for home automation is proposed. The goal of the negotiation is to find optimal compromise between energy efficiency and comfort, based on services and resources available at the moment and on current state of the user. The algorithm uses the ontology technologies to define the knowledge base used in the negotiation process.

In [121], the authors deal with the problem of applying the correct fault detection and diagnostics (FDD) algorithm to an individual BAS. They propose an ontology that describes the BAS system (using th BIM data) and the FDD algorithms to be able to automatically match suitable algorithm for the given BAS installation.

The same authors in [122] use ontologies and inference to reason about fault propagation in the BAS. Ontological representation of the BIM together with the SWRL rules to reveal causality of faults and their expected propagation through the system.

In [123], an ontology is used for conflict resolution in applications of ambient intelligence in smart households. The ontology models requirements of the users and SPARQL is used for detecting conflicts in requirements (e.g. conflicting lighting setting of two users in the same room).

5.4 Summary

In general, existing semantic models of building automation and management cover different aspects of BMS systems, use various structures for storing semantic data and facilitate BMS data in multiple contexts.

The main contribution of the original research presented in this paper is the novel capability of constructing complex queries over the semantic model that links BMS data with the BIM database.

6 Proposal: The Semantic BMS

In order to facilitate the development of analytical applications to maximal extent, a middleware layer named the Semantic Building Management System (Semantic BMS, SBMS) is being developed. The goal of the project is to provide an BMS-protocol-independent model of intelligent building systems that can be queried according to various parameters. The Building automation system can be viewed as a sensor and actuator network for the purposes of data analysis.

Applications built facilitating the Semantic BMS project will be able to provide decision support for improving building performance and efficiency. Their capabilities will include statistical analysis of both historical and present data or different types of data visualization – usually charts, but sensor data can be successfully visualized using the BIM 3D models or classic 2D maps. Generally, the goal is to allow facility managers to use building automation data in tasks common in business intelligence applications (and in CAFM systems) that usually use different data sources.

In the Chapter 3, there were following tiers of analytical application for the BMS defined:

- Data retrieval;
- Definition of data semantics;
- Analysis;
- User interface and experience.

The tiers of the complex applications covered by the Semantic BMS project are illustrated in the Figure 6.1.

6.1 Design requirements and considerations

During the development of the Semantic BMS, multiple aims were set to achieve the goal stated for the research. The aims were formulated as functional and non-functional requirements laid on a resulting Semantic BMS middleware software artifacts. The following subsections

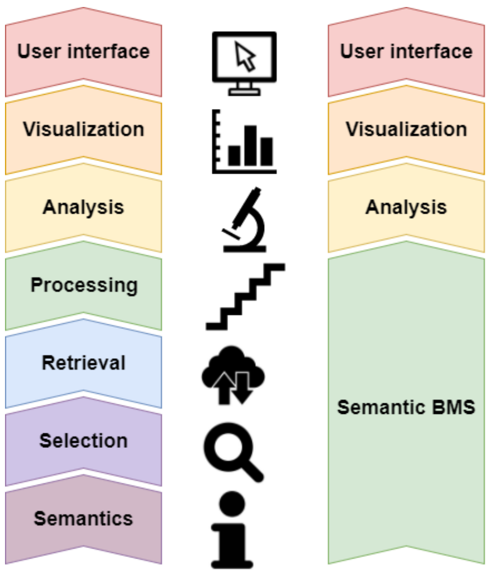


Figure 6.1: Capabilities of the Semantic BMS.

describe both the functional and non-functional requirements that have been laid down prior to the development.

6.1.1 Functional requirements

The functional requirements are described here in a rather informal way, as they were not specified precisely at the beginning of the development. Specific behavior that follows guidelines presented here is described later in this chapter. The main requirements on the Semantic BMS projects were following:

BMS data: The goal of the API is to provide data coming from the building automation and management systems. Such data are usually sensor data that represent measurements of environmental variables or states of equipment that ensure a building operation. The provided data are intended to be used for building operation analysis and facility benchmarking.

Structure: The focus of the framework is to provide clearly structured data that can be queried and further processed by end-user applications. The structure allows for further machine processing and advanced analysis of the provided data, as opposed to existing solutions.

Semantics: The BAS data are annotated by additional semantic information that fully describes the nature of the data. Such information include mainly following characteristics (see also Figure 6.2):

- Location of the sensor or actuator;
- Type of the sensor or actuator – Is the device a temperature sensor or a humidity sensor?
- Measured or controlled variable (physical quantity) – Does the temperature sensor measure water temperature or air temperature?
- Spatial scope of the measured value – Does the electricity consumption relates to a single appliance, a building, or a whole site?
- Temporal scope of the measured value – Is the electricity consumption measured for current month or is it overall consumption?

Semantic querying: The semantic information can be queried to retrieve all the elements that meet specified criteria. The queries can be composed by the user and are executed in real time. The user has full freedom in specifying the request parameters (restrictions on the retrieved data) and information that they wish to receive in the response.

Integration with the BIM: The semantic description of the BAS data uses the Building Information Model (see Section 4.2.1) relations to describe the BMS data. This is possible because all the devices that are encompassed in the BAS and BMS (sensors, PLC, controllers, ...) are already present in the BIM database. Utilization of this relation between the BIM and BMS facilitates further integration with other

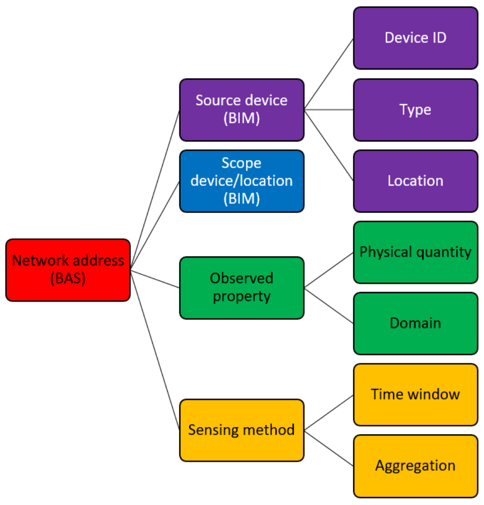


Figure 6.2: Semantic information added to the BAS addresses in the Semantic BMS project.

information systems (such as CAFM systems), allows for use of existing applications (e.g. visualization of the BMS data within the BIM visualization), and provides additional level of semantic information that is not directly present in the Semantic BMS data.

Distinction of a data point and actual sensor or actuator data: The semantic annotation relates to the data point, or more precisely, to a specific (network) address in the BAS, that represent certain piece of equipment. The sensor or actuator data itself (i.e. actual value with timestamps) are not annotated. This brings improved performance and additional querying options, as described later in this chapter.

Operation data querying: The Semantic BMS is able to retrieve both present data from the BAS (i.e. values sensed at the very moment by sensors and present states of actuators) and historical data from archive databases. The data are retrieved according to user needs. The user has full control over multiple parameters of the query: Time period, output data structures, and optional data processing options.

Data processing: The Semantic BMS provides variety of data processing options that serve data that match the users (or developers) needs. However, the aim of the middleware layer is not to perform complex analytical tasks. The data processing capabilities thus provide functions that focus on adjustment of the data structure that simplifies further analysis or visualization, such as:

- Aggregation – Simple aggregation functions such as arithmetic mean or sum.
- Sampling – Adjustment of the timestamps of the recorded data to increase usability of the data by aligning the data to the same moment in time.
- Interpolation – Used mostly in combination with the sampling, interpolation allows to compute the probable value in a given moment based on surrounding data.

Grouping: A number of analytical task in facility management involve comparing parts of the facility among each other (e.g. comparing individual building within the site). In order to minimize the number of requests, the Semantic BMS allow to group the data points according their common characteristics. That way, all the required data can be obtained in one query and then processed separately. An example of such query is a request for all the data from temperature sensors located in a building, grouped by floor. Data for each floor can be processed (e.g. aggregated) separately and the results can be easily compared.

6.1.2 Non-functional requirements

The framework architecture follows certain guidelines that ensure usability of the final software product. Such considerations are considered non-functional requirements placed on the resulting software product. In the following section, architecture considerations will be further described. The considerations are categorized, however boundaries between categories are rather blurred, as the categories are tightly interconnected.

Simplicity: The framework aims to simplify development of new applications facilitating building automation data. The architecture accents convenient integration with heterogeneous systems using any platform (e.g. Java, .NET, JavaScript). To achieve this goal, the framework itself provides familiar, easy-to-use technologies, conventions, and established approaches to service architecture. The main concept of the framework is aligned with the Service Oriented Architecture methodology. The framework consists of separate services (Semantic API, Data Access API) that can be used separately or in a cooperation – allowing chained calls to API methods. The services are implemented using de-facto standard for modern web applications – RESTful APIs, providing JSON (JavaScript Object Notation) formatted data. Selected technologies allow for convenient use of AJAX (Asynchronous JavaScript and XML) in client applications, as demonstrated in Semantic API client application (see 7.1). On the other hand, the framework can be easily used with applications utilizing more robust or complex frameworks. However, the APIs do not rely on platform-specific or vendor-specific technologies, frameworks, or middleware, such as Windows Communication Foundation, OSGi or others, that would restrict deployment options. The framework utilizes existing tools, frameworks, and models to the maximal extent, so the codebase stays compact and maintenance is simplified. All the above described decisions aim to further speed up development of client applications.

Independence: Independence of the SBMS framework consist of several aspects: vendor independence, platform independence, and automation protocol independence. These concepts ensure availability and suitability of the framework. Since above mentioned terms are rather vague, more specific description follows. Use of Open Source software products ensures the *vendor independence*. Naturally, the independence is ensured only on the technological level, not on the license level, where various limitations might apply (e.g. for use of the framework for commercial purposes). The *platform independence* ensures usability of the framework on wide spectrum of devices and operating systems. The platform independence is provided by use of the Java technology. The *automation protocol independence* describes the fact that the abstract model of the semantic is not tailored for one specific automation protocol or system, but aims to describe general concepts found in building automation.

Agility: Design of the Semantic BMS architecture aims to provide multiple deployment options, modular structure, and extensibility. Compact and lightweight services providing Semantic API and Data Access API can be deployed either on single node, or on separate nodes depending on the requirements and capabilities of the facility. The services require lightweight application servers (e.g. Apache Tomcat, Jetty, Grizzly) and can be adjusted to the needs of the user by custom-developed modules. The modules ensure function of the Data Access API (connection to the BAS and to the archive server). This approach is required to support different BAS platforms, archive servers and their combinations. For example, some commercially available solutions (e.g. OSIsoft PI) provide advanced archival server (historian) that computes advanced aggregations over the operation data. If an organization owns license to such solution, certain calls to Data Access API can be simply forwarded directly to the archive server. In other cases, the processing must be performed using database query language (such as SQL), or, in some cases, directly in the application code of the module. The next use of custom modules is expected as an implementation of application security which is likely to require integration with organization's identity management. Furthermore, the model and APIs can be reasonably easily extended by new properties when required by user, which is possible due to the use of established modeling and querying frameworks. Due to the heterogeneity of used technologies, modular and extensible architecture of the middleware is necessary to keep the Semantic BMS usable and suitable to the maximal extent.

Versatility: The SBMS framework aims to remain as versatile as possible on both its frontiers – underlying BAS systems and applications facilitating the framework. On the side of BAS, the framework provides interfaces and mechanisms to develop custom adapters for additional support of different automation protocols. The semantic model is abstract and general, thus usable for different implementations of automation systems. On the other boundary of the framework, versatility means wide range of technologies and applications that can facilitate the APIs. From the users' point of view, the SBMS framework supports types of usage ranging from strategic-level task employing aggregate computations to tasks that require drill-down to an input sensor.

Security: Security of the information available through the framework is important aspect of framework design. Some (or even all) of the data available in building automation data are considered sensitive for various reasons (see 4.9). Non-existent or too simple authorization mechanism would limit the number of users of the framework and available data. Only administrators and other super-users with high level of privileges would be allowed to use the system. Therefore, the framework aims to support fine-grained authorization mechanism. The framework allows to validate privileges against two criteria: Data (e.g. locations, devices, data points) and Actions (Read/Modify). Mechanism of plug-in modules allows users to add their own authentication methods. The security architecture of the framework allows to expose the interfaces to wide range of users with precisely limited privileges.

6.2 Architecture design

The structure of the Semantic BMS middleware is introduced in the Figure 6.3. There are two main parts of the middleware layer, denoted as Data providers and Semantics providers in the figure. This (arbitrary) notation distinguishes two types of information available via the middleware layer.

Since the terms *Data* and *Semantics* are generally vague, a more precise definitions are provided below.

The term *Data* denotes sensor readings, actuator commands, user-defined values and other variables provided by the automation system. The Data providers (the most important being Data Access API) receives network addresses in the BAS and returns raw or processed reading and measurements. The Data Access API serves as a mere proxy that reduces complexity of end user applications. It provides unified interface for accessing the data from multiple automation

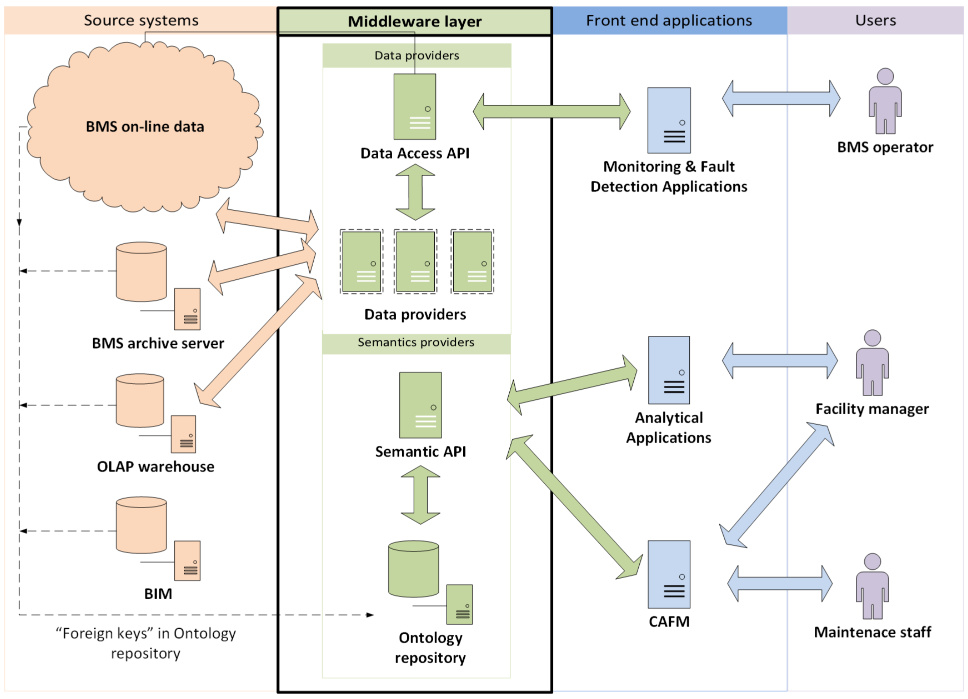


Figure 6.3: Semantic BMS overview.

systems and data stores. However, It does not provide any semantic enrichment of the BAS data.

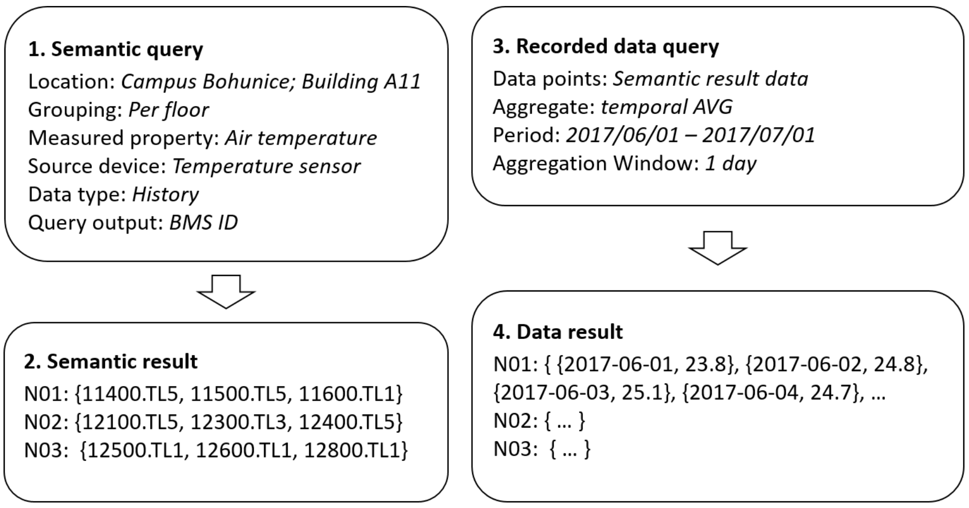
On the other hand, the *Semantics* describes information about the data points – network addresses available in the BAS. The Semantics providers (the most important being Semantic API) provides semantic annotations of the data points that provides meaning to the measured values.

This distinction separates the information about a variable (i.e. data point) from actual values the variable holds in given time (i.e. sensor readings or actuator commands). The semantic description of the sensor is common for all the measured values. The only attribute they differ from each other is their time stamp. This principle is reflected by the division of the middleware to the mentioned *Data* and *Semantics* parts.

This division brings also some practical benefits. The number of the BAS addresses (sensors, actuators, and other) is naturally by orders of magnitude lower than the amount of the records that represent the archival sensor readings. The Semantic BMS annotates only the BAS addresses, not each individual measurements, which saves storage space and improves query performance for the intended use cases.

Data retrieval through the Semantic BMS is a two-step process. First, a request to the Semantic API is sent. The request specifies characteristics of data points of interest. The response contains a list of data points that meet the given criteria, possibly grouped by certain parameter. In the second step, the list of data points is passed to the Data Access API that gathers the actual sensor readings and possibly processes them. The process is schematically illustrated in the Figure 6.4.

In the following section, individual parts of the Semantic BMS are presented. First, the Semantic BMS Ontology is presented. The ontology serves as a data source for the semantic annotation. Second, the Semantic API is presented. Last, the Data Access API is discussed. After that, information about the project repository follows.



1. Semantic query
Location: *Campus Bohunice; Building A11*
Grouping: *Per floor*
Measured property: *Air temperature*
Source device: *Temperature sensor*
Data type: *History*
Query output: *BMS ID*

2. Semantic result
N01: {11400.TL5, 11500.TL5, 11600.TL1}
N02: {12100.TL5, 12300.TL3, 12400.TL5}
N03: {12500.TL1, 12600.TL1, 12800.TL1}

3. Recorded data query
Data points: *Semantic result data*
Aggregate: *temporal AVG*
Period: *2017/06/01 – 2017/07/01*
Aggregation Window: *1 day*

4. Data result
N01: { {2017-06-01, 23.8}, {2017-06-02, 24.8}, {2017-06-03, 25.1}, {2017-06-04, 24.7}, …
N02: { … }
N03: { … }

Figure 6.4: Illustration of the two-step data retrieval process.

6.3 Semantic BMS Ontology

One of the aims of the presented research is to provide an BMS-protocol-independent model of intelligent building systems that can be queried according to various parameters. The Semantic BMS Ontology is a result of the effort and defines the data model used in the Semantic BMS project.

The Building automation system can be viewed as a sensor and actuator network for the purposes of data analysis. The semantic description of sensor networks is a subject of extensive research, resulting in frameworks and tools such as the SensorML language, the Observations & Measurements (O&M) model or the Semantic Sensor Network ontology (SSN). However, for the use in the domain of building automation, particular differences have to be taken into account.

The Semantic BMS Ontology (SBMS Ontology) proposed in this chapter is thus an extension of the SSN ontology, addressing domain-specific requirements of building automation data analysis. The SBMS remains "fully backwards compatible" with the SSN ontology (meaning no modification were made in the SSN itself, it was only extended) and at the same time provide greater semantic description strength for the target domain than the "pure" SSN. The RDF/XML definitions for the Semantic BMS ontology are available at `http://is.muni.cz/www/255658/sbms/v2_0/`. The sample data for the ontology can be found in the Semantic BMS project repository (see Section 6.7).

The proposed ontology aims to represent information (data) available for operation analysis. It does not aim to describe a physical topology of the building automation network or physical properties of the sensors and actuators, such as their operating range. Standardized building automation protocols themselves also provide limited meta-data description of the system as well as other services such as a data store. Similarly, the BIM and CAFM systems provide additional sources of semantic information. As a result, the semantic description of the BMS is not required to contain some information that would be a duplicate (copy) of data available in the BMS, BIM or CAFM systems.

The aim of the presented research is to enrich the BMS with semantic links to entities present in other systems (BIM, CAFM systems) and add a new layer of semantic meta-data that are not available elsewhere. The Semantic BMS Ontology thus contains unique identifiers that can be used to identify the individual from the ontology with an element in the BIM or BMS. The actual integration of the BIM and BMS (e.g. in some kind of user interface) is out of the scope of this research.

The SBMS ontology enriches the SSN by adding the concept of a data point, which is distinct from a sensing device. While a sensing device is a source of data presented by a data point, a data point conceptualizes a measured value available in automation systems. It describes a representation of measured data in a building automation software. Other basic types of data available in the BMS are also modeled. Further description is provided in subsection 6.3.3.

The building operation data are generally not meant to be publicly available on the Internet (as opposed to scientific sensor data). The data are collected purely for internal needs, such as operation analysis. In some cases, they are considered confidential. This is a major difference from the open linked data, as they are required to be publicly available. The BMS data are usually secured and available only from the intranet.

Since the actual measurements are not meant to become part of the semantic web and the linked open data cloud, there is no strict requirement to represent them using the Resource Description Framework. Although it is certainly an option to use RDF for description of the sensor measurements even in private deployments, a different approach can be used. The Semantic BMS describes only the data point – the address in the BMS that contains data from the particular sensor or actuator (i.e. Address XY provides a current temperature in the Room 101 when queried). For that reason, the SBMS ontology does not provide semantic annotation of the measurement data itself (i.e. individual sensor readings – the temperature of 23.8 °C at 2015-08-30 16:10 in the Room 101). Therefore, significant performance improvement can be reached.

Results of a performance comparison of relational databases and RDF triple-stores is dependent of the specific use case. Such comparisons are presented in [124], [125] and [126], yielding heterogeneous results depending on the examined scenario). In the Semantic BMS use case, the performance boost comprises from several main factors:

- Smaller ontology model bring better query performance (the model describing the data points is by orders magnitude smaller than the model which describes individual measurements);
- Smaller ontology model requires less operation memory, improving overall performance of the server;
- Querying for large amounts of ordered data (sensor measurements) can be performed faster when using SQL instead of a direct query to the ontology model, since relational databases are optimized for such types of data retrieval and processing. The relational databases and SQL can be also used for efficient computation of aggregations over the raw data.
- The queries that extensively exploit data relations are resolved using the semantic RDF data stored as a graph. Such queries require table joins in the traditional relational databases and are thus very expensive. On the other hand, native triplestores used for the RDF data are specifically aimed and optimized for a fast resolution of such queries.

Annotated data points can then be queried for specific values at given time using automation protocol methods or SQL (in a case of historical data in a data store). In the presented work, gathering of the operation data is performed by the Data Access API available in the Semantic BMS middleware layer.

Even though the data from the BMS are not publicly available, use of semantic technologies (ontology languages) allows integration with other external sources such as weather data, reasoning and potential compatibility for existing querying and analytical tools due to compatibility with the SSN ontology.

Selected extensions and adjustments present in the SBMS ontology are described in detail in the following subsections.

6.3.1 Specializations of SSN

Figure 6.5 presents key concepts of the SBMS ontology and their relations (inverse properties are omitted from the figure). All of them are derived from the SSN, except the `sbms:DataPoint` class that is introduced in the Section 6.3.3.

The center point of the SSN is the `ssn:Observation` concept. The observation connects all actors and objects that take part in the process of obtaining measurement data, and thus provide all available semantic information via its properties. The specialized `sbms:Observation` as well as properties defined in the SBMS limits domains and ranges to those concepts that are usable in the domain of building automation. The changes affect the definitions of the feature of interest, the observed property, the sensing methods and the sensing devices.

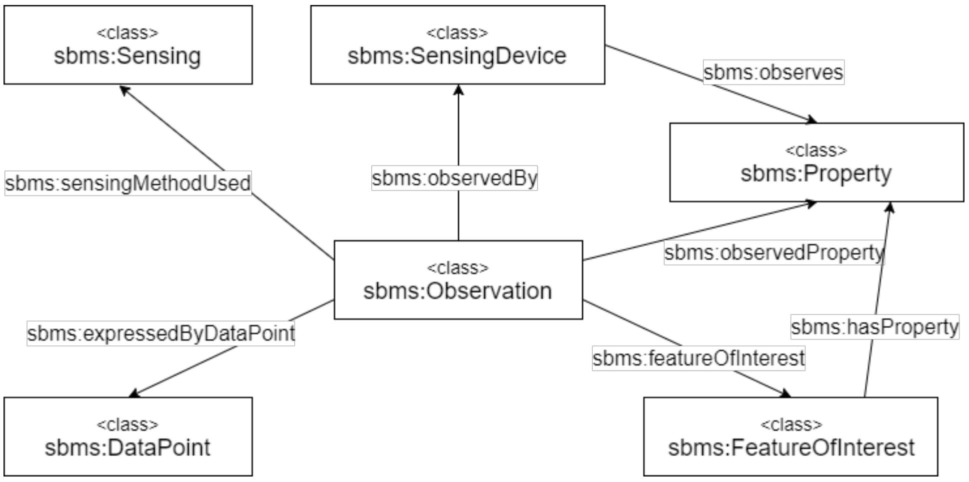


Figure 6.5: Key concepts of the Semantic BMS ontology.

Key concepts for accurate semantic annotation of sensor/actuator data are the *Observed property* (OP) and the *Feature of Interest* (FoI). The concepts were defined in the O&M framework and adopted by the SSN ontology. The FoI represents an object of measurement. The OP represents specific information that we observe. In the domain of building automation, we can demonstrate the concepts on examples such as energy consumption (OP) of a specific building (FoI) or speed (OP) of a specific fan (FoI).

The SBMS ontology further specializes the concepts for use in the domain of building automation. General purpose for this specialization by sub-classes and sub-properties is to distinguish individuals from the specific domain of BIM and BMS from general individuals described by the SSN. Individuals of the `sbms:Observation` are expected to be described by a certain data point in the BMS. This feature differentiates the SBMS subclass from the general `ssn:Observation`. Furthermore, there is a specific requirement that the Semantic BMS places on certain elements in the ontology – they have to be equipped with an ID (data property of string literal type) that can be used to identify the individual in other information systems.

The Semantic BMS requires such IDs for all the sites, buildings, rooms, floors, devices and data points by using the `owl:hasKey` clause on the appropriate classes. This allows defining ranges and domains for properties in a way that ensures that when querying the ontology correctly, all the individuals in the result can be linked to their respective representations in other information systems.

The `ssn:Feature of Interest` is restricted by the subclass `sbms:Feature of Interest` to be site, building, floor, room, or device further described in the BIM database. This restriction reflects the nature of data available in the BMS – since the BMS ensures operation of the building, every piece of information available in the system can be related to a specific part of a facility or to a piece of installed equipment.

A similar restriction is applied to the `ssn:Sensing Device` by the subclass `sbms:Sensing Device`. Every piece of information available in the BMS must be measured by some device connected in the building automation network. Such devices are naturally present in the BIM database. As a result, each individual of `sbms:Sensing Device` has to be a device described in the BIM database, represented by `sbim:Device` class (see Section 6.3.2).

6.3.2 Semantic description of BIM elements

The SBMS ontology contains simplified model of selected elements from the BIM systems (see Figure 6.6; inverse properties are omitted from the figure). Such concepts reside in the dedicated namespace `sbim`. Namely, it contains concepts describing locations and parts of the facilities (`sbim:Site`, `sbim:Building`, `sbim:Floor`, `sbim:Room`), and devices (`sbim:Device` and its descendants) that provide sensor data or that are observed by the building automation system. Classes representing specific types of building equipment are adapted from the IFC 4 specification.

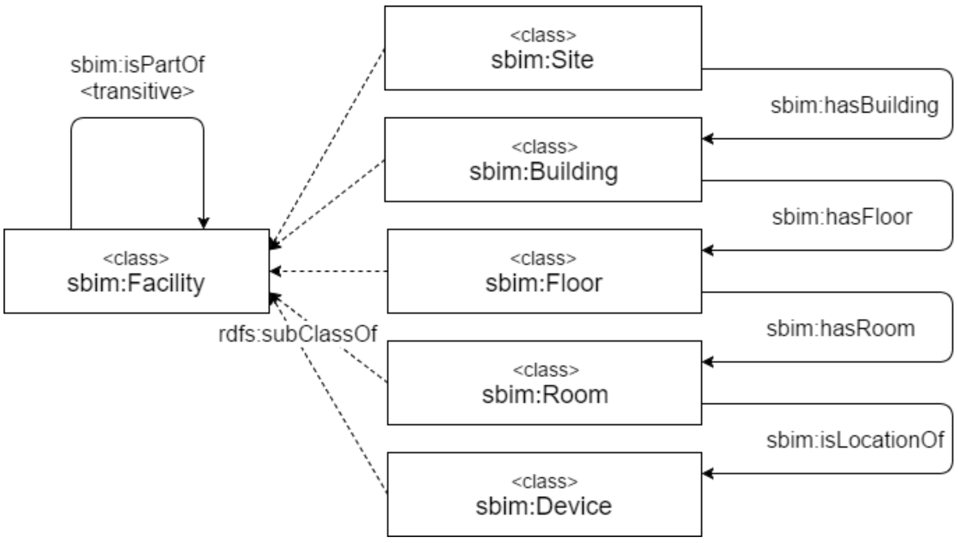


Figure 6.6: Representation of BIM elements in the ontology.

As stated above, individuals of `sbim:Device` can represent both the data source (i.e. `sbms:Sensing Device`) and the object described by them (i.e. `sbms:Feature of Interest`). Individuals representing locations can be described by particular data, acting thus as `sbms:Feature of Interest`.

The SBMS represents spatial relations within the built environment as tree hierarchy ("Site – Building – Floor – Room"). The BIM elements represented in the SBMS ontology are interconnected by properties that form the tree hierarchy, i.e. "a room on a floor" or "a floor in a building". However, the ontology facilitates a generic relation `sbim:isPartOf` (sub-property of `dul:isPartOf` with restricted range and domain) that is defined as transitive, thus creating indirect relations such as between a room and a building, skipping relation to a floor (the room is a part of the building if the room is located on the floor that is located in the building). Transitive relations are inferred by a reasoning engine and are used for certain queries that are provided by the Semantic BMS application interfaces (namely, the grouping of results as described in Section 6.4).

The concept of Site is not always present in the BIM representations of a facility, as the model often describes one project on one site (However, it is present in the IFC specification). In the field of FMIS and CAFM, the concept of the site is well-established, as large organizations often operate in multiple locations. Facility managers then can compare sites among each other or filter the data.

The equipment installed in the building is represented in the Semantic BMS ontology as well. The class `sbim:Device` serves as a root for a sub-class tree derived from the IFC4 specification where different types of devices (called Elements in the IFC) are defined in a hierarchy. Individuals of those sub-classes then represent individual devices that take part in the building automation. The hierarchy of device types is used during querying to limit the results to those coming from a certain type of device (e.g. temperature sensor) or describing a certain type of device (e.g. pump or fan where operational state is monitored). The device type hierarchy is not shown in the Figure 6.6 due to space and clarity considerations.

Each individual in the ontology which has its counterpart in the BIM has datatype property `sbim:hasBIMId` containing ID of the individual in the BIM system. This property is used for linking the representation in the SBMS with the original data source in the BIM – it serves as a bridge between the RDF graph and systems that uses different technologies, such as relational databases or different modeling languages. The property is enforced for each such individual by the `owl:hasKey` clause in the class definition. The string literal is used for storing the BIM ID instead of the resource identifier itself, as there are certain requirements and limitations to the format of a URI that could collide with the format of the BIM ID.

The SBMS thus requires translation of certain part of the BIM database to the OWL or another ontology language. At this point, only basic attributes are required by the ontology. Complex description of locations and devices is stored in the BIM. The SBMS is not aimed to be primary storage for the BIM data. Instead, the ontology repository duplicates BIM data and is updated whenever source data change. This approach was chosen because it allows for efficient retrieval of frequent queries ("get all temperature sensors from the first floor of the building B2") and keeps the semantic model and BIM-to-OWL translation relatively simple. For the translation, SimpleBIM approach presented in [87] could be adopted.

The Zone elements (i.e. groups of rooms – or generally spaces, as defined by the IFC) are not currently modeled in the Semantic BMS ontology, as there are no use cases that would facilitate data points that are related to zones. However, such data points exist (e.g. data points describing a state of a security zone spread across several rooms) and the use cases might emerge in the future and they will require adding the Zone concept to the model.

The representations of BIM individuals are modeled as subclasses of classes defined in the BOT ontology (mentioned in the Related work overview) using the `rdfs:subClassOf` statements. The BOT ontology serves for identical purposes but lacks some of the features described above (representation of the site, universal transitive `isPartOf` property and representation of device types). Exploration of possibilities of deeper employment of the W3C BOT ontology is a subject of further work, as well as possible alignment with the ifcOWL (see [86]).

6.3.3 Semantic description of BMS Addresses

The SBMS ontology adds representation of an address that publishes observed data in the BMS. Individuals of the `sbms:Address` class represent objects that can be addressed in the BMS (i.e. they have unique ID). Each individual of the class is required to have a data property `sbms:hasBMSId` similar to `sbim:hasBIMId` described in the previous section.

The `sbms:Address` is root class for all the objects available in the BMS. Further, following subclasses of `sbms:Address` are defined (see Figure 6.7; inverse properties are omitted):

- `sbms:DataPoint` – Represents reading from a sensor, value of an output (actuator) or other value that is usually a scalar and represent current state of the system. The data points can be of following types:

  - `sbms:Input` – A reading from a sensor;
  - `sbms:Output` – A value sent to an actuator;
  - `sbms:User Defined` – A parameter of an regulation or automation algorithm that can be changed by users (e.g. set point value of air conditioning)

- `sbms:History` – An address that stores historical values provided by a particular data point;
- `sbms:Algorithm` – A process that controls behavior of installed equipment; utilizes inputs, outputs and user defined parameters.

The subclasses serve as a device to differentiate types of addresses in the BMS when querying the ontology for required data. There are situations when for one property, there are multiple addresses that are somehow related to it and the class tree allows to assign the correct semantic meaning to them. Such example is a room temperature. Each In-Room-Controller usually publishes two values in the BMS: actual temperature in the room and the required value (Set-point). These two values have all the other attributes identical. They both come from the same sensing device, have the same feature of interest and observed property. The address types differentiate them, as the actual temperature is of type `sbms:Input` and the set-point is of type `sbms:User Defined`. Also, there can be addresses that can be accessed to retrieve historical data for the data points, which are individuals of the `sbms:History` class.

The reason to introduce the concept of `sbms:Address` is that there is a difference between the concepts of device in the BIM and the BMS. Certain devices are represented in the BIM as independent devices, but they are primitive and do not directly communicate within the BMS. An example of such device is a temperature sensor, which is a simple thermistor in casing, connected to Programmable Logical Controller (PLC). The sensor is represented in the BIM as a separate entity, but the data are accessible via the PLC. Additionally, other

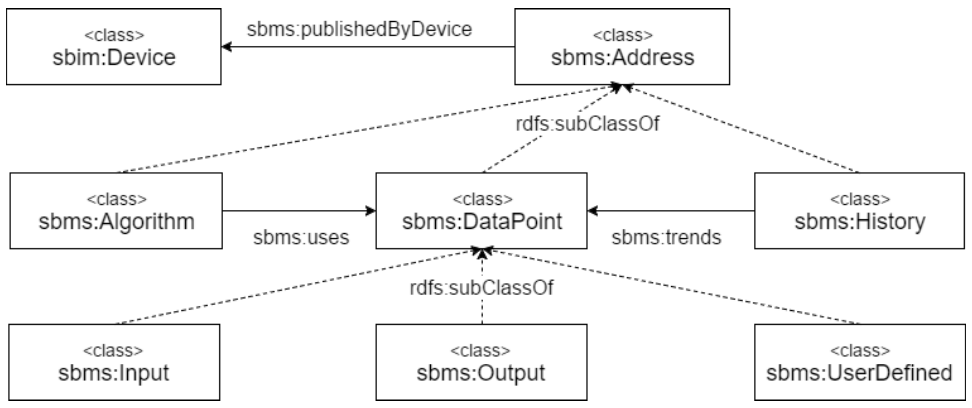


Figure 6.7: Representation of BMS elements in the ontology.

devices publish multiple data points related to their operation. An example of such device is a Variable-Frequency Drive (AC Drive) that offers various data about its operation (e.g. fan speed, input power, operation time).

6.3.4 Semantic description of Property types

The concept of Observed property is represented by `ssn:Property` class in the SSN. This class has no properties and restrictions. For the purposes of the Semantic BMS, and with respect to needed querying capabilities, the concept of the Observed property is structured to separately describe a physical quantity and a property domain (see Figure 6.8).

Each individual of the `sbms:Property` class has two properties: `sbms:hasPhysicalQuality` and a `sbms:hasPropertyDomain`.

The physical quality is represented by individuals of the `ucum:Physical quality` class that are part of the OWL transcription of the Unified Code Of Units For Measurement (UCUM)[1]. The UCUM ontology also provides additional properties to the physical qualities, such as units of measure, as defined by the UCUM.

Together with a physical quantity (e.g. energy, temperature,…), a property domain (`sbms:Property Domain`) is also defined. Individuals

1. Avilable from `http://idi.fundacionctic.org/muo/ucum-instances.html` and `http://purl.oclc.org/NET/muo/ucum/`

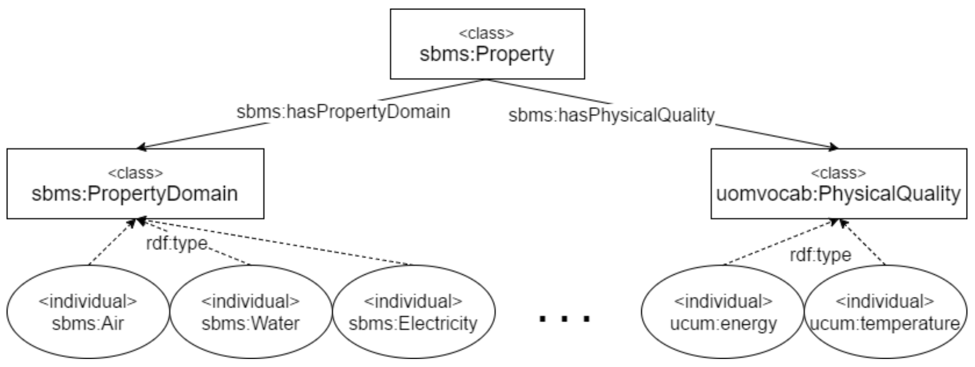


Figure 6.8: Semantic description of Property types.

in this class are for example `sbms:Air` or `sbms:Water`, allowing for a distinction of water temperature and air temperature. The measured quantity is the temperature in both cases, property domain specifies environment that is observed.

This distinction is needed in the field of building automation to filter e.g. only air temperature or only water temperature sensors when querying the Semantic BMS ontology.

6.3.5 Sensing type

Most of the data points represent simple sensor reading at given time. However, there are certain data points that provide more complex information and that are commonly used in the building automation. To distinguish such data points from direct sensor reading and also between different types of such "complex" data points, the `sbms:Sensing` class is introduced in the ontology.

Typical examples of such complex data points are those that describe energy consumption. By Energy consumption, we mean a value that has implicit temporal aspect as it measures consumption over some time period (e.g. last month, current year, current day). That distinguishes it from input power that senses current consumption (thus, energy consumption can be computed as integration of input power function over a given time period). To properly interpret energy consumption, the observer needs to be aware of two parameters – the consumption value itself and a time period the value describes.

Another example of such complex measurement is outside temperature that is computed as average from different sensors across the whole site. In other cases, minimal or maximal values from multiple sensors are provided by the BMS.

The distinction of above-mentioned cases (especially different time windows related to the data points) is crucial for meaningful querying of the ontology. The Sensing attribute allows a user to specify which data are of interest – e.g. consumption over a month or over a year.

The Semantic BMS ontology specifies several sensing types (subclasses of the `sbms:Sensing`) that are typical for the building automation domain (see Figure 6.9). First, stateless and stateful sensing can be distinguished.

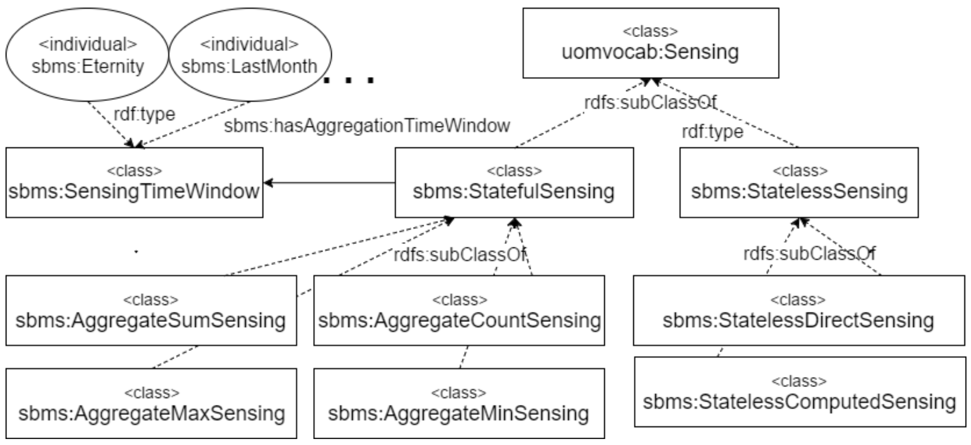


Figure 6.9: Semantic description of sensing methods.

Stateless sensing produces a value which is independent of previous values measured by the sensor (e.g. temperature sensor). Further, stateless sensing can be direct (e.g. value from one temperature sensor) or computed (e.g. average temperature computed from several sensors).

Stateful sensing derives present value from historical values measured by the same sensor. A typical example of stateful sensing is energy consumption metering as described in the above text. Other cases are data points that measure a maximal value of certain quantity over some time window (e.g. peak consumption in last 15 minutes). For stateful sensing, aggregation function (e.g. sum, total, average, minimum, maximum) and time window can be specified. The time window does not describe any specific time period framed by the start timestamp and the end timestamp. Instead, it describes duration – e.g. month or year. This is possible because the SBMS ontology describes only the data points, not the measured values. The time period described by the value changes over time (the data point provides consumption over last month, so the described period naturally changes each month), but the duration stays (almost) the same – calendar month.

6.3.6 Influences among data points

The Semantic BMS Ontology introduces simplified model of influences between variables in the BAS system. The ontology does not aim to semantically describe algorithms. It is limited to linking input and output data points to programs implemented in components of the BAS/BMS as described in the Section 6.3.3.

That way, it is possible to capture association of input and outputs – room temperature and set-point data points representing room temperature and set-point value related to a specific room serve as inputs for regulation algorithm of a respective AC unit, thus influencing values of various data points that the algorithm controls (e. g. fan speed, temperature of the air supply, or state of the valve).

Furthermore, the ontology describes “indirect” influence. Indirect influences are not implemented in regulation and control algorithms. Instead, they are observed as a result of physical processes (usually heat transfers) occurring in the built environment. The indirect influence usually occurs between output of one algorithm and input of another.

As an example, we consider a data point A representing openness of a valve that mixes cold and hot water (see Figure 6.10. The state of the valve is controlled by a data point acting as an output of an algorithm. Next, there is a data point B representing water temperature past the valve (temperature of the water mixture). Thus, the data point A value indirectly influences the data point B value – the measured temperature depends on the state of the valve.

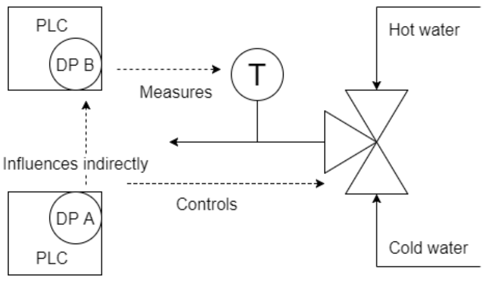


Figure 6.10: Indirect influence.

6.3.7 Semantic BMS graph querying

Since the OWL is based on Resource Definition Framework (RDF), every OWL graph can be queried using the SPARQL Protocol and RDF Query Language (SPARQL being a recursive acronym).

An example of SPARQL query follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
PREFIX sbim: <http://is.muni.cz/www/255658/sbms/v2_0/SemanticBIM#>
PREFIX sbms: <http://is.muni.cz/www/255658/sbms/v2_0/SemanticBMS#>
PREFIX sbmsd: <http://is.muni.cz/www/255658/sbms/v2_0/SemanticBMSData#>

SELECT *
WHERE
{ ?datapoint sbms:hasBMSId ?bmsId;
              a ?type.
  ?type rdfs:subClassOf sbms:DataPoint
  FILTER ((?type != sbms:DataPoint)
    && NOT EXISTS {?subtype rdfs:subClassOf ?type
    FILTER (((?subtype != ?type)
      && (?subtype != sbms:DataPoint))
      && (?subtype != owl:Nothing))
  })
```

```
?datapoint sbms:expressesObservation ?obs;
           sbms:hasBMSId "11304.AI3".
?obs sbms:observedBy ?source.
?source sbim:hasBIMId ?sourcebimId;
        a ?sourcetype.
?sourcetype rdfs:subClassOf sbim:Device
FILTER ((?sourcetype != sbms:Source)
  && NOT EXISTS {?subtype rdfs:subClassOf ?sourcetype
  FILTER (((?subtype != ?sourcetype)
    && (?subtype != sbms:Source))
    && (?subtype != owl:Nothing))
})
?source sbim:hasInstallationInRoom ?sourcelocationF.
?sourcelocationF sbim:hasBIMId ?sourcelocation.
?obs sbms:featureOfInterest ?scope.
?scope sbim:hasBIMId ?scopebimId;
       a ?scopetype.
?scopetype rdfs:subClassOf dul:PhysicalObject
FILTER ((?scopetype != sbms:Scope)
  && NOT EXISTS { ?subtype rdfs:subClassOf ?scopetype
  FILTER (((?subtype != ?scopetype)
  && (?subtype != sbms:Scope))
  && (?subtype != owl:Nothing ))
})
OPTIONAL {?scope sbim:hasInstallationLocation ?scopeLocF.
  ?scopeLocF sbim:hasBIMId ?scopelocation}
?obs sbms:sensingMethodUsed ?sensing.
?sensing a ?sensingtype.
?sensingtype rdfs:subClassOf sbms:Sensing
FILTER ((?sensingtype != sbms:Sensing)
  && NOT EXISTS {?subtype rdfs:subClassOf ?sensingtype
  FILTER (((?subtype != ?sensingtype)
  && (?subtype != sbms:Sensing))
  && (?subtype != owl:Nothing ))
})
OPTIONAL {?sensing sbms:hasAggregationTimeWindow ?sensingwindow}
?obs sbms:observedProperty ?property.
?property sbms:hasPhysicalQuality ?propertyquality;
          sbms:hasPropertyDomain ?propertydomain.
?datapoint sbms:publishedByDevice ?publisher.
```

```
  ?publisher sbim:hasBIMId ?publisherbimId
}
```

The query obtains all available information stored in the Semantic BMS repository about a particular data point. A typical use case for such query is a service available from the BMS user interface that provides a user with the available semantic information about a certain data point that the user encounters in the BMS user interface. This query corresponds to the test case 1 in the Section 7.6.

There is a sample data set prepared for the ontology.[2] The sample data contain only the asserted (ABox) triples. Before a meaningful query can be executed, inferred graph has to be constructed from the explicit assertions by the reasoning engine. For more details on inference over the Semantic BMS ontology, refer to Section 6.4.4.

Results of SPARQL query can be returned in various serialization formats, such as SPARQL-JSON. Due to the complexity of both SPARQL queries and SPARQL-JSON responses, a middleware layer containing the Semantic API is proposed. The following section describes the Semantic API in more detail.

6.4 Semantic API

The Semantic API provides interface for querying of the semantic model defined by the Semantic BMS Ontology. The ontology itself can be queried using SPARQL language, however this is not a common knowledge among software developers. Since the aim of the Semantic BMS project is to allow integration of the BAS data into powerful end-user applications, the need of SPARQL knowledge would constitute major obstacle to reaching the goal. Instead, the Semantic API provides RESTful API that communicates using JSON. The selection of used technologies and protocols allows the developers to select programming languages and frameworks according to their needs and likes. However, it also provides the "pure" SPARQL endpoint that allows developers to communicate directly with the repository if they desire to do so.

2. Available from `https://gitlab.fi.muni.cz/xkucer16/semanticBMS/tree/master/TestBench`.

It is worth noting, that the Semantic BMS Ontology as described in the previous section defines only the *TBox* of the model (terminological of the component, i.e. structure of the model). The *ABox* (assertion component – actual data contained in the model) must be stored in a data store that can be queried. The Semantic BMS architecture contains an ontology repository designated to storing of the *ABox data*. The Semantic API then communicates with the repository, manages inference process based on the assertions (see Section 6.4.4), and queries the resulting model (*ABox* data extended by assertions and inferred knowledge provided by the *TBox* model).

The Semantic API provides only the semantic information about data points that is available in the Semantic BMS Ontology. That means, the Semantic API does not provide the measurements itself. The aim of the API is to identify sensors and actuators that meet certain criteria specified by a query. The API simplifies the method of querying the ontology model, however it does not simplify the model itself. It does not obfuscate the model and every relation contained can be explored by the API. Fort more effective queries, SPARQL endpoint is available.

The API was developed with respect to the *EN 15221-7 Facility Management - Part 7: Guidelines for Performance Benchmarking* standard in order to support typical queries used for building operation benchmarking.

The API provides following “endpoints” (interfaces):

- Data points – Serves for obtaining semantic attributes of data points and for finding data points that satisfy criteria defined in the API call.

- Trends – Provides similar functionality as the Data points endpoint but focuses on storages of historical data (usually, historical data are available at different locations then present values).

- Relations – Allows for browsing through process relations (see Section 6.3.6) between elements of the BMS and BIM.

- Types – Provides unauthenticated and unauthorised access to the registers defined in the ontology (device types, data point types, physical qualities,...).

- Query – Allows for direct querying the ontology repository using SPARQL language. Results are formatted as Comma Separated Values (CSV) or as W3C SPARQL JSON.

6.4.1 API querying

As an example of API goals and capabilities, the "Data points" endpoint will be discussed. The endpoint allows consumer to query the ontology repository for data point IDs and filter them according to various criteria. Consumers can also specify additional attributes they wish to receive besides the data point IDs. Table 6.1 summarizes possible filtering criteria (marked by X in the "Filtering" column), additional attributes that can be obtained together with the BMS ID of a data point in the query result (column "Result"). The attributes correspond with the properties defined in the ontology model, with two notable exceptions. The `sbms:FeatureOfInterest` is renamed to `Scope` and the `sbms:SensingDevice` is renamed to the `Source` in order to provide more universally understandable terms for the facility management domain and to avoid possible confusion with other widely used terms in the fields of facility management, building information modeling, and building automation.

Based on the applied filters and required attributes, the SPARQL query is constructed by the Semantic API. The SPARQL query is then executed on the ontology model maintained by the Semantic API. The returned results are then converted to the JSON format. The query specification, as well as the JSON response, respect the object-like notation that is outlined in the Table 6.1, using traditional "dot notation" to represent hierarchy of objects and their properties. Example of the query is provided in the Section 6.4.2.

Results grouping

An additional feature of the Semantic API is data point grouping. This capability can be facilitated in many benchmarking use cases where aggregations for certain subsets of a facility are computed. Such subsets might represent floors, building, or the whole sites, or certain device types, or different qualities.

Table 6.1: Data points endpoint parameters.

Entity	Component	Primary key	Filtering	Result	Grouping
Datapoint	BMS ID	X	X		
	Type		X	X	X
Source	BIM ID		X	X	
	Type		X	X	
	Location		X	X	X
Scope	BIM ID		X	X	
	Type		X	X	X
	Location		X	X	X
Sensing	Type		X	X	
	Window		X	X	
Property	Quality		X	X	
	Domain		X	X	
Influence	Scope		X		
	P. Quality		X		
	P. Domain		X		

The Semantic API is capable of returning the results grouped by the common attribute value (i.e. the same floor where the measurement takes place). The groups can than be readily processed by the Data Access API that either returns the raw data or directly computes the desired aggregation over the data point group.

Table 6.1 shows parameters that are usable for result grouping. As an example, consider the location parameters (Scope location/Source location). Both can be grouped on the desired level of hierarchy (site, building, floor, room). Query results are then separated to respective groups in order to be easily used as an input in aggregate computations such as “average temperature for each floor in particular building”.

This feature of the Semantic API cannot be fully implemented using the SPARQL language and must be partially performed in the code of the API. However, the implementation heavily uses the inference capabilities of the ontology languages (see the sections 6.3.2 and 6.4.4 for details).

6.4.2 Usage example

When using the API, a query to the Semantic API is constructed. Example of such query follows:

```
GET https://.../sbms/semantics/datapoints/
?fields=bmsId,type,source.bimId,source.type,scope.bimId,
scope.type,sensing.type,sensing.window,publisher.bimId
&grouping=noGrouping
&source.type=TemperatureSensor
&property.domain=Air
&property.quality=temperature
```

The API responds with the following JSON:

```
{
  "results": {
    "noGrouping": [
      {
        "bmsId": "11304.AI3",
        "type": "Input",
        "publisher": {
          "bimId": "BHA14N01013MAKB001"
        },
        "source": {
          "bimId": "BHA14N01012MABT001",
          "type": "TemperatureSensor"
        },
        "scope": {
          "bimId": "BHA14N01012",
          "type": "Room"
        },
        "sensing": {
          "type": "StatelessDirectSensing"
        }
      }
    ]
  },
  "groups": [
    "noGrouping"
  ]
}
```

The response presents following information regarding a data point with BMS address of `11304.AI3`:

- Data point type – Specifies the role of the data point in the BMS. Possible types can be Input, Output, or User-defined value (see subsection 6.3.3).
- Data source (Sensing device) – Specifies BIM Id of the device that provides the BMS with the data.
- Source device type – Describes the type of the sensing device (e.g. Flow meter).
- Scope (Feature of Interest) – A scope specifies object the data point value is related to. Possible scopes are a location (e.g. building when measuring energy consumption) or a device (e.g. fan in the case of data point representing fan speed).
- Scope type – Specifies if the scope is a site, building, floor, room or a device.
- Sensing method – Different data point implement different sensing methods. The simplest case is direct sensing (e.g. room temperature). Other (indirect) sensing methods employ some kind of computation or aggregation over some time period (e.g. energy consumption total for the last month).

Presented examples of the Semantic API capabilities aim to show how the data stored in Semantic BMS ontology can be queried using convenient and easy to implement methods, thus largely simplifying development of various analytical and decision support application for the field of building operation analysis.

6.4.3 Design

This section aims to describe design of the Semantic API and used technologies.

The Semantic API is written in the Java programming language (specifically, Java 8) and relies purely on open source and freely available (in a sense of a price – however, the license term naturally applies) technologies, framework and libraries.

The Semantic API relies on Apache Jena framework for the OWL-related tasks. From the Apache Jena, following tools are used:

- TDB – Native triple store, NoSQL database for storing RDF graphs. The Semantic API uses the TDB for storing the *ABox* data.
- RDF/XML parser – Used for accessing the Semantic BMS Ontology itself stored in the XML files (the *TBox*).
- TDB API – Allows access to the TDB using the Java language.
- ARQ – SPARQL query engine is used for query composition and execution. ARQ uses object-oriented approach for query construction that provides performance boost when compared to primitive "String concatenation based" approach.
- Reasoner – The Apache Jena provides variety of reasoners that are able to derive inferred knowledge from the *TBox* and *ABox* assertions.
- OWL Model – Combining the *TBox*, *ABox* (from the TDB) and the reasoner, the inferred model can be created by the Apache Jena and used for querying by the ARQ.

At the moment, the SPARQL query construction is static – meaning that the template of the query and the order of the clauses is generally the same regardless of the query being executed. This results in sub-optimal execution times for certain types of queries (refer to Section 7.6 for more details). The current implementation favors the more often and simpler, faster queries in expense to the complex ones that are not expected to be executed often. The dynamic query construction that would dynamically reorder the clauses depending on the specific query and possibly some heuristic approaches (e.g. execution times of previous queries) are subject to further research.

For the RESTful API, Oracle Jersey project was used. The Jersey implements the JAX-RS interfaces and serves as a JAXD-RS reference implementation.

The Semantic API complies to the Servlet 3.0 specification an is intended and tested to be deployed to Apache Tomcat (version 8.5 and higher) or another Servlet 3.0 container.

6.4.4 Inference process

Before querying the ontology, inference of the explicit assertions has to be performed to derive the implicit relations defined by class and property hierarchies defined by the OWL. In the Semantic BMS, the inference is performed by the Apache Jena framework during the application startup or after the data modification is performed by the user.

Apache Jena provides a variety of reasoners that can be used for inference. Simpler reasoner will usually ignore certain restrictions provided by the OWL language, resulting in a simpler model that is created faster and also leads to faster querying. On the other hand, it does not allow for strict validation of the model (since some restrictions are ignored) and complex queries. Selection of appropriate reasoner is crucial to reach a suitable trade-off between model complexity and query performance.

For the purposes of the Semantic API, the fastest and most limited OWL reasoner (`OWLMIcroReasoner`) performing only RDFS reasoning, proved sufficient for the purposes of querying. This combination allows for fast model construction (several seconds) and sufficient query performance (see Section 7.6).

There are obvious limitations caused by this approach. Limited reasoning capabilities do not allow to thoroughly check the validity of the ontology model after each data update. The Semantic API is capable of checking for model consistency on the application logic level. However, if invalid data were added to the ontology directly, by bypassing the provided API methods, invalid data could remain unnoticed in the ontology until in-depth validation by more complex reasoner is performed.

6.5 Data access API

The data access API ensures communication among end-user applications and underlying data sources – namely building automation networks and data warehouses that store archive data about building operation. It provides an abstraction layer that hides implementation details of the automation protocols and database schemas and publishes the data via the unified interface. Further, the API provides basic data processing tasks, such as sampling and interpolation of the data or simple data aggregations. The capabilities of the Data Access API are comparable to the Extract and Transform phases of the ETL workflow established in the traditional BI and OLAP systems. The Data Access API is designed as an extensible platform for integration of different *data providers* that retrieve data from multiple data sources. However, the Data Access API does not provide the implementations of the data providers.

The following sections cover the Data Access API in more detail.

6.5.1 Role in the Semantic BMS

The Data Access API is used during the second step of the data retrieval process as described in Section 6.2. In the first step, data points of interest are identified using the Semantic API. Next, the data points (network addresses) are passed to the Data Access API together with additional parameters that fully specify the data request. The Data Access API then obtains required observed values and processes them.

The two-step approach brings multiple advantages. First, it allows to save results of the semantic query and use it repeatedly to obtain the data from the same data point in different points in time. A typical use case is a periodically executed report. The semantic query (the first step) is executed once, during the report design. The specification of the data point set becomes a part of the report definition. During the periodical query execution, only the Data Access API call takes place. This brings a higher performance of the querying process, as the calls to the semantic model are generally rather expensive, especially for large numbers of data points (refer to the Section 7.6).

Second, it allows enriching the query in between the two steps. The enrichment comprises of adding additional attributes to the data point list. The added attributes can be further used during the data processing that is performed by the Data Access API. An example of a computation that facilitates the enriched data point specification is a computation of average temperature in a building. The Semantic API call retrieves a list of temperature sensors, each of them located in a different room. To obtain accurate characteristics of the building temperature, the size of the room might be required to be considered (e.g. in cases when sizes of the rooms significantly differ). The Semantic

API can assign each data point to the installation room. However, it does not provide information about the room size, as this type of data is out of the scope of the semantic model. However, the area (or volume) of the rooms can be obtained from the BIM system, based on the room IDs provided by the Semantic API. The query for the Data Access API can be extended by the data retrieved from BIM, adding the room area or volume as an attribute to each of the data points. The Data Access API uses the areas or volumes to compute the weighted average of the temperatures, using the room size to assess the weight of a temperature value.

6.5.2 Types of data sources

The Data Access API distinguishes two categories of data sources:

- Addresses that publish the present value of the observed property at the time of the query. Such addresses are represented as individuals of the `sbms:DataPoint` class in the Semantic model – see Section 6.3.3. The data point is directly connected to the sensor or actuator and instantly publishes current state of the device.

- Addresses that publish archival (historical/trend) data of the observed property. Such addresses are represented as individuals of the `sbms:History` class in the Semantic model – see Section 6.3.3. The history address represents a data store that collects measurements from the specific data point over time according to a certain set of rules.

This division does not specify which access method is used for data retrieval. Usually, the building automation network contains both types of the data. Additionally, certain historical data can be usually accessed via different access methods.

For the use case of Masaryk University and the BACnet protocols, there are `TrendLog` objects accessible via the BACnet protocol, that collect and store observed values in time as records with a timestamp. Additionally, historical data are stored in the Historian relational database which can be accessed and queried via the SQL.

The `TrendLog` objects are usually located directly on the automation devices (e.g. PLCs). However, the `TrendLog` objects have some drawbacks. When located on the PLC, the TrendLog has a fixed capacity that is limited by a memory size of a PLC. Second, accessing the data via the BACnet protocol provides only basic querying options and limited query performance caused by limits of the automation devices and used transport media.

The Historian database has no fixed size limit and provides superior query performance and capabilities. On the other hand, only a subset of all the available data points is stored in the archive database due to the performance and capacity considerations. As a result, for certain observed properties, the full history is not available.

This use case illustrates the second kind of categorization of data sources based on the access protocol. The typical use cases are:

- Addresses available via the open automation protocol stack – This scenario applies for automation networks that communicate via open, standardized protocol such as BACnet, KNX, LonWorks, or Modbus. For such protocols, Java implementation of the protocol stack is available.
- Addresses available through relation databases and SQL – This scenario applies for many permanent archive data stores, historian servers and OLAP warehouses. In case they provide a direct connection to the relational database, SQL query language can be used for retrieving the data. The SQL can be also used for data processing, as it provides support for data joining and aggregations. The general rule is that the aggregations should be performed by the database engine rather than by the Data Access API due to the performance considerations.
- Addresses available via the inter-process communication – This is an option for closed protocols and protocols, where protocol stack in Java is not available. However, this is not expected to be a frequent scenario. In the field of building automation, this scenario is a case when using the OPC (Open Platform Communications, see Section 4.5).
- Addresses available via web technologies – This scenario applies to situations where there is a gateway installed in the automation

network, possibly a BMS user interface web server that also provides programming interfaces (APIs). It is also applicable for various archival servers that provide a custom API instead of direct access to the database.

The Data Access API aims to abstract the users of the API from the different access methods, as it provides the common API and functions regardless of the data origin. However, it distinguishes between the present values and archival data, as the variety of data processing methods is much greater for the archival data.

6.5.3 Data processing

Besides the simple data acquisition via the unified interface, the Data Access API provides also a variety of data processing options. This functionality is needed since the raw data originating from the building automation systems are often not ready and fit for instant use in analytical scenarios. The data processing fulfills a similar role as the Transform phase in the ETL workflow. The goal of the data processing is to prepare the data for further analysis.

The Data Access API supports a wide range of use cases. From straightforward data retrieval that passes the raw data to the application, to ready-to-use complex aggregates. The first case is suitable for complex BI applications that need to provide complete drill-down to the source data, data mining tasks, or advanced analyses. The latter use case is suitable for dashboard views, portlets, mobile applications, tenant portals, and other simple components where visualization of an aggregate characteristics is sufficient.

The Data Access API serves primarily as a guarantee of the functionality provided for the end-user applications. The mechanism of data processing is left as a responsibility for the data providers that connect the Data Access API to the data sources. It is possible that some data providers will use pre-computed aggregates available in the data warehouses, although it is expected that most of the data processing will take place on-the-fly, during or after the data retrieval. This is likely the case for data providers that connect directly to the BAS to obtain data. As an example, BACnet protocol supports `TrendLog` objects. However, to perform aggregation or sampling, all the data records have to be transferred from the BAS and then processed by the data provider, as the BACnet itself does not provide tools for such data processing.

The performance of this approach might prove insufficient in some scenarios (however, this is rather a feature of the BAS than the Data Access API itself). For that reason, archive databases are considered more appropriate data sources for querying and processing larger amounts of data. In such scenarios, the data processing can be at least partially performed by the database engine.

The Data Access API is a RESTful API that provides the data in two formats – JSON and Comma separated values (CSV) for different uses. The JSON is more suitable in custom applications tailored specifically for use with the Semantic BMS. The CSV format is used when integrating the Semantic BMS with a general-purpose business intelligence, reporting, and analytical applications.

The Data Access API provides an endpoint that follows the distinction of the present and archival values as described in the previous section. The API is designed to accept results of the Semantic API calls as input parameters for data retrieval queries. For that reason, the Data Access API respects the address grouping performed by the Semantic API (see Section 6.4.1). The addresses in the query can be grouped and the data processing (e.g. aggregation) is then performed for each group separately. Due to this functionality, the API can provide multiple results in one query. This function is illustrated in the Section 6.2 and Figure 6.4.

A description of the API functions follows.

6.5.4 API functions

The endpoints aim to cover a wide variety of scenarios that can occur in building operation benchmarking, analysis, or visualization, and provides convenient methods that prepare data for different tasks.

For the present values, the most basic and simple query is a raw data retrieval. The most basic API call retrieves one value from the BAS network.

The API is also capable of receiving a list of data point addresses. The data provider reads the present values from the network and returns an unaltered list of the obtained values. The obtained values are regarded as having the exact same time stamp, even though the exact moment the value was retrieved from the BAS network will slightly differ among the data points.

The set of values from the automation network with the same time stamp is referred to as *Slice*. The Slice represents a state of a (subset of) the BAS network in given moment.

Furthermore, the API can compute an aggregate from the obtained values. The call to the API then contains list of the data points and specification of the aggregate function, resulting in one scalar value that represents the computed aggregate. The aggregate functions that can be used are described in the Section 6.5.5. The functions that are applicable for the Slice data structure.

Next, the API accounts for value grouping. In the request, the addresses can be divided into multiple groups. The API respects this division when dispatching the request. The API is capable of returning the raw reading from the BAS with the grouping preserved, or is capable of computing the aggregates for each of the groups separately. The example use case is a request that asks for average temperature on each of the floors of the building. The request will contain addresses of the temperature sensors, grouped by floor. The API will obtain the data from the network and compute the average from them, for each group separately. The result will contain average value for each of the groups, fulfilling the request.

For the archive data, the variety of API calls is significantly broader. The basic functionality is similar. The API is capable to obtain a list of values recorded by given data point or History address. The request must specify the address and two timestamps that frame the time period of interest. The API retrieves the data and returns a list of historical values ordered by their timestamps.

The data structure consisting of values equipped by their time stamp, ordered by the time of origin, is referred to as *Series*. The Series can be also aggregated using one of the aggregation functions available, resulting in a one scalar value returned that represent the aggregate of the source Series.

In the more complex use cases, multiple addresses are usually specified in the request. In such cases, grouping of the results is always supported and behaves the same ways as in the case of the present values.

Similarly as with the present values, there is an option to request multiple Series in one query, result being denoted as *Set of Series*. As in the case of a single Series, the Set of Series can be aggregated. The aggregation can be performed in two manners:

- Each of the Series is aggregated separately and the result contains a list of aggregations, each of them representing aggregate of one of the original Series. This can be used for computing average temperature for each room on a floor in a single query.
- In the first step, each Series is aggregated separately. In the second step, the resulting aggregates are aggregated once more using the same mechanism. A typical use case is finding a highest temperature that occurred in as building during a time period. In the request, addresses of the temperature sensor archives are specified. First, the maximal temperature for each room is detected. Next, from the resulting minimal values (results of the first aggregation), the global maximum is selected. Other use cases and exceptions to this rule are described in the Section 6.5.5.

The API is also capable to obtain a value from a data point at a certain moment in the past. However, often there will be no record stored for the exact moment requested. For that reason, the API performs *Interpolation*. Basic types of interpolation are supported – the API uses the closest earlier value available, the closest latter value available, or performs linear interpolation of the two closest values, taking into account their distance in time to the moment requested (see Figure 6.11).

If multiple addresses are specified together with the time stamp, a *Slice* (see above) is generated by the API, representing interpolated values in the given moment.

The Interpolation process is also employed when generating the *Sampled Series* data structure. The Sampled Series is a list of records ordered by a time stamp. However, The data are not transferred directly from the archive. Instead, they are sampled to form regular intervals (e.g. each hour, each midnight, every ten minutes) between the time stamps occurring in the resulting (Sampled) Series. The Data Access API uses the closest values in the Series to interpolate the value of

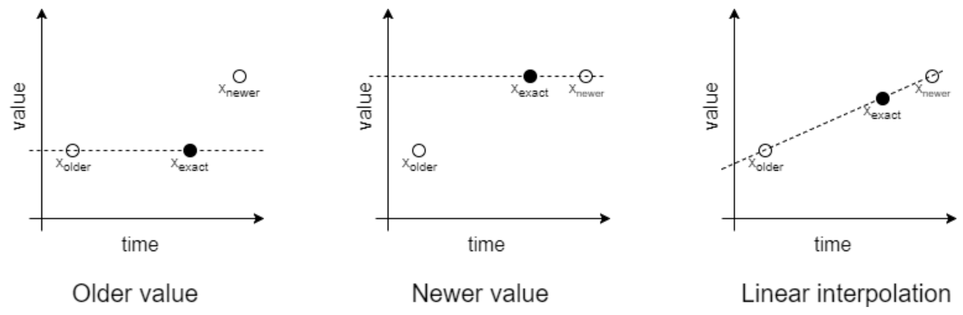


Figure 6.11: Types of interpolations.

the data point in the requested intervals. Figure 6.12 illustrates this mechanism.

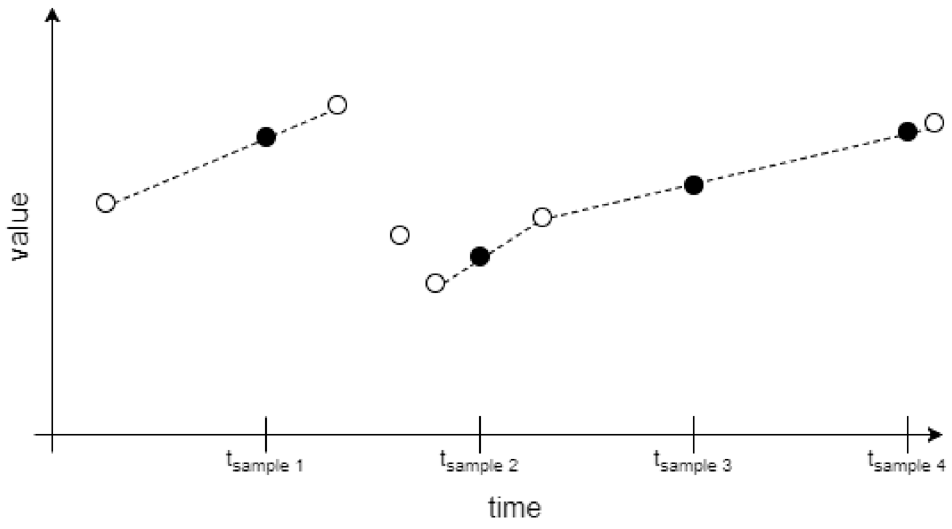


Figure 6.12: Sampling with linear interpolation.

This function is intended for use mainly when visualizing the data, as regular intervals between the values simplify visualization. This is even more useful for situations where multiple data points are visualized together or compared. In that case *Set of Sampled Series* can be provided by the API. In that case, all the time stamps will be synchronized among the Series in the set.

For the specification of intervals between the samples, two methods can be used. Either is the interval defined by a duration (e.g. in seconds), or by CRON-like expression that allow for complex configuration (e.g. each Monday and Friday at 8:15).

In many cases, it is more convenient to present the data available in the Set of Sampled Series in a different structure. The records with the identical time stamps (but from different addresses) form a Slice (see above). Therefore, they can be represented as a *Series of Slices* – each of the generated time stamps is accompanied by a set of key-value pairs (i.e. dictionary), key being the address and value the interpolated data. This structure thus groups together values with the same time stamp, as opposed to the Set of Sampled Series which groups together values form the same address.

A distinct feature of Series of Slices is that each of the Slices can be further processed by one of the aggregation function, resulting in a single Series of aggregated values. An aggregation of Slices can be used for computing the average temperature in a building in one hour intervals. Providing a list of addresses of temperature archives for the rooms in the building, the data are sampled in one hour intervals. Then, for each set of records for the given hour, the values are aggregated and the average is computed.

A combination of sampling and aggregation per Series is provided by *Series of Windows*. When using this method, time periods are defined in the request using start point, end point and either duration, or CRON expression to define size of the aggregation windows. Other components of the request are a specification of a desired aggregation function and an address(es). The Data Access API retrieves the data and computes the aggregation for each of the time windows separately. The results form a Series of aggregated values (e.g. list of average temperatures for each day of month, ordered by timestamps). The API is also able to produce *Set of Series of Windows* when passed multiple addresses in the request.

The Figure 6.13 illustrates differences between Set of Sampled Series, Series of Slices and Set of Series of Windows.

The user might also decide to perform additional aggregation over the *Set of Series of Windows*. The windows for the same time interval are treated as a *Slice* and aggregation is performed on them. A typical example of such request goes as follows: Retrieve all temperature

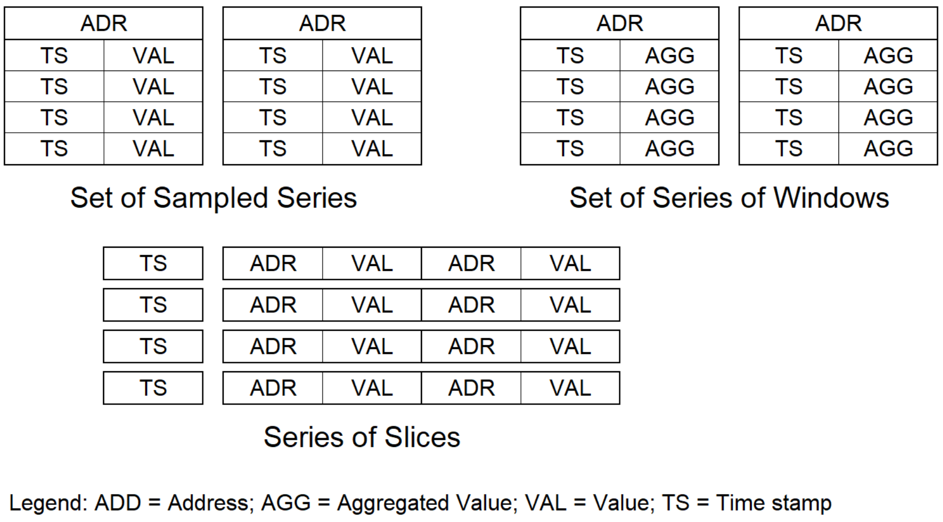

ADR
TS
VAL
AGG
Set of Sampled Series
Set of Series of Windows
Series of Slices
Legend: ADD = Address; AGG = Aggregated Value; VAL = Value; TS = Time stamp

Figure 6.13: Comparison of the Data Access API data structures.

readings from a building. For each of the sensors, compute a *Series of Windows* (e.g. average temperature for each day). Next, Compute aggregation over the windows with the same time period (e.g. overall average of a temperatures for the whole building for each day).

The Table 6.2 summarizes all the data structures provided by the API and options available for them.

6.5.5 Aggregations provided by the API

The Data Access API provides aggregation functions, resulting in a scalar value returned as a response to a list of data points passed in the request. The raw values can be aggregated using one of the built-in aggregation functions defined below. However, some of the functions are not available or meaningful for every type of data.

The following list provides detailed description of the aggregation functions provided by the Data Access API:

- Average – An arithmetic mean of the obtained values.
- Weighted average – Weighted mean of the obtained values. For this type aggregation, the request must contain also the weights for the individual data points. The weights do not have to sum

Table 6.2: Available API methods. Each row represents basic method specification. O denotes methods where grouping can be optionally specified.

Method	Input type	Temporal specs.	Grp.	Agg.	Samp.	Win.	Result
1	DP	None					Scalar
2	DPs	None	O				Slice
3	DPs	None	O	X			Aggregate
4	Trend	Interval					Series
5	Trend	Time stamp			X		Scalar
6	Trend	Interval			X		Sampled Series
7	Trend	Interval		X			Aggregate
8	Trend	Interval		X		X	Series of Windows
9	Trends	Time stamp	O				Slice
10	Trends	Time stamp	O	X			Aggregate
11	Trends	Interval	O				Set of Series
12	Trends	Interval	O	X			Set of Aggregates
13	Trends	Interval	O	X			Aggregate
14	Trends	Interval	O		X		Set of Sampled Series
15	Trends	Interval	O		X		Series of Slices
16	Trends	Interval	O	X	X		Series of Aggregates
17	Trends	Interval	O	X		X	Set of Series of Windows
18	Trends	Interval	O	X		X	Series of Aggregates

up to 1. The typical use-case for weighted average is described in the Section 6.5.1 – the room areas or volumes are used as a weights for a computation of an average temperature in a building.

- Temporal average – In this case, the weighted average is computed using the validity period of each value as its weight. The validity period is computed as a duration between the time stamp of the record in question and the time stamp of the record that follows. This approach allows to correctly process data points that are change-of-value driven. For such data points, a new value is stored in the archive when the difference from a last recorded value exceeds defined threshold. When examining such archive, records have different period of validity. Computing the average with equal weights for all the records would distort the results. By using the validity period as weight for the value, the resulting value characterizes the data more accurately.

- Weighted temporal average – This approach combines the weighted average (with the weights supplied in the request) and the weights computed from the validity periods of the records. First, the temporal average (see above) is computed for each of the series. In the second step, the weights from the request are combined with the resulting temporal averages to compute the resulting value. This aggregation can be used to obtain fitting characteristic of a temperature in a building over a certain time period, with respect to floor area of individual rooms and validity periods of recorded measurements.

- Sum – Sum of all the retrieved values.
- Maximum – Maximal value from the retrieved set.
- Minimum – Minimal value from the retrieved set.
- Median – Median value of the retrieved set.
- Difference – Computes the difference between the first and the last value in the data set. This function used with time Series when dealing with the data points that represent constantly rising total such as total energy consumption. To compute the consumption for a given time period, the first record in the time interval and the last record in the interval are subtracted.
- Count – Number of the records retrieved. Can be used for detection of frequent changes of a monitored value. For certain sensors and actuators, frequent changes are a sign of a faulty behavior of the system.
- Ratio – This aggregation is suitable for the binary data. It is a specific type of an average, where all the values are either 0 or 1. It computes a ratio between the data points that are active (with a value of 1) to the total number of the data points. The resulting value is computed as a sum of the values divided by the count of the values. A use case for this aggregation might be a situation when the user wants to examine extent of air conditioning use in a building. The API receives request that contains a list of binary data points, representing state of each of the air conditioning units. The resulting value tells the user

what's the ratio of the AC turned on to the *total number* of the AC drives installed.

- Weighted ratio – The weighted ratio accounts for different weights assigned in the request to the data points. Each value (which is either 0 or 1) is multiplied by its weight. Then, the sum of the weighted values is divided by the sum of the weights. The use case for this type of aggregation combines room areas with data points that signal if the room is currently air-conditioned. The weighted ratio then computes the fraction of the total floor area that is currently air-conditioned.
- Temporal ratio – This function behaves similarly to the temporal average, using the validity period as a weight of the value.
- Weighted temporal ratio – As in the previous case, the behavior is similar to the Weighted temporal average.
- Logical AND – Used exclusively with the binary data, this aggregation performs logical AND over the set of the obtained values. This aggregation could be used to check if all the lights in the building are switched off during a night.
- Logical OR – Used exclusively with the binary data, this aggregation performs logical OR over the set of the obtained values. This aggregation could be used to check if the building is empty or if there are still some occupants, using data from motion sensors or occupancy sensors in the rooms.

Generally, there are three main types of data structures that the aggregation is computed on: Slice, Series, Sample window (effectively just a series) and Set of Series, as described in the previous section. For most of the aggregations, there is no difference between Slice and Series when computing the aggregate value – the timestamps are ignored. When dealing with the Set of Series, the function is first applied individually on each of the Series. The results are then processed by the same function to retrieve the one resulting scalar. However, there are aggregates that are available only for the Series (e.g. the Temporal average) or for the Set of Series (e.g. Weighted temporal average). In the last case, the Temporal average is applied on each of the Series in the first step. In the second step, the Weighted average is applied on the aggregates resulting from the first step.

The Table 6.3 summarizes the types of aggregation and types of data that can be used (and are meaningful) with them.

Table 6.3: Usage of aggregation functions with different types of data. X stands for typical use. (X) stands for limited use that require specific use cases or setups.

	Slice	Series	Sampled series	Set of Series	Real values	Binary values
Average	X		X	(X)	X	
W average	X				X	
T average		X			X	
WT Average				X	X	
Sum	X	(X)	(X)	(X)	X	X
Max	X	X	X	X	X	
Min	X	X	X	X	X	
Median	X	(X)	(X)	(X)	X	
Difference		X	X		X	
Count		X		(X)	X	X
Ratio	X		X	(X)		X
W Ratio	X			X		X
T Ratio		X				X
WT Ratio				X		X
AND	X	X	X	X		X
OR	X	X	X	X		X

As stated above, the responsibility of the data processing is left to the Data Providers (see the Section 6.5.6). However, the Data Access API core provides basic, more or less naive implementations of the aggregation algorithms that can be readily used by the developers of the Data providers. The implementation of the algorithms does not employ parallelism and other techniques that would significantly improve performance. Moreover, a whole dataset has to be transferred to the Data Access API in order to compute the aggregation. This fact alone causes performance decline. The larger datasets are expected to be processed by a different subject, the data originator (e.g. database engine).

The aggregation functions described in this section are expected to cover majority of the use cases that occur in the domain of building operation analysis. Nevertheless there will certainly be use cases that will require more complex data processing. In such cases, the unaltered, complete data set must be obtained through the Data Access

API. The analysis can be then performed in the business intelligence tool.

6.5.6 Data providers

The Data Access API is designed as an extensible platform for integration of different *data providers* that retrieve data from multiple data sources. However, the Data Access API does not provide the implementations of the data providers. Such implementations are left for the users of the Semantic BMS project or for a future work. Instead, it offers unified interfaces that can be implemented by the data providers to communicate with the Data Access API. In conclusion, the Data Access API publishes the interface that can be used by the end-user applications and defines interfaces that must be implemented by the data providers on the other end of the data flow process. To prove usability of the API, reference implementations of the data providers are part of the Semantic BMS project (see Section 7.2).

The definitions of the data provider interfaces ensure that the data provider can provide the data and output structures that are necessary for the Data Access API to properly function, as defined in the Section 6.5.4.

The data providers are responsible for the data processing offered by the Data Access API to its consumers. The reason behind this design decision is that certain data providers (such as relational databases) are suitable to efficiently provide data processing and aggregation. Performing the processing by the database server is generally more efficient than transferring the data to the Data Access API and perform the processing in code.

However, this is not the case for all the data sources. Data providers that connect to the automation network are usually capable of obtaining only the raw data and the processing must be performed on the level of the data provider and the Data Access API. Since the data processing comprises several complex tasks, the Data Access API provides abstract implementations of the data providers that can be facilitated by the developers. The abstract classes provided in the Semantic BMS handle the data processing tasks so the developer of the data providers can focus purely on providing access to the data sources using following straightforward methods:

- Retrieve a present value for each of the BAS data point addresses from a list:
  ```
  Map<Address, RawValue> getValues (
      List<Address> dataPoints,
      boolean allowCache
  )
  ```
- Get all the available records for a given time period for all the addresses specified:
  ```
  Map<Address, Trend> getRecords (
    List<Address> trends,
      ZonedDateTime from,
      ZonedDateTime to
  )
  ```
- Get the closest older record than the specified timestamp for each of the addresses specified:
  ```
  Map<Address, Trend> getClosestOlderRecord (
  List<Address> trends,
      ZonedDateTime timestamp
  )
  ```
- Get the closest newer record than the specified timestamp for each of the addresses specified::
  ```
  Map<Address, Trend> getClosestNewerRecord (
  List<Address> trends,
  ZonedDateTime timestamp
  )
  ```

Depending on the type of the data source, the implementation provided in the abstract classes might or might not be used. In cases the data source (e.g. database server) allows performing the data processing tasks, overriding the methods implemented in the Data Access API will likely result in better performance. However, this could be considered premature optimization. Depending on the desired use cases, the default implementation might prove sufficient.

There are certain situations when the Data Access API must perform the data processing. Such scenarios occur when there is a request for an aggregation that spans through multiple data providers. In that case, the Data Access API must compute the aggregation based on the raw data provided by the individual data providers. Sample use case is a median of values that are accessible via different data providers. In that case, the processing must be performed on the merged data set.

For such situations, query routing mechanism (QRM) must be implemented in the Data Access API. The QRM decides if the desired aggregation can be computed solely by the data providers, needs additional processing on the level of the API, or must be computed from the merged raw data. Depending on the decision, the QRM then issues proper requests to the data providers.

The architecture of the Data Access API is presented in Figure 6.14.

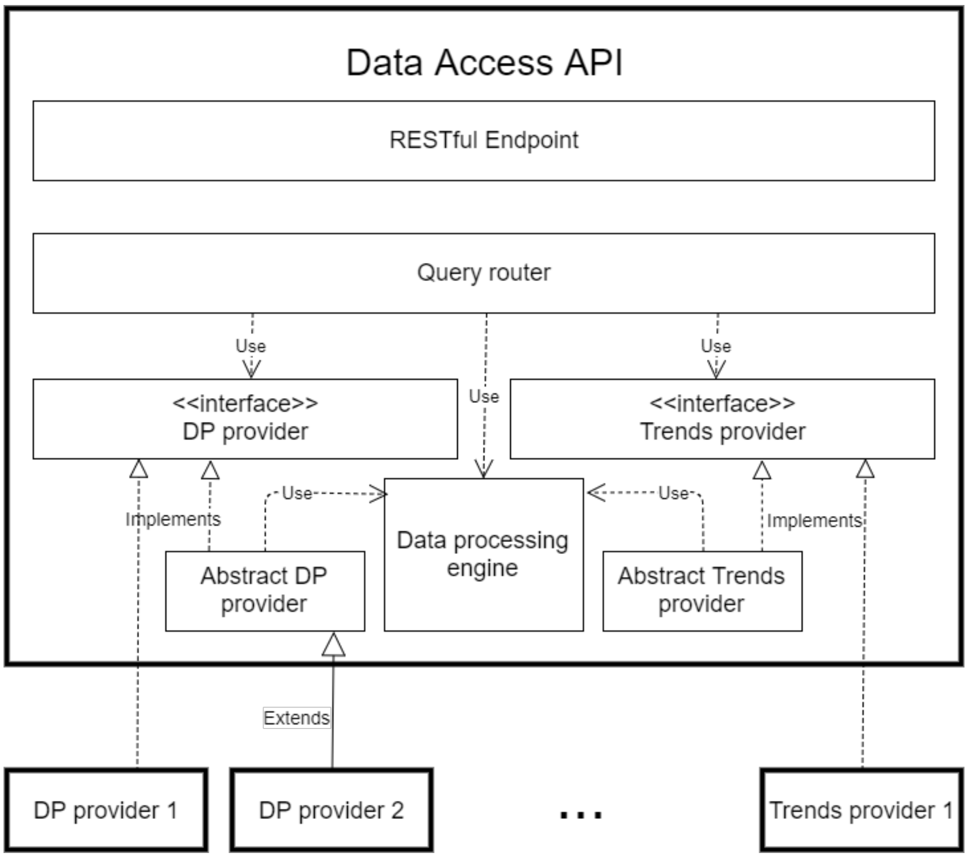


Figure 6.14: Architecture of the Data Access API.

6.5.7 Technologies

The selected languages, frameworks, and libraries were selected to be widely known (or at least well-documented), open source, and multi-platform. Such requirements are even more important for the Data Access API than for the Semantic API, as the Data Access API is expected to be extended by developers and users.

The Data Access API is implemented using Java 8 language. It is a RESTful API, using Oracle Jersey for a RESTful functionality. The Data Access API complies with Servlet 3.0 specification and is designed to be deployed to Apache Tomcat or another Servlet 3.0 container.

Besides the Jersey, it utilizes CronUtils[3] library that allows parsing CRON expressions that are used In certain data processing tasks.

6.6 Data security

The data security is a crucial aspect of the automation data processing, as the data can be easily misused during a hostile attack. The data security is discussed in more detail in Section 4.9.

This section describes the security mechanisms that are part of the Semantic BMS project.

The privacy and integrity of the data is out of scope of the Semantic BMS. Those aspects of security are ensured by the HTTPS protocol that is recommended for the communication with the API. The support of the secure communications via the HTTPS protocol is provided by the container in which the Semantic BMS APIs are deployed (e.g. Apache Tomcat).

The authentication is implemented using HTTP *Basic Access Authentication*. Although insecure by itself, it is considered sufficient in conjunction with the HTTPS protocol.

The authentication and authorization mechanism is not a function of the core of the Semantic BMS project. Rather, the authorization is left for implementation in extension modules. The Semantic BMS can be extended by authentication and authorization modules (AA modules) that meet requirements of the specific deployment.

3. Available from: `https://github.com/jmrozanec/cron-utils`

Generally, the Semantic BMS publishes data from other information systems and infrastructures – building automation systems (BAS, BMS) and building information models (BIM). The user privileges can be extracted from the mentioned systems and re-used when authorizing the queries to the Semantic BMS APIs. The Semantic BMS does not store any data. For that reason, it does not aim to store the user privileges. It is expected that the user privileges will be evaluated against an external service or database.[4]

The Semantic BMS passes the requests and responses to the AA module that authenticates the user and decides if the user is authorized to perform the query. If so, the AA module decides if he or she is allowed to retrieve the resulting records. The evaluation of the privileges is performed individually for each record from the result set. This way, the result set is filtered to contain only such records that the user is allowed to view.

In compliance with the usual approach, the authorization process considers two dimensions – user role and resource-based access control.

There are only two user roles defined by the Semantic BMS – Admin and User. The User role has read-only access to the semantic data. The Admin role can modify, add, and remove the semantic data. The distinction of the roles applies only for the Semantic API. The data provided by the Data Access API are always read-only.

The resource-based level security allows to specify which subset of the data can be accessed by an individual user. The Semantic BMS however does not allow to set different user roles for different data (e.g. the Admin for one building and the User for another). The Admin role should be granted only for the system administrators that ensure the operation of the whole Semantic BMS middleware layer. It should not be granted to regular users.

Splitting the API to two distinct parts – Semantic API and Data Access API – requires to authorize (and authenticate) them separately, as both kinds of information (semantic information and observed values) can be misused.

4. However, the authorization modules naturally can use their own dedicated privilege stores if the developers decide to do so.

When authorizing the Semantic API queries, the full range of semantic information could be potentially used. However, such functionality requires embedding the authorization process to the ontology repository querying. As an example, consider a user that possesses rights to view data from one particular building. The user places query to obtain network addresses of temperature sensors. In such situation, neither the query nor the result set contain an attribute that specifies the building in which the temperature sensor is located. Thus, the query must be extended to obtain the missing attribute (location of the sensor), then filter the results according to access privileges, create a projection of the result set (i.e. hiding the additional attribute) and return the result set.

This approach was considered too complex and with a significant performance impact. For that reason, this complex authorization mechanism is not implemented. Instead, the BMS ID (network address in the BAS system) is considered the determining attribute. This attribute is always present either in the query or in the result set. This is caused simply by the purpose of the Semantic API – it serves for finding information about a network address or to find network addresses based on various criteria. For the Data Access API, the query always constitute of BMS addresses.

Furthermore, it allows to share the authorization mechanism between the Semantic API and Data Access API. The data access privileges are very likely defined in the BAS/BMS system that is used to support regular operation of the facility. The privilege definitions should be made available to the Semantic BMS APIs by an authorization extension module and then used during the query authorization. This authorization policy follows a simple rule – if the user is allowed to see the data from a data point, there is no risk in providing him with additional semantic information.

To summarize this section, the Semantic BMS provides interfaces to implement custom authentication and authorization modules. The AA modules have four main responsibilities:

- Authentication – provided by an user name and a password (coming from the HTTP Basic Access Authentication request header), the AA module decides if the user is authenticated.

- Role authorization – Based on the user name, the AA module assigns the user to one of the available user groups.
- Resource-based query authorization – Based on the data point address (BMS ID) specified in the query, the AA module decides if the user is allowed to execute the query.
- Resource-based result authorization – Based on the data point address (BMS ID) specified in the result set, the AA module decides if the user is allowed to retrieve the result.

6.7 Project repository

The project repository for the Semantic BMS is located at `https://gitlab.fi.muni.cz/xkucer16/semanticBMS`, currently containing:

- SBMS Ontology definitions;
- Implementation of the Semantic API and the Ontology repository;
- Proof-of-concept Client for the Semantic API (SBMS Client);
- Implementation of the Data Access API definition and sample module for the BACnet automation protocol. As of August 2017, the Data Access API is not in a stable version and undergoes continuous development. The basic principles and architecture presented in this work are however settled.
- Test Bench – Set of tools that can be used during the Semantic BMS testing and deployment (see 7.6).
- Sample data set created using mock-up data generated by the Test Bench tools.

6.8 Summary

This section of the thesis presented the Semantic BMS project as a method to solve the issues related to analysis of building operation analysis. The Semantic BMS project consist of semantic model and a middleware layer – set of RESTful APIs that allow both querying of the semantic model and the automation network itself.

The Semantic BMS distinguishes between the data point and the actual measurement. The data point represent an address in the building automation systems that publishes measurements. The data points are annotated in the ontology model to enrich them with additional semantic information. The data points then can be selected using the Semantic API. Actual data (measurements) coming from the data points can then be retrieved using the Data Access API.

The ontology model (Semantic BMS Ontology) was described using the OWL language that is suitable for formal description of the model and allows to use logic reasoning over the asserted axioms.

The Semantic BMS Ontology is derived from the W3C Semantic Sensor Network Ontology and extends it to the use in the domain building automation. Employment of the SSN introduces the concepts of Observations and Measurements and Stimulus-Sensor-Observation design pattern into the domain of building automation, where it is a novel view on the measured data.

The Semantic BMS focuses not only on the semantic description of the data points, but also on querying of the ontology model. Thus, the middleware layer provides access to the semantic model to simplify integration. This is the main contribution of the research. The Semantic BMS provides convenient interfaces that allows to gather semantic data about data point or easily select data points that satisfy required criteria without the need to fully understand specifics of the building automation systems or specifics of the ontology languages.

The Semantic BMS pursues the same goal when dealing with the automation data. The Data Access API provides easy-to-use endpoints that can be facilitated by application developers in an BMS-protocol-independent manner and using widely known and supported technologies.

7 Evaluation: End-user applications and use cases

This chapter provides overview of different evaluation methods applied to the Semantic BMS project. The evaluation focuses on different aspects of the project functionality and capabilities:

- Suitability for application developers – evaluated by the proof-of-concept client.
- Suitability for system integrators – evaluate by the use case implementations of the data providers for the Data Access API.
- Suitability for facility managers – evaluated by the proposed use cases and by comparing the Semantic BMS to other available approaches.
- Suitability of performance – evaluated by performance benchmarking.

The first consideration is the suitability for developers. The goal is to answer the question if the Semantic BMS APIs are suitable for development of end-user applications. This was tested by a demonstration application *Semantic BMS client* that illustrates capabilities of the APIs. The application is available at the project repository (see the Section 6.7).

The second evaluation direction also focuses on software developers. In this case, ability to extend the Semantic BMS by automation data providers is examined. This question was assessed by development of data providers for the use case of Masaryk University (See the Section 2.1). The data providers for BACnet protocol and Delta Control Historian archive server were implemented and successfully tested.

The third aspect of the Semantic BMS usability is its suitability for actual facility management benchmarking task. For this purpose, two use cases were defined. The use cases were inspired by experience with the operation of the University Campus and by the European Standard *EN 15221 Facility Management, Part 7: Guidelines for Performance Benchmarking*. Detailed description of the use cases provides overview of benefits that employment of the Semantic BMS brings to the benchmarking process and proves suitability of the APIs for the planned use cases.

The last consideration regarding the Semantic BMS usability is the performance, especially performance of the Semantic API querying, since performance is often an issue when employing the semantic web technologies.[1]

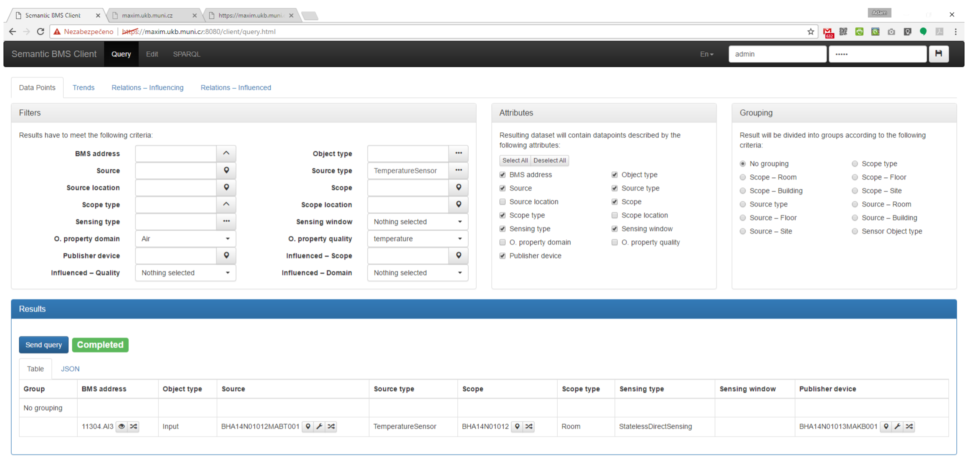
All the above mentioned aspects of the Semantic BMS evaluation are covered in the following sections.

7.1 Semantic BMS client

The SBMS Client was developed as a lightweight tool for testing and demonstration purposes. The client is Javascript application running in web browser, developed using HTML 5 and jQuery framework. The client has following capabilities:

- Query the ontology repository using the Semantic API (See Figure 7.1). Results returned in a custom JSON format or in a tabular view.
- Query the ontology repository using SPARQL.
- Insert data into the ontology repository using web user interface.
- Insert data into the ontology repository using batch import from a tabular format.
- Remove data from the ontology repository.
- Show system integration capabilities[2]:
  - BIM – Show locations of elements that are represented both in SBMS and BIM on a map (See Figure 7.2).

1. The Data Access API performance is not tested, since it largely depends on the building automation system itself.
2. These features are specific to the environment of Masaryk University since they must be tailored to meet requirements of specific information systems used by the organization


Filters

Results have to meet the following criteria:

BMS address		Object type	
Source		Source type	TemperatureSensor
Source location		Scope	
Scope type		Scope location	
Sensing type		Sensing window	Nothing selected
O. property domain	Air	O. property quality	temperature
Publisher device		Influenced – Scope	
Influenced – Quality	Nothing selected	Influenced – Domain	Nothing selected

Attributes

Resulting dataset will contain datapoints described by the following attributes:

- ✓ BMS address
- ✓ Object type
- ✓ Source
- ✓ Source type
- ☐ Source location
- ✓ Scope
- ✓ Scope type
- ☐ Scope location
- ✓ Sensing type
- ✓ Sensing window
- ☐ O. property domain
- ☐ O. property quality
- ✓ Publisher device

Grouping

Result will be divided into groups according to the following criteria:

[✓ No grouping]

Results

Group	BMS address	Object type	Source	Source type	Scope	Scope type	Sensing type	Sensing window	Publisher device
No grouping									
	11304.AI3	Input	BHA14N01012MABT001	TemperatureSensor	BHA14N01012	Room	StatelessDirectSensing		BHA14N01013MAKB001

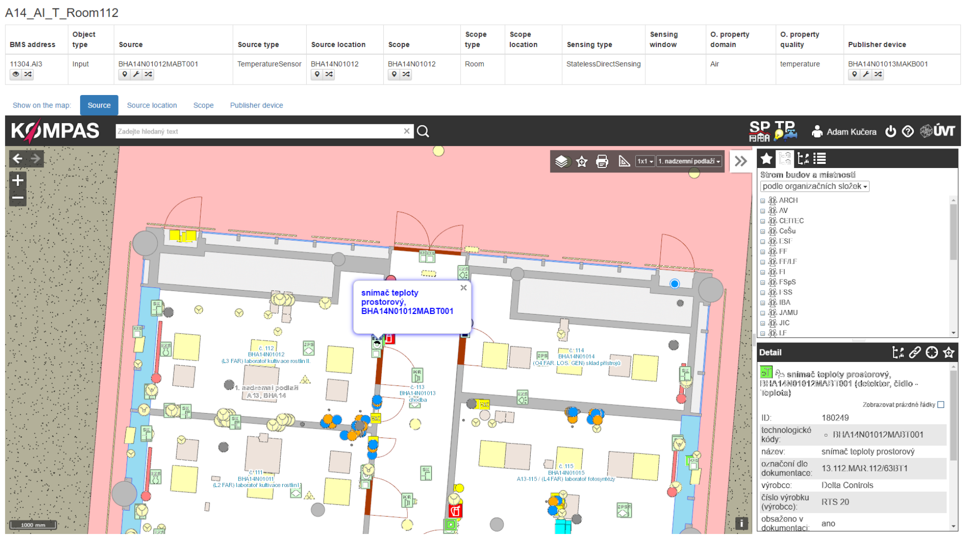
Figure 7.1: Semantic API Client – Querying the ontology repository.

- – CAFM – Show maintenance history of given device (usually available for Source device).
- – BMS – Show present value or historic data for given data point.

7.2 Implementation of the data providers

The Data Access API provides the interface to the front-end application. The interfaces with the BAS is not integral part of the Semantic BMS in order to allow extensions of the system. The modular system allows to employ user's own connectors to the BAS (called *Data providers* in the Semantic BMS). The Semantic BMS provides interfaces and utility classes that largely simplify development of custom data providers.

To assess an usability of proposed approach, sample data providers were implemented for the use case of Masaryk university. Three data providers were implemented.



A14_AI_T_Room112

BMS address	Object type	Source	Source type	Source location	Scope	Scope type	Scope location	Sensing type	Sensing window	O. property domain	O. property quality	Publisher device
11304.AI3	Input	BHA14N01012MABT001	TemperatureSensor	BHA14N01012	BHA14N01012	Room		StatelessDirectSensing		Air	temperature	BHA14N01013MAKB001

Show on the map: Source Source location Scope Publisher device

snímač teploty prostorový, BHA14N01012MABT001 (detektor, čidlo - Teplota)

Zobrazovat prázdné řádky

ID:	180249
technologické kódy:	BHA14N01012MABT001
název:	snímač teploty prostorový
označní dle dokumentace:	13.112.MAR.112/63BT1
výrobce:	Delta Controls
číslo výrobku (výrobce):	RTS 20
obsaženo v dokumentaci:	ano

Figure 7.2: Semantic API Client – Integration with BIM system.

BACnet data provider is used for gathering on-line data (present values) from the BAS that uses BACnet as a communication protocol. It uses multi-threaded access to the network, shrinking the data retrieval to minimum when obtaining data from multiple devices. The data provider relies completely on utility abstract classes provided by the Data Access API core. The utility classes handle data aggregation and other tasks required by the Data Access API. The BACnet data connector implements only the essential data acquisition methods.

BACnet trend data provider serves to access historical data that are stored in the *Trend* objects in the automation devices in the network. Since the BACnet protocol does not provide any data processing methods (e.g. aggregation computations), the data provider leaves the data aggregation tasks to the Data Access API core. Both BACnet-related provides share the same Singleton class that represent the virtual BACnet device that access the BAS network.

Delta Controls Historian data provider uses JDBC connection to an SQL database containing data archived by the Delta Controls Historian application. There is a wide variety of SQL-based archiving applications for automation protocols, however the database schemes are proprietary. The SQL language provides rich data processing capabilities. For performance reasons, it is desirable to transfer as much data aggregation tasks to the database server. The provided implementation illustrates this approach in conjunction with the Data Access API core.

7.3 Use case 1: Room environment

For the first use case, evaluation of room environment (humidity, temperature) is the goal of benchmarking process. The reason for such benchmark might be internal (to ensure satisfaction of facility users) or they can be enforced by law. For example, In Czech Republic, there is a lower limit of room temperature for office spaces (20°C) given by a government ordinance. To achieve the benchmarking goal, multiple methods can be used, compared in Table 7.1.

Table 7.1: Use case 1: Method comparison

Approach	Survey	Loggers	BAS	SBMS
Sample size	Variable	Small	Large	Large
Precision	Low	High	High	High
Investment	None	Low	High	Very high
Delay	Weeks	Weeks	Days	Hours
Difficulty				
– Setup	Very high	High	Moderate	Low
– Repetition	High	Moderate	None	None
– Data collection	Very high	High	Moderate	None
– Evaluation	High	Low	Low	Low
– Refocusing	High	High	Moderate	None

The least precise and accurate method is a poll of an employee (user, customer) satisfaction by opinion survey. When performing a survey, no financial investment is required. However, preparation of the questionnaire, collecting of the results and processing of data are time-consuming tasks. The resulting data are of qualitative nature. The data are made quantitative in later processing steps (placing ratings on a discrete scale, computing mean and other statistical methods). Those are properties discourage from frequent repetition of the benchmark using method of survey. Naturally, results of opinion survey cannot be used as evidence of compliance with standards or law.

To collect more accurate results, facility manager might decide to buy certain number standalone loggers and place them in "sample rooms", constantly monitoring and logging room temperatures for desired period. In the end of the measurement period, loggers are collected, data are downloaded to a computer and processed. Alternatively, the data can be obtained by wireless radio technology when using certain types of loggers. Main disadvantages of this approach are sample size (number of loggers is significantly lower than number of rooms), complexity of a benchmarking setup (necessity to select representative rooms – e.g. different orientation, area, number of windows) and delay between setup of the benchmark (placing meters) and available results because of the time that continuous measurement of temperatures deserves.

Installation of a BMS in a facility brings a huge leap in a simplicity of this benchmarking process. Sensors are usually placed in all rooms of a facility and temperatures are measured and logged constantly, thus removing the gap between benchmark setup (placing the loggers) and data retrieval (downloading measured data) – data are instantly loaded from the archival database. However, as stated before, typical BMS systems do not provide a method to conveniently select appropriate data points and usually require data transformation. When using BMS data, repetition of the whole benchmark is as simple as running query prepared for the previous run. However, refocusing the benchmark (i.e. evaluate temperatures in a different set of rooms) requires performing a time-consuming task of determining appropriate data points again. That said, a delay from an intent to a benchmark result is usually several hours or even a few days.

Engaging the Semantic BMS into a benchmarking process resolves the above-mentioned problems related to the use of BMS data – it facilitates identification of data points and simplifies data retrieval tasks.

7.3.1 API Query examples

To further demonstrate the simplicity of use, consider two situations:

1. Quick assessment of average room temperatures during last month for all buildings of a particular site (identified in BIM as "BHA").

2. Thorough analysis of air temperatures and respective set-point values in all rooms of one particular building (identified in BIM as "BHA14").

For the first situation, historical data about room temperatures will be used, so the user is interested only in data point addresses (BMS IDs). The user will setup the query to the `trends` endpoint with following parameters:

- Object type: `Input`
- Source type: `Temperature Sensor`
- Source location: `BHA`
- Scope type: `Room`
- Property domain: `Air`
- Property quality: `Temperature`
- Sensing type: `Stateless Direct Sensing`
- Grouping: `Scope-Building`

That will convert to the following REST query:

```
GET https://.../sbms/semantics/trends/
?fields=bmsId,dataPoint.bmsId
&grouping=scope.building
&dataPoint.type=Input
&dataPoint.source.type=TemperatureSensor
&dataPoint.source.location=BHA
&dataPoint.scope.type=Room
&dataPoint.property.domain=Air
&dataPoint.property.quality=temperature
&dataPoint.sensing.type=StatelessDirectSensing
```

The response will then return all the data points that meet given criteria, grouped according to the building they are located in (the following example is limited to 5 sensors in each building due to space considerations):

```
{
  "results": {
    "BHA15": [
      { "bmsId": "93.TL61", "dataPoint": { "bmsId": "116.AI11" }},
      { "bmsId": "93.TL88", "dataPoint": { "bmsId": "118.AI10" }},
      { "bmsId": "93.TL95", "dataPoint": { "bmsId": "118.AI1" }},
      { "bmsId": "93.TL96", "dataPoint": { "bmsId": "118.AI7" }},
      { "bmsId": "93.TL20", "dataPoint": { "bmsId": "118.AI2" }}
    ],
    "BHA14": [
      { "bmsId": "93.TL24", "dataPoint": { "bmsId": "111.AI11" }},
      { "bmsId": "93.TL45", "dataPoint": { "bmsId": "113.AI1" }},
      { "bmsId": "93.TL46", "dataPoint": { "bmsId": "113.AI2" }},
      { "bmsId": "93.TL47", "dataPoint": { "bmsId": "113.AI3" }},
      { "bmsId": "93.TL48", "dataPoint": { "bmsId": "113.AI4" }}
    ],
    "BHA23": [
      { "bmsId": "93.TL04", "dataPoint": { "bmsId": "147.AI11" }},
      { "bmsId": "93.TL30", "dataPoint": { "bmsId": "149.AI1" }},
      { "bmsId": "93.TL32", "dataPoint": { "bmsId": "146.AI1" }},
      { "bmsId": "93.TL34", "dataPoint": { "bmsId": "148.AI1" }},
      { "bmsId": "93.TL36", "dataPoint": { "bmsId": "149.AI2" }}
    ]
  },
  "groups": [
    "BHA14",
    "BHA15",
    "BHA23"
  ]
}
```

The returned BMS IDs can then be used for retrieving the historical data and computing the aggregation using appropriate tools, such as SQL. The Data access API, which is planned as part of the Semantic BMS middleware, but is not implemented yet, will have the capability to consume the JSON response of the Semantic API together with supplemental parameters (aggregation method, a time window) and provide the results directly in the second step.

In the second situation ("Thorough analysis of air temperatures and relative air humidity in all rooms of one particular building"), query parameters will be different:

- Object type: `Input`
- Source type: `Sensor`
- Source location: `BHA14`
- Scope type: `Room`
- Property domain: `Air`
- Sensing type: `Stateless Direct Sensing`
- Grouping: `Scope: by Room`

In this situation, the query takes advantage of the ability of ontology languages and tools to provide inference resolving and inheritance of classes. In the query, user is able to specify Source type to `Sensor`, which is parent class for both `TemperatureSensor` and `HumiditySensor`. The API will, however, return all sensors that are placed in rooms, thus under certain circumstances, other types of sensors might occur in the result set (most probably air pressure sensors, since the results are limited by `Air` domain). To distinguish different types of sensors, `Source Type` was added to the list of retrieved attributes.

Query string will change accordingly, resulting in following response (the following example is limited to 5 sensors in each building due to space considerations):

```
{
  "results": {
    "BHA14N03031": [
      { "bmsId": "11219.AV8",
        "source": { "type": "TemperatureSensor" }},
      { "bmsId": "11219.AV1",
        "source": { "type": "HumiditySensor" }}
    ],
    "BHA14N03032": [
      { "bmsId": "11220.AV8",
        "source": { "type": "TemperatureSensor"}},
      { "bmsId": "11220.AV1",
        "source": { "type": "HumiditySensor"}}
    ],
    "BHA14N02014": [
      { "bmsId": "11212.AV8",
        "source": { "type": "TemperatureSensor"}},
      { "bmsId": "11212.AV1",
        "source": { "type": "HumiditySensor"}}
    ],
    "BHA14N03025": [
      { "bmsId": "11221.AV8",
        "source": { "type": "TemperatureSensor"}},
      { "bmsId": "11221.AV1",
        "source": { "type": "HumiditySensor"}}
    ],
    "BHA14P01015": [
      { "bmsId": "11203.AV8",
        "source": { "type": "TemperatureSensor"}},
      { "bmsId": "11203.AV1",
        "source": { "type": "HumiditySensor"}}
    ]
  },
  "groups": [
    "BHA14N03031",
    "BHA14N03032",
    "BHA14N02014",
    "BHA14N03025",
    "BHA14P01015"
  ]
}
```

7.4 Use case 2: Energy consumption

The second use case aims to illustrate other aspects of benchmarking where BMS data and Semantic BMS can be used to improve the benchmarking process, namely drill-down capabilities and benchmarking period.

Table 7.2: Use case 2: Method comparison

Approach	Metering	Invoices	Parsing	BAS	SBMS
Investment	None	None	Moderate	High	Very high
Delay	High	Moderate	None	Low	None
Period	Arbitrary	Fixed	Fixed	Arbitrary	Arbitrary
Drilldown ability	Unlimited	None	None	Unlimited	Unlimited
Difficulty					
– Setup	Moderate	Low	Low	Moderate	Low
– Repetition	Low	Low	None	None	None
– Data collection	Very high	High	None	Moderate	None
– Refocusing	Moderate	Low	None	Low	None

The benchmark focuses on energy consumption, specifically electricity. Benchmarking methods for this use case are compared in Table 7.2. The two basic approaches employ human labour to collect data. Three other approaches use a certain level of automation at the expense of initial cost.

They, however, differ in the data source. Metering based approach uses values collected from energy meter devices located in the facility. The data are copied from the displays of meters by human workers and then (preferably) digitalized. Inexpensive energy metering devices can be purchased to monitor even individual devices (plug-in energy monitors). For that reason, the metering based method allows for a good ability of drill-down and scalability concerning benchmarking period with low to none initial investments. However, data collection is time demanding task and there are no historical data ready in the moment of the benchmarking process setup, contrary to the other methods.

When applying the invoice based approach, data are collected by a human operator from the invoices from energy vendor. This is inexpensive method both concerning investments and time demands. On the other hand, it does not allow for drill-down (invoices are usually provided as one per building or one per site) and the benchmarking period is fixed to the accounting period of the energy invoices.

In the field of CAFM, automated parsing of invoices is considered as a viable option for energy monitoring. In some cases, invoices can be provided in a convenient machine-readable format such as XML. This approach requires specialised software tools for parsing invoice data, rendering this method more demanding regarding initial investment and less time consuming, compared to the previous approach.

Using the BMS data for energy consumption benchmarking combines advantages of parsing based method with the metering-based method – abilities of drill-down, flexible period and automated data collection. However, as in the previous use case, selecting appropriate data points might prove as a time-consuming task, depending on a size of a BMS installation and a facility. Similarly, data collection might comprise a development of a complex application for data gathering.

The SBMS further addresses the issue of data selection by facilitating semantic annotation of the data points. Instead of manual discovery of the required data points in the BMS user interface or archive database, the data points can be selected using attributes such stored in the SBMS ontology (e.g. Scope, Property domain, Physical quantity). During the data collection, the SBMS APIs can be utilised by developers, rendering the development of reporting application significantly easier and more straightforward.

7.5 Use case 3: Automated report of energy consumption

The third use case extends the Use Case 2 to illustrate the feasibility of the Data Access API for the BAS data retrieval.

The goal of this use case is to create a report that summarizes energy consumption (electricity, water, and heat) from multiple buildings located at one site. This use case is inspires in a reporting scenario that is used at the University Campus of Masaryk University.

7.5.1 Current solution

Currently, the report is available via an Excel spreadsheet that is stored on a shared drive. The spreadsheet is updated monthly by an PowerShell script that accesses the archive database, download the required data and updates the spreadsheet. At first glance, this solution might appear rather cumbersome to the reader. However, there are valuable advantages to this solution:

- Simplicity of the software development – Using the PowerShell environment, .NET libraries, SMB/CIFS file sharing and Task Scheduler in MS Windows OS, the required programming was limited to a mere minimum, facilitating existing tools and technologies. Programming of this report was a question of several hours in a situation where no advanced framework was available.
- No access to the database for the end users – The users do not access the data store directly (as opposed to the solution using client-side technology such as VBA). The database schema used by the archive server (Historian) does not allow to set user privileges to the individual data point archives – user either sees all the data, or none.
- Automatic updates of the spreadsheet – the data are readily available to the users when they open the spreadsheet. There is no delay caused by a data download and there is no need for the user to initiate the update.

That said, there are several important drawbacks of this solution:

- Difficulty of the address collection - The IDs (addresses) had to be gathered manually by system operators by browsing the BMS user interface and identifying the data points of interest. This is a time consuming process with limited re-usability. The data were gathered for the purpose of the report creation and cannot be used for another purposes, as there is no central storage of such data and the only source that is aware of the existence of such data are the workers that took part in the report creation.
- Delay – The consumption data are retrieved from the archive database, since there was no method to directly access the BAS network. However, due to the nature of the archiving process in the BACnet networks, the data in the archive emerge with certain delay that may be several days in extreme cases. For that reason, consumption for the last month is added to the report with a delay of three days.

- Coarse user privileges – In the current setup, there is only one level of user privilege. User either has access to the file on the shared drive /and thus is able to see all the data), or does not have access and sees none. A fine grained user privileges that would allow different users to view different subsets of the data (e.g. according to the energy type or location) requires creating different reports (spreadsheets) with a different access rights on the file system (SMB/CIFS) level.

- User interactions with the file – All the users of the report use the same shared file. If one user modifies it, the report changes for all the others. Even worse, if someone damages the file (e.g. by deleting or altering the data), this error will affect all the users.

- Inflexibility – Users can view only the predefined report – They can't view older data and the consumptions are saved to the file once a month. More frequent reading can't be obtained using the existing report. All changes must be performed by the developer in the generating script.

7.5.2 Benefits of the Semantic BMS employment

The Semantic BMS solves multiple issues of this use case. First, it removes the repeated and expensive data gathering – with the Semantic BMS employed, the semantic data about the data points can be collected just once and reused. Even though the Semantic BMS does not remove the need to gather the semantic information, it highly improves its usability. This benefit is however described in detail in sections devoted to the use cases 1 and 2.

This section aims on benefits that brings the Data Access API to the tasks of a data retrieval.

There are several levels on which the Data Access API can be deployed into the scenario.

The Data Access API is able to directly access the BAS network, thus solving the problem of delay. It maintains the benefit of not allowing users to access directly to the data store, as they communicate only with the Semantic BMS API. Furthermore, the API provides fine-grained authorization that solves the problem of the coarse user privileges.

The Data Access API allows to move the data acquisition tasks from the server to the client computer, as it is no more required to grant access to the archival database to the end users – the Semantic BMS provides fine grained authorization mechanism. Moving the data gathering to the clients also solves the problem of the shared file, as each user can have its own copy. This solution sacrifices the automatic updates feature, but it offers much greater flexibility in exchange.

There are several approaches to implementation client-side driven data retrieval. The main requirement is to have the data available in a spreadsheet processor, namely Microsoft Excel for the purposes of the facility management staff at the Masaryk University. The best option seems to be development of a simple Excel add-in using the Visual Studio Tools for Office (VSTO) technology. The add-in able to add new features into the program and modify the documents. A main disadvantage is that users have to install the add-in to be able to work with it, however the simplicity of use the add-in can provide compensates for this complication, even more for a use within an organization without the need to distribute the add-in among general public.

The add-in is not developed yet, however the concept of its design is rather clear and straightforward. The concept is not an original one – this approach is commonly used in business intelligence and facility management domain. The particular inspiration comes from the OSIsoft PI Datalink tool that was seen by the author at an OSIsoft product presentation.

The add-in that would support this use case (among many others) should allow user to query both the Semantic API and Data Access API to retrieve the addresses of interest and the building operation data.

7.5.3 Request parameters

When the end users are in charge of data retrieval, it is possible to let them specify additional parameters that are currently fixed because the data gathering is marshaled by the server-side script. The most important parameters in this use case are time specs – period of interest (e. g. that last year) and the sampling interval (e. g. first day of each month). The other parameters will remain constant for the spreadsheet.

Assume that the addresses where gathered using the Semantic API as shown in the use case 2. To retrieve the meter data, a call to the Data Access API must be performed.

There are several parameters that must be specified in the request:

- Addresses of interest – The JSON payload received from the Semantic API can be used without modifications as an input for the Data Access API.

- Structure and data format of the response – This is specified by the URL of the specific API RESTful method that is used, as each of the methods conforms to one of the output structures (e.g. Scalar, Slice, Series, Set of Series as defined in the Section 6.5.4.

- Aggregation and/or Sampling type – These parameters define how the API should process the raw data. For this specific scenario, there are several ways to achieve the expected result. The different approaches are described below.

- Temporal parameters – If querying for the historical data, temporal parameters must be used. The basic parameters are the start time stamp and end time stamp. If sampling or aggregation windows are used, a pattern for dividing the overall period must be provided (either by duration or by CRON expression).

There are multiple methods to obtain the consumption of energy for given time period. A main consideration is the type of the data that are available. For energy consumption, a typical setup is a data point that represent total consumption (i.e. the same way traditional energy meter works), however it might be the case that the BAS system records only increments for a defined period (e.g. day or month). The goal of the report is to present consumption for a defined period. That renders following options:

1. The BAS records increments and the period is the same as the required by the report. No data processing is needed, the raw data can be used.

2. The BAS records increments and the period is longer than the period required by the report. This is the worst case scenario, but it is also very improbable. However, should such situation occur, raw data would be obtained and processed by the client to obtain reasonable estimate.

3. The BAS records increments and the period is shorter than the period required by the report. The Sum aggregate must be computed from the raw data for each of the desired periods. Series of Windows structure should be used.

4. The BAS records a total and the period is the same as required. The simplest option is to retrieve the raw data and compute intermediate consumptions using capabilities of a spreadsheet processor such as Microsoft Excel.

5. The BAS records a total and the period is longer than the period required by the report. This also a case where the raw data are not suitable for the required report. In this particular situation, sampling would be used together with the linear interpolation to obtain estimates of the consumption in the required intervals.

6. The BAS records a total and the period is (significantly) shorter than the period required by the report. This is the most probable case. In this situation, the best option is to use the Series of Windows structure together with the Difference aggregation function that subtracts value of the first record from the value of the last record. Alternatively, the data can be sampled in the desired intervals and the consumption can be computed from the retrieved totals in the spreadsheet processor (this is the same approach as in the fourth case in this list).

The last mentioned configuration being the most probable, call to the Data Access API is illustrated using this scenario.

7.5.4 Example of the Data Acess API call

The request to the Data Access API thus specifies the following parameters:

- Addresses of interest – result of the previous Semantic API call (see Use Case 2)
- Output format – Series of Windows;
- Aggregation – Difference;
- Sampling – None;
- Start of the period (e.g. 2017-01-01);
- End of the period (e.g. 2017-07-01);
- Window definition – for a window of a calendar month, CRON expression `0 0 0 1 * ?` should be used;
- Grouping – Grouping can be used to divide the energy meters to groups based on the type of energy they measure (i.e. electricity, gas, water – using the source device type described in Section 6.3.2). The data then can be easily divided into separate sheets in the Excel workbook.

This parameters will result in the following Data Access API call:

```
POST https://.../sbms/data/trends/seriesOfWindows
?aggregation=difference
&start=2017-01-01
&end=2017-07-01
&window=0 0 0 1 * ?

Request body:
{
  "results": {
    "EnergyMeter": [
      { "bmsId": "93.TL91",
       "dataPoint": { "bmsId": "11600.AI11" }
        "scope": {
          "bimId": "BHA14",
          "type": "Building"
        },
        "property": {
          "domain": "Electricity",
          "quality": "Energy"
    }
  },
  { "bmsId": "93.TL92",
   "dataPoint": { "bmsId": "11700.AI11" }
    "scope": {
      "bimId": "BHA15",
      "type": "Building"
    },
    "property": {
      "domain": "Electricity",
      "quality": "Energy"
    }
  }
],
"GasMeter": [
   { "bmsId": "93.TL93",
   "dataPoint": { "bmsId": "11600.AI12" }
    "scope": {
      "bimId": "BHA14",
      "type": "Building"
    },
    "property": {
      "domain": "Gas",
      "quality": "Energy"
    }
  },
  { "bmsId": "93.TL94",
   "dataPoint": { "bmsId": "11700.AI12" }
    "scope": {
      "bimId": "BHA15",
      "type": "Building"
    },
    "property": {
      "domain": "Gas",
      "quality": "Energy"
    }
  }
],
"WaterMeter": [
  { "bmsId": "93.TL95",
       "dataPoint": { "bmsId": "11600.AI13" }
        "scope": {
          "bimId": "BHA14",
          "type": "Building"
        },
        "property": {
          "domain": "Water",
          "quality": "Volume"
        }
      },
      { "bmsId": "93.TL96",
       "dataPoint": { "bmsId": "11700.AI13" }
        "scope": {
          "bimId": "BHA15",
          "type": "Volume"
        },
        "property": {
          "domain": "Water",
          "quality": "Volume"
        }
      }
    ]
  },
  "groups": [
    "Electricity",
    "Gas",
    "Water"
  ]
}
```

Resulting in a response similar to the following:

```
{
  "results": {
    "EnergyMeter": {
      "93.TL91" : {
       "2017-02-01" : 124.5,
        "2017-03-01" : 110.2,
        "2017-04-01" : 112.8,
        "2017-05-01" : 144.2,
        "2017-06-01" : 102.2,
        "2017-07-01" : 111.8
      },
      "93.TL92" : {
       "2017-02-01" : 122.5,
         "2017-03-01" : 111.8,
         "2017-04-01" : 109.2,
         "2017-05-01" : 127.1,
         "2017-06-01" : 100.0,
         "2017-07-01" : 101.5
      }
    },
    "GasMeter":{
      "93.TL93" : {
       "2017-02-01" : 8.1,
         "2017-03-01" : 9.2,
         "2017-04-01" : 4.4,
         "2017-05-01" : 8.6,
         "2017-06-01" : 5.1,
         "2017-07-01" : 111.8
      },
      "93.TL94" : {
       "2017-02-01" : 0.5,
         "2017-03-01" : 1.8,
         "2017-04-01" : 0.2,
         "2017-05-01" : 2.1,
         "2017-06-01" : 0.0,
         "2017-07-01" : 1.5
      }
    },
    "WaterMeter": {
      "93.TL95" : {
       "2017-02-01" : 30.4,
         "2017-03-01" : 28.1,
         "2017-04-01" : 27.4,
         "2017-05-01" : 26.9,
         "2017-06-01" : 32.4,
         "2017-07-01" : 30.5
      },
      "93.TL96" : {
       "2017-02-01" : 25.4,
         "2017-03-01" : 26.8,
          "2017-04-01" : 29.4,
          "2017-05-01" : 22.3,
          "2017-06-01" : 25.8,
          "2017-07-01" : 28.5
        }
      },
    },
    "groups": [
      "Electricity",
      "Gas",
      "Water"
    ]
}
```

The JSON object can then be parsed by the add-in a the data can be used to construct the spreadsheet. One of the possible representations of the data is shown in the Table 7.3. The table presents the electricity consumption only – the gas and water consumption tables should be placed in separate sheet.

Table 7.3: Visualization of the Data Access API call: Electricity consumption

From	To	Consumption for BHA14	Consumption for BHA15
2017-01-01	2017-02-01	124.5	122.5
2017-02-01	2017-03-01	110.2	111.8
2017-03-01	2017-04-01	112.8	109.2
2017-04-01	2017-05-01	144.2	127.1
2017-05-01	2017-06-01	102.2	100.0
2017-06-01	2017-07-01	111.8	101.5

The presented use case illustrates capabilities of the Data Access API, its intended use and benefits it will bring to the benchmarking in facility management. Although this use case is not fully implemented at the moment, we believe that it suitably present its feasibility and usefulness.

7.6 Query performance

This section discusses performance of the Semantic API. The performance of the Data Access API is dependent on number of parameters (e.g. size of the BAS network, performance of the background database warehouse, performance of the protocol stack used, design decisions during a data provider development) that are protocol-specific or even deployment-specific, thus a performance benchmarking was not carried out. The rest of this section describes the performance of the Semantic API.

The query performance assessment was performed on a sample data set. The benchmarking procedure details, tools used, detailed results and instructions for replication are available at the project repository in the `TestBench` directory. In the following section, test aims, methods and results will be discussed.

The main aim of the benchmarking is to prove that the query performance is sufficient for intended purposes, as the performance was not a primary design goal. The secondary aim was to identify possible bottlenecks of the semantic model.

For the performance tests, artificial mock-up facility of an organization was used. The data set consist of several sites and buildings. Each building is equipped with a PLC, an energy meter and two data points – overall and last month energy consumption. Each room is equipped with a temperature sensor, a humidity sensor, a motion sensor and a PLC that publishes following data points: 1) Air temperature, 2) Air humidity, 3) Set-point temperature, 4) Occupancy.

For each of the data points, there are two trends (containers for archive data) defined - one in the PLC itself, the other in the archive historian database.

The benchmarking procedure consists of several Semantic API calls. Therefore, the query tests the performance of the whole Semantic API and all its components. Presented query execution times represent complete round-trip of the query, including processing on the client. However, the SPARQL querying execution time is the dominant component of the overall execution time. The execution time of the other phases is insignificant compared to the duration of the SPARQL query execution (see the end of this section for detailed measurements).

The use cases defined in the Section 7.3.1 were executed (labelled as UC1 and UC2 in the following text). General test cases were investigated as well. The defined test cases query the API for:

1. All available information about a specific data point;

2. Small number of data points based on strict criteria – All temperature sensors that measure an air temperature in a room in a certain building;
3. Large number of data points based on loose criteria – All humidity sensors in the database;
4. Large number of data points based on loose criteria with the "bottleneck" attribute – all humidity sensors in the database and their data point type)
5. Selecting a large number of data points based on loose criteria with all the available information retrieved – All temperature sensors in the database + all attributes.

The test cases 1-3, UC1 and UC2 represent the typical intended use of the framework. The scenarios 4-5 are used to explore limits of the API capabilities and they are not expected to be executed often in the production.

The queries were performed on two data sets. Dataset A comprises of 5 sites, each with 5 buildings of 3 floors, containing 750 rooms in total, resulting in over 3000 data points. The total number of explicitly asserted predicates in the triple store is over 82000. Dataset B comprises of 10 sites, 10 buildings each, 5 floors per building, totaling 10 000 rooms, 40 000 data points and over 1 000 000 asserted triples. To put the numbers into the context, at Masaryk University, about 5000 data points is currently archived in the archive database.

The results are summarized in the Table 7.4. The *DPs* column indicates a number of data points retrieved in the response. The *Attr.* column indicates a number of attributes retrieved in the query response for each of the data points. The *Param.* column indicates a number of restricting parameters used in the query request. *Dur.* stands for Duration. Times in the table are averages of repeated measurements. Comments on the performance benchmark results follow.

The benchmarking was performed on two hardware configurations: A) a 6 years old desktop computer with a quad-core AMD processor, 4GB of RAM and an HDD and B) a new laptop equipped with a dual-core Intel Core i5-6200U processor, 8GB RAM and an SSD. The latter performed significantly better (the execution times were

Table 7.4: Results of the performance benchmarking.

Test case	Data set A		Data set B		Attr.	Param.
	Dur. [s]	DPs	Dur. [s]	DPs		
UC1	1.8	300	17.0	2000	3	7
UC2	3.5	60	22.4	200	3	6
1	0.5	1	4.8	1	13	1
2	0.9	30	14.3	100	2	6
3	1.2	750	16.0	10000	2	2
4	19.1	750	247.2	10000	3	3
5	115.0	750	1612.4	10000	14	3

cut in half) especially in test cases 4 and 5, probably due to higher CPU performance. This promises even more compelling performance in case the Semantic API is deployed on proper server hardware. In the results table, only the times for the configuration B are presented due to the space limitations. The memory consumption reached the maximum of 1.32 GB during querying to the data set B.

The use case 4 identifies one of the bottlenecks of the current model design. The query requires retrieving a data point type. Data point types are modeled as classes. To get desired information, a complex and expensive clause has to be composed. The use case 5 increases the complexity of the query, resulting in even longer execution times that render the scenario almost unusable in production. However, a count of returned data points is extreme, especially for the data set B. We expect this type of query not to be frequently used. Thus, current query composition strategy favors fast execution of the "typical" queries over the most expensive query (5).[3] However, there is a potential for further optimization. There are two directions of optimization efforts – 1) optimization of query composition and 2) adjustments to the ontology model.

The SPARQL query execution time plays utterly dominant role in the overall query execution time. Even for the test case 5 on the data set A (that takes almost 2 minutes to execute), all the other parts

3. e.g. one of the rejected strategies yields 40% performance increase for the query no. 5, but makes the execution of the UC2 15 times slower.

of the request execution and retrieval (network communication, request parsing, query composition, results parsing, processing and formatting) does not require more than 350 ms. For the test case 1, the non-SPARQL-execution time is under 150 ms.[4]

Generally, the performance may be considered satisfactory for use in web applications, especially when used for the reporting. In such use cases, the applications are not fully interactive and users are willing to tolerate certain delay needed for a report generation.

7.7 Summary

This chapter provided overview of evaluation efforts that were applied to the Semantic BMS project. The evaluation aimed to four main aspects:

- Suitability for application developers – evaluated by the proof-of-concept client.
- Suitability for system integrators – evaluate by the use case implementations of the data providers for the Data Access API.
- Suitability for facility managers – evaluated by the proposed use cases and by comparing the Semantic BMS to other available approaches.
- Suitability of performance – evaluated by performance benchmarking.

The evaluation showed that the Semantic BMS fulfills the basic requirements that are needed for sensible application in the production environment. However, the Semantic BMS certainly is not the one and only facility benchmarking solution suitable for everyone. Furthermore, there are additional aspects related to the topic that were not covered yet in this thesis. In the following chapter, usability of the framework is discussed in broader context.

4. The times are recorded with the debug logging enabled. Disabling the logging would further improve performance, as it disables expensive output writing operations, but the logging is necessary to measure the execution times.

8 Discussion: Usability of the SBMS framework

The Semantic BMS provides novel approach to automation data semantics. The project aims to facilitate building automation data for facility benchmarking. This is very valuable source of information, however it is certainly not suitable for every use case.

This chapter aims to answer the following questions (the question numbers correspond with the subsection numbering):

1. What are the benefits of the Semantic BMS compared to the (more) traditional methods?
2. Which organizations are able to benefit from the Semantic BMS?
3. What are the investment costs?
4. What are the step for successful deployment of the Semantic BMS?
5. How will the Semantic BMS influence the benchmarking process according to the *EN 15221*?
6. What are the security considerations when deploying the Semantic BMS?

The following sections discuss the topics established above. The goal is not to provide definitive answers but to provide the reader with broader context of the presented work.

8.1 Semantic BMS in the context of business intelligence and data warehousing

The traditional approach to storing and analysis of sensor and building automation data is data warehousing that uses On-Line Analytical processing databases (OLAP) as described in section 4.8. This section aims to compare the Semantic BMS to the OLAP approach and propose a manner of collaboration between existing data warehouses and the Semantic BMS.

First, key features of the OLAP are reviewed. Such features may be considered drawbacks for certain use cases in the building automation.

The OLAP warehouse is not a primary source of a data. The OLAP solutions usually rely on the Extract-Transform-Load (ETL) framework to import data into the warehouse. The typical sources of data are On-Line Transaction processing (OLTP) databases, text (CSV, XML) files. Generally speaking, the warehouse extracts data that are already permanently stored elsewhere.

For the automation systems, the OLTP databases are usually called Historian. The Historian solutions support directly various automation protocols to download and store the historical data. Such products often provide certain OLAP features, such as continuous data aggregation. Examples of such products are Delta Controls Historian (pure OLTP), Mango Automation (Historian, BI platform) by Infinite Automation or OSIsoft PI platform (Historian and OLAP warehouse), although they are usually commercial and expensive products. Such products can be used as a replacement for the Data Access API, either partial or complete, depending on exact capabilities of used products, required use cases and accepted limitations.

There is an ambiguous design decision in the warehouse deployment. The data are often aggregated during the Transform phase of the ETL process. This is beneficial for many use cases, as it reduces storage requirements and speeds up the often queries. However, it limits the drill-down capability of the warehouse, rendering continuous aggregation of incoming data useful mostly for rigidly defined benchmarks and reports where the method, outcome, data set and all other parameters are completely clear during the design phase. This is usually true for benchmarks based on financial indicators. However, this might not be the case in building automation use case scenarios. It prevents post-mortem, ad-hoc drill-down to the source data that can be useful when inspecting an unexpected behavior of the system and finding the outliers.

However, the main feature missing in the Historian and OLAP solutions is the semantic description of the measurements provided by the Semantic API. Available products provide only limited semantic information (network address, data point name) and force the users to organize the data points for the analysis manually. Furthermore, the

Semantic BMS provides public and standardized (by implementing the Observations & Measurements framework) schema for the data.

Nevertheless, there are reasons to use the OLAP databases. The Semantic BMS is designed as a complement to the data warehouses and Historian solutions. The Semantic BMS solves certain drawbacks of OLAP but does not aim to replace it.

There is a rule of thumb of data processing that says that is always better to bring computation as near as possible to the data than to bring data to the computation. The reason is basically 40 years of development of database technologies (especially the relational databases and the SQL language). The database solutions are extremely efficient in data handling and processing. The design of the Data Access API recognizes the strength of data warehouses and the APIs allow to use built-in features of a warehouse that is used along the Semantic BMS.

Features of the Semantic BMS can be mapped to the ETL method. The Semantic BMS ensures the Extract (Semantic API and the Data Access API) and Transform (Aggregate functions provided by the Data Access API) Phases. It has no custom data store for measurements and no tools for pushing the data to a warehouse. The reason is that it is intended for a different role in the data analysis workflow. However, it can be adapted to other roles as well.

The rest of this section elaborates on the basic scenarios in which the Semantic BMS is expected to be deployed.

The primary role of the Semantic BMS (namely the Data Access API) is to extract the data from an automation network or archival database (Historian, probably OLTP solution, or OLAP warehouse), possibly transform them by some aggregation function and pass them to the business intelligence tool. The strengths of the Semantic BMS in this use case is the added semantic information for selection of the data and consolidation of the data sources.

The Data Access API allows integration of different data sources such as distributed automation devices and archival databases so they act as one unified data source. The Data Access API also aims to provide wide range of data transformation options so the application can choose the data format that is tailored for the intended use, from raw sensor data that are passed for further processing by advanced business intelligence tools to pre-processed data that are ready to be presented by rather simple end-user visualization application. For the use with simple visualization applications, the Semantic BMS provides a variety of possible aggregations that reflect specifics of the facility management domain and that can be performed on data coming from sources that do not provide tools for data aggregation, such as automation devices.

It is worth noting that some modern and popular data analysis frameworks require copying the data to their own data store (e.g. Elastic Kibana that facilitates Elasticsearch storage). The Semantic BMS also aims to support such deployment scenarios. In this case, Semantic BMS is in fact deployed as a source for the warehouse, not as a client. To be more precise, the Semantic BMS will be likely deployed in between two data stores – Historian database for the building automation and the analytical warehouse such as Elasticsearch. In this case, the Load phase of the ETL process will be dispatched and managed by the analytics warehouse or another application, as the Semantic BMS cannot push the data into the business intelligence tool.

As indicated at the beginning of this section, data warehouses and analytical frameworks use the ETL process to get the data from permanent sources such as text files, XML files, CSV files, or both relational and NoSQL databases. However, the Semantic BMS is different from such permanent data sources. The Semantic BMS allows accessing the building automation system directly. Thus, repeated API calls with the same parameters can return different value over time. This must be considered when designing the data flow in the business intelligence framework.

However, this approach brings advantages when dealing with the domain automation. The ability to inspect data instantly, with no delay typically present in the OLAP warehouses, can be facilitated in situations when comparison of the data with the current state of the physical world (e.g. state of a device, weather conditions) is needed (although this use case lays outside the domain of facility benchmarking).

The main contribution of the presented research remains the semantic model and the Semantic API that can be used in every scenario independently on the role of the Data Access API in the deployment scenario. Without the semantic information, it is inefficient to load all the available data into the OLAP warehouse. Instead, a different process is advised when using the Semantic BMS.

For example, Semantic API can be used for selection of appropriate data points for the use case. The respective measurement data are then loaded into the business intelligence tool (possibly using the Data Access API). The retrieved measurement data must be annotated by semantic information that was obtained during the first Semantic API call. Additional enrichment of the data can be performed in this step, such as the addition of room areas from the CAFM or BIM system, or enriched by weather data. The data then can be aggregated, queried, and visualized by tools available in the business intelligence tool such as Elastic Kibana or Pentaho.

8.2 Target users

Naturally, the Semantic BMS project is not suitable for every facility and even not for every facility that has building automation system installed. There are certain prerequisites that need to be met to make the Semantic BMS beneficial. Additionally, the Semantic BMS brings certain investment costs. For that reason, it is crucial to assess the deployment costs, requirements, and potential benefits to be sure that the Semantic BMS employment is meaningful under given conditions. The main aspects that have to be taken into account are:

- Facility size;
- Facility operation costs;
- Number of facility personnel;
- Usage of facility benchmarking and reporting;
- Usage of a BIM software;
- Usage of a CAFM and BI software;
- Size of a BAS network;
- Usage of BAS protocols;
- Usage of a BAS/BMS software;
- Coverage of BIM, CAFM and BAS;

- Reliability and level of detail of a BAS project documentation.

More detailed discussion about the individual aspects follows.

The first three aspects are closely related. The size of the facility must be large enough so the facility benchmarking in general is able to render significant savings by detecting ineffective operation. The second argument related to the size of the facility is related to the facility manager's knowledge and mental capacity. If the facility managers are familiar every detail of the facility, either because the facility is small or there is a large number of facility staff, there is no added value in the semantic model – the knowledge is already available and can be probably passed on by human-to-human communication or using text document and spreadsheets. As a rule of thumb, facility benchmarking in general (and the Semantic BMS) yields more benefits as the size of facility grows and the number of facility staff shrinks.

Facility managers in large facilities usually perform reporting that in general aims to assess performance and efficiency of a facility. If there are certain benchmarking processes already established in the organization's facility management, introducing new scenarios employing Semantic BMS should not constitute any troubles. If not, there might be a reasonable cause for that – e.g. the facility is newly built and the facility management staff focuses largely on commissioning of the facility, removing malfunctions, adjusting the operation and other tasks. In such case, reporting should become a regular practice later in the facility life-cycle. However, if there are no benchmarking scenarios performed yet, there are probably simpler and more straightforward benchmarks than will bring instant results. After that initial phase, the Semantic BMS can be used for fine-tuning of the facility operation.

The next six aspects are related to the ICT environment of the organization. As the main purpose of the Semantic BMS is to integrate data from existing information system, analysis of existing software environment is essential for assessment of the Semantic BMS suitability.

The Semantic BMS requires existing BIM database to be able to meaningfully function. There is probably no point to gather data about a facility required by the Semantic BMS and nothing else. The Semantic BMS facility model is very limited and counts on additional information stored in the BIM system. Of course, a potential users of the Semantic BMS might decide to start building a BIM system because of their desire to use Semantic BMS. In such case, there must exist another use cases for the BIM system within the organization, because data acquisition for the BIM database is enormously time consuming and costly when not performed as intended – The process of a BIM model creation assumes that the data are gathered continuously from early development phases of the project. However, the mentioned other use cases are not hard to find – generally, the BIM system brings significant benefits to facility maintenance if used correctly.

The second information system that is closely related to the Semantic BMS is CAFM system, or more generally, business intelligence tools. The question is if such systems are already used by the organization. If so, employment of the Semantic BMS will probably be much smoother. There is a viable option to use existing CAFM or BI applications as a front-end for Semantic BMS middleware, simplifying the deployment both for the users and the developers.

The third and necessary ICT system related to the Semantic BMS is the BAS and BMS itself. For the BAS/BMS there are multiple considerations. First, the size of the installation(s) – number of devices, sensors and data points, number of different subsystems and technologies controlled and monitored by the BAS (e.g. HVAC, lighting, energy monitoring). The more data points and the more system incorporated, the more useful the Semantic BMS can be, as it helps to handle complexity of a large system. As with the small facility, the same rule applies for small BAS installation – the knowledge is fully contained in the minds of a facility management staff.

The second consideration relates to automation protocols used. If there is more than one automation protocol in use, the Semantic BMS can bring the benefit of data integration into the common platform, as it is able to add general semantics to the data points. The Semantic BMS is also capable of retrieving and processing data from multiple sources at once regardless on the underlying automation protocol.

Next, investigation of BAS protocols and archiving applications actually used at the facility takes place. Assessment of the implementation costs must be performed and compared to the expected benefits of the Semantic BMS deployment. Several cases might occur:

- The Semantic BMS contains implementations of data providers for given automation protocol and archive server or they are publicly available for other source. This is rather unlikely at the moment, as there are only data providers of the use case of Masaryk University implemented (i.e. for BACnet protocol).
- The data provider for the used protocol does not exist, however there is an Java protocol stack for the protocol. This should be the case for most of the other standardized automation protocols such as KNX.[1] In this situation, adding support for a new protocol is conveniently simple and straightforward when utilizing the API provided by the Semantic BMS.
- There is no Java implementation of a protocol stack, but there is an implementation written in another language. In this case, the implementation becomes more complex and requires employment of inter-process communication. The Semantic BMS does not provide direct support for such use cases at the moment.
- There is no available protocol stack implementation. This is the worst case scenario, but also the least probable. This use case requires implementation of the whole protocol stack, preferably in Java language.

Next, BMS applications that are already in use at the facility must be taken into account. Potential of BAS data for benchmarking of performance and efficiency captured attention of commercial software vendors and capabilities of their products grow with each new version. Although they do not provide as rich semantic information and such complex querying capabilities, they might be considered sufficient for smaller installations or required scenarios and desired benchmarking metrics.

Furthermore, even if the organization uses the BIM, CAFM software, and the BAS, the question of the *coverage* of such systems must be answered. In many cases, the BAS and/or the BIM systems does not fully cover every facility (e.g. site, building) that the organization operate. If only a fraction of used facilities could be covered by the

1. Java implementation of KNX protocol stack is provided by the Calimero project available form `http://calimero-project.github.io/`.

benchmarking procedure (and by the Semantic BMS), the benefits will be reduced correspondingly. However, the Semantic BMS can then still be successfully used for support of an everyday operation or for a partial benchmarking. It is worth noting that coverage by the BAS and the BIM do not necessarily overlap. If the BIM models cover 60% of organization's facilities and the BAS covers 60% of organization's facilities, in the wort case, there is only 10% of the facilities that can be covered by the Semantic BMS.

The last consideration relates to the fact that neither of the BIM and BAS do not contain some of the information required by the Semantic BMS ontology in a machine-processable way. This information has to be derived from multiple sources:

- From the knowledge of facility operators;
- By inspecting the BAS structure, algorithms and user interface;
- From the BAS project documentation or the BIM database.

Acquisition from these data sources likely cannot be fully automated. However, if the BAS project documentation is part of the BIM model or is provided in convenient structure and format and providing (e.g. Comma separated values files or Excel spreadsheets listing available data points with their attributes), it can significantly lower amount of work needed to create the Semantic BMS model. On the other hand, if the documentation is incomplete, outdated or provided in inappropriate format (the worst case probably being scans of hand-drawn drawings), creation of the data model might prove every time consuming and costly.

The following chapter is thus dedicated to suggestions how to lower the investment costs of the Semantic BMS deployment.

8.3 Semantic BMS investment costs

The investment costs to the Semantic BMS are evaluated as the highest from all the compared approaches. However, there are several arguments and considerations that can be used to justify the higher costs.

In other approaches, data have to be gathered mostly manually, as described in the previous sections. A collection of the semantic data is under way each time BMS data are used for reporting – with or without using the Semantic BMS. However, the costs of human work devoted to the identification of required data points are hard to evaluate. These repeated, additional, and hidden costs represent a factor that should be taken into account. The Semantic BMS allows to collect the information once and use it repeatedly without any additional expenses.

Next, the cost of the building automation and BMS is very high itself. However, the data that are collected in the BMS are not fulfilling their potential if they are not used for operation analysis and reporting. At large sites, usage of the Semantic BMS can help to return investments to the building automation, as a reporting and analysis can be performed more often and in more flexible manner, leading to more efficient building operation.

The Semantic BMS encounters an obstacle common to virtually every aspect of facility management and in-house ICT. The costs in these areas are not considered as investments by stakeholders. This is caused by the fact that earnings and losses from a primary business are quite easy to measure, but the facility management does not directly influence them. Savings in every-day operation accomplished by facility management and other professions are much more hidden, as they cannot be evaluated so easily. Facility benchmarking, as described in EN 15221-7, is a step toward continuous evaluation of facility management benefits. The goal of the Semantic BMS is to provide tools for such benchmarking.

To lower the initial cost, techniques from the software engineering should be used. Agile methodologies introduce incremental development cycle, as opposed to traditional *waterfall*, but expensive and inflexible development scheme. In the same manner as an application is developed by smaller steps (increments) and continuously delivered to the customer, the Semantic BMS can be populated in phases. The phases can be usually defined by use cases (required reports). It is a huge amount of work to collect all data point from building and semantically annotate them. However, to collect all the temperature and humidity sensors from a building and assign them to correct room (other attributes of the data points will be the same for all the sensors or can be derived) is tractable much better.

The last step towards lowering the investment costs of the Semantic BMS is introduced in the future work section – (semi-)automatic population of the ontology from project documentation and heuristic methods for data point meta-data generation.

8.4 Deployment process

If the Semantic BMS seems beneficial for one's facility operation, a deployment of the system must be performed. The deployment process should contain following steps, preferably in indicated order:

1. Define initial use cases – This step was probably already performed before the decision to deploy Semantic BMS was made. The use cases should be now defined in more detailed manner, to be able to decide which data have to be gathered to fill the ontology model.

2. Deploy the middleware layer to a test server.

3. Initialize the triple-store with mock-up or sample data.

4. Test system functionality, compatibility, and performance, using the provided tools (Semantic BMS Client, Test Bench).

5. Asses suitability of the Semantic BMS – First-hand experience with the system should help to understand capabilities of the system and evaluate benefits it is able to bring.

6. Define the data collection process – Based on the experience with the sample applications, it should be clear which data are needed by now.

7. Start data collection – This will probably be time consuming process. Start collecting the data at the beginning of the deployment process and collect them in parallel to other tasks, so they are ready as early as possible.

8. Implement data providers for BAS if needed. This step can be executed in parallel with the ongoing data collection.

9. Prepare front-end applications – Integrate Semantic BMS into existing tools or create new applications. This is another time-consuming task that should be performed in parallel with the data collection.

10. Create the TDB with the custom facility data.

11. Deploy the Semantic BMS into production environment.

8.5 Semantic BMS in facility benchmarking

For the evaluation of the framework usability and suitability for intended purposes, European Standard *EN 15221 Facility Management, Part 7: Guidelines for Performance Benchmarking* was used. In the following section, capabilities of the framework are presented in context of the tasks, methods, processes, and goals defined by the standard.

8.5.1 Benchmarking aspects

The EN 15221-7 defines four main aspects that form each benchmarking process. Facilitating Semantic BMS features in the benchmarking process will affect each of these aspects, making specific facility management benchmarking methods more effective and flexible. Further description of the aspects follows, together with the applications of the Semantic BMS.

Perspective of the initiator: In the standard, two initiators are considered – customer/consumer and internal/provider.

The Semantic BMS is intended mainly for internal benchmarking. However, in the internal benchmarking, both sides are present – the organization is both provider and consumer of the service (in this case, the service is building operation aided by automation systems).

When benchmarking from the perspective of provider, the main tasks where Semantic BMS can be utilized are equipment reliability and operation costs benchmarks. When benchmarking from the perspective of consumer, Semantic BMS can be plugged into tasks related to quality of the building environment (e. g. room temperature and humidity).

The objectives of the benchmarking process: The standard proposes several most often objectives to the benchmarking process:

- Find new ideas;
- Get data to prepare a main decision or to resolve disputes;
- Reduce costs while maintaining a similar service level;
- Improve the service level while maintaining similar costs;
- Improve the use of resources.

For all these objectives, Semantic BMS can be used for selecting appropriate data, making data comparisons, or discover relations among data. Easy drill-down using location-based aggregation and system integration (BIM, CAFM, BAS) allows facility managers to simplify task such as outliers in energy consumption, environmental variables (too high or too low temperatures), which can be used in any of the mentioned objectives.

The point in time at which the benchmarking is performed: The Semantic BMS allows rapid setup time from the first benchmark intent and idea to data gathering, as well as frequent repetition of the benchmarking process and flexible readjustment.

The benchmarking sample: The EN-155221-7 standard mentions two most frequent sample types – samples from similar sector of primary activities of organization and those with different sectors of primary activities. Since the framework allows for flexible data selection, filtering, and grouping, it brings significant flexibility of sample selection into the area of internal benchmarking.

For large organizations, such as universities, building automation data, due to its detail, can be used for comparison among different types of facilities – differing in its use (offices, lecture rooms, laboratories), materials used (brick, concrete), installed equipment, users (different organization units) and various other aspects.

8.5.2 Benchmarking purpose and scope

Further, the EN-15221-7 defines several most frequent benchmarking purposes, as well as benchmarking scope. The benchmarking scope consist of several components, namely Content, Measure, Comparator, Domain and Frequency. The benchmarking scope differs depending on the purpose of a benchmarking exercise, as presented in Table 8.1, which is adapted from the EN-15221-7.

Table 8.1: Typical benchmarking purposes (adopted from the *EN 15221-7*)

	content			**measure**						**comparator**			**domain**			**frequency**		
purpose	strategy	process	performance	finance	space	environment	service quality	satisfaction	productivity	internal	competitor	cross-sector	local	national	international	one-off	periodic	continuous
Identification of improvement options	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Resource-allocation decisions	x			x			x				x	x		x	x	x		
Prioritisation of problem areas		x	x	x	x	x	x	x	x	x			x			x		
Verification legal compliance		x			x	x				x	x		x	x		x	x	
Identification of best practices	x	x		x	x	x	x	x	x		x	x		x	x	x		
Budget review and planning	x			x	x	x				x	x		x	x		x	x	x
Alignment with corporate objectives	x			x	x	x				x	x	x	x	x	x	x	x	
Improvement of process effectiveness		x		x	x	x	x	x	x	x	x		x	x		x	x	
Assessment of property performance			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
- Assessment of cost effectiveness			x	x						x	x	x	x	x	x	x	x	x
- Evaluation of floor space usage			x		x					x	x		x	x		x	x	
- Appraisal of environmental impacts			x			x				x	x		x	x		x	x	
- Assessment of service quality shortfalls			x				x			x	x	x	x	x	x	x	x	x
- Evaluation of end-user satisfaction			x					x		x	x		x	x		x	x	
- Appraisal of individual productivity			x						x	x	x		x	x		x	x	

The usability of the SBMS framework for different purposes will be further described in the following text.

This goal of "Identification of improvement options" can be achieved by utilizing SBMS framework in situations when an unsatisfying state can be detected and/or derived from the building automation data.

SBMS framework can be easily utilized for "Budget review and planning" purpose using energy consumption data, that are prevalent part of building automation data. Furthermore, compared to the consumption data available from energy vendor invoices, the building automation data accessed via SBMS framework provides metadata (scope of the measured value, source device) and greater detail (the BAS can be equipped with larger number of energy meters than is necessary for invoicing).

Building environment data (room temperatures, humidity) can be used for indirect “Evaluation of end-user satisfaction” – BAS data can be used to verify compliance with desired workplace environment conditions. Energy consumption data can be also used for “Appraisal of environmental impacts”. Combination of detailed data (room environment), aggregation functions and rather coarse-grained data (energy consumption) can be used for “Assessment of cost effectiveness” (e. g. detection of rooms that are unnecessarily cooled to low temperatures or heated to high temperatures).

Further, different components of the benchmarking scope and relation to use of the SBMS will be discussed.

Content: Content component describes the operation level where the benchmarking focuses. As can be seen on the previous examples, the SBMS framework can be successfully used on the “Strategic” level and “Performance” level. For the “Process” level, benefits of automation data and the SBMS framework are not as evident, however BAS data can be in same cases used as a base for process evaluation.

Measure: Considering the Measure component, the EN-15221-7 distinguishes two main categories of measures – qualitative and quantitative. Quantitative measures are defined as tangible and collected using information systems, qualitative as intangible and collected through focus groups, employee surveys and other methods. BAS data are purely quantitative measures. However, because of some unique properties (among traditionally used facility management data sources), they can be used to partially replace some of the qualitative measures proposed by the EN standard. As an example, we can consider employee satisfaction with room temperature and humidity. We can replace this qualitative measure with quantitative using BAS data, as mentioned before. It is possible to collect data about room temperatures and humidity from the sensors and verify compliance to defined criteria. It is worth noting that the result of the benchmarking process will be different. In one case, the result is information, if individuals are satisfied with their workplace conditions (user poll). In the other, if workspace conditions are aligned with defined generally acceptable conditions. However, the individual desires are no necessarily aligned with the generally acceptable conditions.

Comparator: From the point of view of the Comparator component, the BAS data are intended purely for internal comparisons, i.e. for use within one organization.

Domain: Use for BAS data is not limited to any specific Domain (local / national / international), however different climate and weather conditions in case of international domain can account for differences in measurements coming from different parts of the world. The SBMS framework can be used to overcome this issue by enriching plain BAS measurements with semantic data that can be used for relating with weather conditions data source.

Frequency: The BAS data and SBMS framework can be successfully used in all three categories of benchmarking according to the aspect of "Frequency". For the "One-off" benchmark, capabilities of the SBMS semantic layer can be utilized. SBMS allows facility managers to quickly select desired data based on clear, understandable criteria, shortening and simplifying the process from benchmark definition to data retrieval. For the periodic and continuous benchmarking, plugging SBMS layer into the definition of the benchmarking report simplifies management of such reports. When using semantic descriptors in report definitions, physical changes in the physical BAS does not affect reports' functionality – the only place where update needs to be performed is the ontology store of the SBMS framework. Furthermore, nature of the BAS measurements allows continuous benchmarking with virtually every desired period, which is not usually possible with other types of facility management data (that is updated in much longer periods).

8.5.3 Benchmarking outputs

As benchmarking outputs, the EN-15221-7 proposes indicative list of key performance indicators (KPI), specifically key ratio comparators (KRC), noting that since there is no absolute baseline, the indicators must be compared as ratios among similar, comparable facilities. The standard defines six KPI classes (Financial, Spatial, Environmental, Service quality, Satisfaction, Productivity). The SBMS framework can be best used when collecting the Financial, Environmental and Satisfaction KPIs.

Environmental KRC contain indicators derived from the energy and water consumption. That makes it natural domain of use for BAS data and the SBMS framework, facilitating its querying capabilities and integration with the BIM data store. Furthermore, the BAS data allows to perform in-depth drilldown, if a facility is equipped with secondary energy metering devices. It also allows for increased frequency of benchmarking process, since BAS measurements are more frequent than consumption measurements intended for invoicing. Increased frequency can be used for example for comparing energy costs during weekdays and weekends or during terms and holidays in case of a university.

Financial KRCs that can be derived from the BAS data are similarly those derived from the energy consumption, because energy costs account for significant part of overall facility management costs. The SBMS framework largely simplifies collection of KPIs such as "Energy cost per square meter" or "Energy cost per employee", that can be later plugged as an input into the computation of various Facility Management Cost KRCs.

As mentioned in previous text, considering Satisfaction KRCs, sensor measurements of workplace environment can be used together with facility user surveys to evaluate compliance with desired workplace conditions (temperature, humidity).

8.5.4 Benchmarking process

The EN-15221-7 standard defines benchmarking as a process with following distinct phases:

1. Preparing:
    - (a) Set objectives (purpose and scope)
    - (b) Define methodology (indicators and benchmarks)
    - (c) Select partners (peers and code of conduct)

2. Comparing:
    (a) Collect data (collect and validate)
    (b) Analyze data (determine and normalize)
    (c) Determine gaps (compare and explain)
    (d) Report findings (communicate and discuss)
3. Improving:
    (a) Develop action plan (tasks and milestones)
    (b) Implement plan (change and monitor)
    (c) Process review (review and re-calibrate)

The SBMS framework significantly influences several steps of the benchmarking process for certain use cases, as discussed in the following text.

In the preparation phase, the EN-15221-7 clarifies difference between indicator, measure and benchmark. An indicator describes type of collected value(s), such as "Energy consumption per square meter". Measure is (usually aggregate) characteristic property of the collected data, such as minimum, maximum, mean, median or other statistical measure. A benchmark is the selected measure of the indicator used for performing comparison between the peers and for defining a goal of the benchmarking process.

Plugging the BAS data into the benchmarking process allows to define more precise and more specific indicators based on sensor measurements, as well as it allows for broader range of partners/peers (i.e. entities that are compared), that can be conveniently selected using semantic description provided by the SBMS framework. Furthermore, the SBMS framework is intended to be used as a base for business intelligence applications utilizing BAS data and providing simple and flexible definition and calculation of various measures.

In the stage of benchmark comparison, the SBMS framework aims to simplify and accelerate collection of BAS data (which is prevalent issue in "vanilla" BAS systems), with extra advantage if benchmark is intended to be run repeatedly, when the SBMS framework provides convenient method to describe desired data using semantic metadata.

The task of data analysis and reporting is the part of benchmarking process where existing or newly developed business intelligence applications may be utilized, using APIs provided by the SBMS framework for data selection, aggregation and collection. In later phases, the SBMS frameworks is utilized in monitoring the implemented changes and achieved goals in the same way as in previous phases.

8.5.5 Benefits of the Semantic BMS

The Semantic BMS allows to use building automation data in benchmarking process. The BAS data provide more detail and freshness. The presented novel approach to facility benchmarking facilitates continuous improvement of the facility performance, efficiency and sustainability. The benchmark runs can be repeated frequently, adjusted to specific data sets and tweaked precisely to meet current requirements of the organization and facility manager. Furthermore, the BAS data allow for complex drill-down into the source data which significantly simplifies detection of possible problematic aspects of building operation.

The querying capabilities are designed to support the Deming cycle management method (plan-do-check-act) in facility benchmarking which is naturally needed for meaningful benchmarking and is recommended by the EN 15221-7.

8.6 Summary

This chapter discussed the role of the Semantic BMS in facility benchmarking and in facility management in general. Additionally, this chapter elaborated on security considerations and comparison to other approaches to data analysis. The main parts of the chapter were dedicated to discussion about suitability of the Semantic BMS for different users and requirements that must be met for the Semantic BMS to bring expected benefits to the users. The chapter provided guidelines to assess benefits of the presented work.

9 Future work

Although the specification of the Semantic BMS model and middleware reached a stable version that covers all the intended capabilities, there are certain areas of that deserve further research in order to make the Semantic BMS easier to use or to increase its capabilities. Selected areas of future work are described in the rest of this chapter.

9.1 Applications of the Semantic BMS

Naturally, the Semantic BMS, being a middleware layer, is only a half way through successful use of building automation data in facility benchmarking. Easy to use, comfortable, and convenient end-user applications must be developed to provide facility managers with tools (and data) they can use.

The motivation for this research originated in an attempt to analyze building operation by statistical and machine learning methods. The problem was to get the data that could be later analyzed – the workflow proved to complex and time consuming. The Semantic BMS was driven by a desire to remove the complexity of data gathering. This problem was successfully solved. The Semantic BMS significantly simplifies the data acquisition process.

The next step is thus to resume the attempts to analyze the data to support facility management decisions. The data can be examined using established business intelligence methods, however there is definitely a room for new experimental approaches. The future research topics might address (un)reliability of the sensor data or data mining methods from time series.

Besides the analytical methods, user interfaces are another area of future work. Examining of building automation data in map-like schemes (e.g. floor plans, site maps) or in 3D BIM models seem as promising approach to provide quick assessment of the data set. The Semantic BMS supports such visualization methods due to its close integration with the BIM models.

9.2 Automation of data gathering

The Semantic BMS allows for storing collected data for repeated use. Although the ontology provides a significant leap forward in re-usability of once collected data, manual process of data gathering is still complex and time-consuming. A problem of (semi-)automated methods for populating the ontology repository presents a significant challenge for further research. The most promising approach to simplification of this process relies on deeper integration with the Building Information Model. The easiest way of collecting needed data is during the design phase of the installation itself. Incorporating the acquisition of the semantic meta-data would significantly simplify the process of linking the data points with the associated BIM elements. Even the traditional documentation consisting of CAD drawings, spreadsheets and other documents can be prepared in a way that lowers the complexity of the task (however, such documentation can provide only limited capabilities compared to the BIM). Another direction of a research leads toward heuristic methods of semantic data acquisition. The required semantic attributes can be derived (with a limited level of certainty) either from the BMS (e.g. names of the data points – if the name contains substring such as "temp", "temperature" or "_T_", it might be a temperature sensor or set-point value) or from the BIM (e.g. type of the device might in some cases be used to derive the observed property).

9.3 Deriving more complex model

The information available in the ontology model invites to use inference techniques and rule engines (e.g. using SWRL language) to derive new knowledge. This is very powerful feature of the OWL. Although the reasoning plays significant role in querying capabilities of the Semantic API, the Semantic BMS at the moment uses only a fraction of reasoning capabilities of the ontology language.

This holds a potential for future research related to the project. Reasoning capabilities of the OWL could be used to infer new knowledge about processes occurring in the automation system. The observed properties influence each other in a way that is not easy to derive form the brief overview of the automation system. The Semantic BMS currently provides properties to model such relations (see Section 6.3.6 for more details). However, exploring the relations and maintaining them in the model is rather complicated task. Advanced reasoning over the asserted statements (axioms) in the Semantic BMS could be used to discover such relations automatically.

Alternatively, data mining techniques could be used to heuristic exploration of correlations in trends of the sensor readings. Such information could be used in the same way as inferred information – to find and derive new knowledge about the large web of processes and relations occurring and happening in the facility, that influence its operation costs and user comfort.

10 Conclusions

The Semantic BMS project aims to provide additional semantics to data points available in building automation systems.

The project design was based on long term experience with the operation of University Campus of Masaryk University, serving as an example of a large "intelligent" site equipped with hundreds of automation devices. Suitability of the model was assessed by use case scenarios and by performance benchmarks performed on artificially generated data.

The research presents novel semantic model that is adapted to the domain of building automation and addresses multiple differences between the domain of traditional sensor measurements and the field of building automation. The model is designed to be automation protocol independent and describes available data in a way that can be utilized during decision support tasks needed for building performance analysis, evaluation and improvement.

The model itself provides structured, rigid description of building automation data, that can be queried and results can be further machine-processed. On top of the semantic model, the middleware layer is built. It provides complex querying capabilities unavailable in current solutions. Flexibility of the APIs brings unprecedented ease of data selection to the domain of building automation.

The Semantic BMS focuses on providing tools for user-friendly, flexible, and dynamic querying over the building automation data using criteria that are comprehensible for facility managers, which distinguishes it from other available solutions. Using additional semantic layer, the process of obtaining and inspecting key performance indicator data is significantly simplified. The Semantic BMS project brings building operation data with high level of detail into the field of facility management.

The proposed solution can be adjusted to be used in other domains apart from the facility management. Event though the other domains are out of scope of the research, expected possible uses include:

- Querying over environmental sensor data described by the Observations & Measurements framework;

- Monitoring and fault detection applications for the building automation;
- Integration of the building automation installations into the Smart City network.

Bibliography

1. ALWAER, H.; CLEMENTS-CROOME, D. J. Key performance indicators (KPIs) and priority setting in using the multi-attribute approach for assessing sustainable intelligent buildings. *Building and Environment* [online]. 2010, vol. 45, no. 4, pp. 799–807 [visited on 2015-06-05]. ISSN 0360-1323. Available from DOI: `10.1016/j.buildenv.2009.08.019`.
2. MASARYK UNIVERSITY. *University Campus Bohunice*. Available from: `http://www.muni.cz/kampus`. Referred on August 24, 2014.
3. MANAGEMENT OF THE UNIVERSITY CAMPUS AT BOHUNICE. *About Us*. Available from: `http://www.muni.cz/ucb/general/about`. Referred on August 24, 2014.
4. KUČERA, Adam; GLOS, Petr; PITNER, Tomáš. Fault detection in building management system networks. In: SLANINA, Zdeněk (ed.). *12th IFAC Conference on Programmable Devices and Embedded Systems, PDeS 2013* [elektronická verze "online"]. Neuveden: International Federation of Automatic Control, 2013, pp. 416–421. ISBN 978-3-902823-53-3. Available from DOI: `http://dx.doi.org/10.3182/20130925-3-CZ-3023.00027`.
5. KRIKSCIUNIENE, Dalia; PITNER, Tomáš; KUČERA, Adam; SAKALAUSKAS, Virgilijus. Sensor Network Analytics for Intelligent Facility Management. In: TSIHRINTZIS, George A.; VIRVOU, Maria; WATANABE, Toyohide; JAIN, Lakhmi C.; HOWLETT, Robert J. (eds.). *Proceedings of the 6th International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS2013)*. Amsterdam: IOS Press, 2013, pp. 212–221. ISBN 978-1-61499-261-5. Available from DOI: `http://dx.doi.org/10.3233/978-1-61499-262-2-212`.
6. KRIKSCIUNIENE, Dalia; PITNER, Tomáš; KUČERA, Adam; SAKALAUSKAS, Virgilijus. Data Analysis in the Intelligent Building Environment. *International Journal of Computer Science & Applications*. 2014, vol. 11, no. 1, pp. 1–17. ISSN 0972-9038. Available also from: `http://www.tmrfindia.org/ijcsa/v11i1.html`.

7. KUČERA, Adam; PITNER, Tomáš. Intelligent Facility Management for Sustainability and Risk Management. In: HŘEBÍČEK, J.; SCHIMAK, G.; KUBÁSEK, M.; RIZZOLI, A.E. (eds.). *Environmental Software Systems. Fostering Information Sharing*. Berlin Heidelberg: Springer, 2013, pp. 608–617. ISBN 978-3-642-41150-2. Available from DOI: `http://dx.doi.org/10.1007/978-3-642-41151-9\_57`.
8. ASFANDANDEANDYAR, Muhammad; KUČERA, Adam; PITNER, Tomáš. Semantic Web Technology for Building Information Model. In: ANDREAS HOLZINGER, et al. (ed.). *Proceedings of the 9th International Conference on Software Engineering and Applications, Vienna, Austria*. SciTePress, 2014, pp. 109–116. ISBN 978-989-758-036-9. Available from DOI: `http://dx.doi.org/10.5220/0004999201090116`.
9. ASFANDANDEANDYAR, Muhammad; KUČERA, Adam; PITNER, Tomáš. Smart buildings: Semantic web technology for building information model and building management system. In: *International Conference on Data and Software Engineering (ICODSE) 2014*. IEEE, 2014, pp. 1–6. ISBN 978-1-4799-7996-7. Available from DOI: `http://dx.doi.org/10.1109/ICODSE.2014.7062671`.
10. KUČERA, Adam; PITNER, Tomáš. Semantic BMS: Ontology for Analysis of Building Automation Systems Data. In: *DOCEIS 2016: Technological Innovation for Cyber-Physical Systems*. 2016, vol. 470/2016, pp. 46–53. IFIP Advances in Information and Communication Technology. ISBN 978-3-319-31164-7. ISSN 1868-4238. Available from DOI: `http://dx.doi.org/10.1007/978-3-319-31165-4_5`.
11. KUČERA, Adam; PITNER, Tomáš. Semantic BMS: Ontology for Analysis of Building Operation Efficiency. In: *ISESS 2017: International Symposium on Environmental Software Systems*. 2017. To appear.
12. BACNET STACK: AN OPEN SOURCE BACNET PROTOCOL STACK FOR EMBEDDED SYSTEMS. *About this Project*. Available from: `http://bacnet.sourceforge.net/`. Referred on August 24, 2014.
13. BACNET FOR JAVA. *Project Information*. Available from: `http://bacnet4j.sourceforge.net/`. Referred on August 24, 2014.
14. SCADA ENGINE. *BACnet Server API*. Available from: `http://www.scadaengine.com/software3.html`. Referred on August 24, 2014.

15. AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS, INC. *ANSI/ASHRAE Addendum to ANSI/ASHRAE Standard 135-2008*. Available from: `https://www.ashrae.org/File%20Library/docLib/Public/20100305_135_2008_t_for_posting.pdf`. Referred on August 24, 2014.

16. DELTA CONTROLS INC. *Historian*. Available from: `http://www.deltacontrols.com/products/facilities-management/supervisory-software/historian`. Referred on August 24, 2014.

17. INTERNATIONAL FACILITY MANAGEMENT ASSOCIATION. *What is facility management?* 2014. Available from: `http://ifma.org/about/what-is-facility-management`. Referred on August 24, 2014.

18. EUROPEAN COMMITTEE FOR STANDARDIZATION. *EN 15 221 1-7 – Facility management*. 2006-2012.

19. AMERICAN NATIONAL STANDARDS INSTITUTE. *ANSI/BOMA Z65 1-6 – Standard Methods of Measurement*. 2009-2012. Overview of parts is available at `http://www.boma.org/standards/`.

20. EUROPEAN UNION. *Directive 2002/91/EC On the Energy Performance of Buildings*. 2002.

21. POEL, Bart; CRUCHTEN, Gerelle van; BALARAS, Constantinos A. Energy performance assessment of existing dwellings. *Energy and Buildings*. 2007, vol. 39, no. 4, pp. 393–403. ISSN 0378-7788. Available from DOI: `http://dx.doi.org/10.1016/j.enbuild.2006.08.008`.

22. PÉREZ-LOMBARD, Luis; ORTIZ, José; GONZÁLEZ, Rocío; MAESTRE, Ismael R. A review of benchmarking, rating and labelling concepts within the framework of building energy certification schemes. *Energy and Buildings*. 2009, vol. 41, no. 3, pp. 272–278. ISSN 0378-7788. Available from DOI: `http://dx.doi.org/10.1016/j.enbuild.2008.10.004`.

23. TRONCHIN, Lamberto; FABBRI, Kristian. Energy Performance Certificate of building and confidence interval in assessment: An Italian case study. *Energy Policy*. 2012, vol. 48, pp. 176–184. ISSN 0301-4215. Available from DOI: `http://dx.doi.org/10.1016/j.enpol.2012.05.011`. Special Section: Frontiers of Sustainability.

24. VISSCHER, Henk; MAJCEN, Dasa; ITARD, LCM. Effectiveness of energy performance certification for the existing housing stock. In: *RICS COBRA 2012, Proceedings of the Construction, Building and Real Estate Conference, Tempe, AZ: Arizona State University*. 2012, pp. 130–148.

25. HOGG, Nick; BOTTEN, Chris. *A Tale of Two Buildings – Are EPCs a true indicator of energy efficiency?* [Published by Jones Lang LaSalle and Better Building Partnership]. 2012. Available from `http://www.betterbuildingspartnership.co.uk/download/bbp-jll---a-tale-of-two-buildings-2012.pdf`. Referred on August 25, 2014.

26. BUNSE, Katharina; VODICKA, Matthias; SCHÖNSLEBEN, Paul; BRÜLHART, Marc; ERNST, Frank O. Integrating energy efficiency performance in production management gap analysis between industrial needs and scientific literature. *Journal of Cleaner Production*. 2011, vol. 19, no. 67, pp. 667–679. ISSN 0959-6526. Available from DOI: `http://dx.doi.org/10.1016/j.jclepro.2010.11.011`.

27. GORP, John C. Van. Using key performance indicators to manage energy costs. *Strategic planning for energy and the environment*. 2005, vol. 25, no. 2, pp. 9–25.

28. VIKHOREV, Konstantin; GREENOUGH, Richard; BROWN, Neil. An advanced energy management framework to promote energy awareness. *Journal of Cleaner Production*. 2013, vol. 43, pp. 103–112. ISSN 0959-6526. Available from DOI: `http://dx.doi.org/10.1016/j.jclepro.2012.12.012`.

29. JURÁNKOVÁ, Lucie. *Energy management v kontextu CAFM nástrojů [Energy management in the Context of CAFM Tools]*. 2013. Master thesis. Masaryk University, Faculty of Informatics. Supervised by Tomáš PITNER. Available from `http://is.muni.cz/th/256726/fi_m/`.

30. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 16739:2013 – Industry Foundation Classes*. 2009-2012.

31. WONG, JKW; LI, Heng; WANG, SW. Intelligent building research: a review. *Automation in construction*. 2005, vol. 14, no. 1, pp. 143–159.

32. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 16484-5 – Building automation and control systems (BACS) – Part 5: Data communication protocol*. 2014.

33. AMERICAN NATIONAL STANDARDS INSTITUTE. *ANSI/CEA-709.1-B Control Network Protocol Specification*. 2002.

34. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO/IEC 14543 Information technology – Home Electronic System (HES) architecture*. 2002-2012.

35. COUNCIL OF EUROPEAN UNION. *Council directive 2008/114/EC of 8 December 2008 on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection*. 2008. Available from: `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=cellar:ba51b03f-66f4-4807-bf7d-c66244414b10`. Referred on 3rd September 2017.

36. ČESKO. *430/2010 Sb.ZÁKON ze dne 21. prosince 2010, kterým se mění zákon č. 240/2000 Sb., o krizovém řízení a o změně některých zákonů (krizový zákon), ve znění pozdějších předpisů*. 2010. In Czech. Current version of the act (after additional amendments) available from: `https://www.zakonyprolidi.cz/cs/2000-240/zneni-20170801`. Referred on 3rd September 2017.

37. LANGNER, Ralph. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*. 2011, vol. 9, no. 3, pp. 49–51.

38. FALLIERE, Nicolas; MURCHU, Liam O; CHIEN, Eric. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*. 2011, vol. 5, no. 6.

39. U.S. DEPARTMENT OF COMMERCE – NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Framework for Improving Critical Infrastructure Cybersecurity*. Available from: `https://www.nist.gov/cyberframework`. Referred on March 14, 2017.

40. BACNET INTERNATIONAL. *Frequently Asked Questions: Why shouldn't I use OPC instead of BACnet?* Available from: `http://www.bacnetinternational.org/faq`. Referred on August 24, 2014.

41. ASHRAE SSPC 135. *BACnet.org: BACNet Overview*. Available from: `http://www.bacnet.org/Overview/`. Referred on August 24, 2014.

42. ASHRAE SSPC 135. *BACnet.org: Addenda and Companion Standards*. Available from: `http://www.bacnet.org/Addenda/`. Referred on August 24, 2014.

43. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 16484-6 – Building automation and control systems (BACS) – Part 6: Data communication conformance testing*. 2014.

44. ASHRAE SSPC 135. *BACnet.org: Vendor IDs*. Available from: `http://www.bacnet.org/VendorID/`. Referred on August 24, 2014.

45. GOLFARELLI, Matteo; MAIO, Dario; RIZZI, Stefano. The Dimensional Fact Model: A Conceptual Model For Data Warehouses. *International Journal of Cooperative Information Systems*. 1998, vol. 7, pp. 215–247.

46. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 Primer*. Available from: `http://www.w3.org/TR/rdf11-primer/`. Referred on August 24, 2014.

47. THE APACHE SOFTWARE FOUNDATION. *Apache Jena: A free and open source Java framework for building Semantic Web and Linked Data applications.* Available from: `https://jena.apache.org/`. Referred on August 24, 2014.

48. WORLD WIDE WEB CONSORTIUM. *OWL 2 Web Ontology Language Primer (Second Edition).* Available from: `http://www.w3.org/TR/2012/REC-owl2-primer-20121211/`. Referred on August 24, 2014.

49. PAUWELS, Pieter; ZHANG, Sijie; LEE, Yong-Cheol. Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*. 2017, vol. 73, pp. 145–165. ISSN 0926-5805. Available from DOI: `http://doi.org/10.1016/j.autcon.2016.10.003`.

50. HOEHNDORF, Robert. *What is an upper level ontology?* [`http://ontogenesis.knowledgeblog.org/740`]. 2010. Available also from: `http://ontogenesis.knowledgeblog.org/740`.

51. W3C SEMANTIC SENSOR NETWORK INCUBATOR GROUP. *Semantic Sensor Network Ontology*. Available from: `http://www.w3.org/2005/Incubator/ssn/ssnx/ssn`. Referred on August 24, 2014.

52. HENSON, Cory A.; PSCHORR, Josh K.; SHETH, Amit P.; THIRUNARAYAN, Krishnaprasad. SemSOS: Semantic Sensor Observation Service. In: *Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems* [online]. Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–53 [visited on 2015-06-05]. CTS '09. ISBN 978-1-4244-4584-4. Available from DOI: `10.1109/CTS.2009.5067461`.

53. COMPTON, Michael; HENSON, Cory; LEFORT, Laurent; NEUHAUS, Holger; SHETH, Amit. A Survey of the Semantic Specification of Sensors. In: *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522*. Washington DC: CEUR-WS.org, 2009, pp. 17–32. SSN'09. Available also from: `http://dl.acm.org/citation.cfm?id=2889933.2889935`.

54. COMPTON, Michael et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web* [online]. 2012, vol. 17, pp. 25–32 [visited on 2015-05-20]. ISSN 1570-8268. Available from DOI: `10.1016/j.websem.2012.05.003`.

55. PROBST, Florian. Ontological Analysis of Observations and Measurements. In: *Geographic Information Science: 4th International Conference, GIScience 2006, Münster, Germany, September 20-23, 2006. Proceedings*. Ed. by RAUBAL, Martin; MILLER, Harvey J.; FRANK, Andrew U.; GOODCHILD, Michael F. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 304–320. ISBN 978-3-540-44528-9. Available from DOI: `10.1007/11863939\_20`.

56. JANOWICZ, Krzysztof; COMPTON, Michael. The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. *3rd International Workshop on Semantic Sensor Networks, Shanghai, China, November 7, 2010*. 2010, no. 668, pp. 15.

57. MICROSOFT CORPORATION. *What is COM?* Available from: `https://www.microsoft.com/com/default.mspx`. Referred on August 24, 2014.

58. OBJECT MANAGEMENT GROUP. *Documents Associated With CORBA, 3.3*. Available from: `http://www.omg.org/spec/CORBA/3.3/`. Referred on August 24, 2014.

59. ORACLE CORPORATION. *Remote Method Invocation Home*. Available from: `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html`. Referred on August 24, 2014.

60. MICROSOFT CORPORATION. *.NET Remoting*. Available from: `http://msdn.microsoft.com/en-us/library/vstudio/72x4h507(v=vs.100).aspx`. Referred on August 24, 2014.

61. ORACLE CORPORATION. *Enterprise JavaBeans Technology*. Available from: `http://www.oracle.com/technetwork/java/javaee/ejb/index.html`. Referred on August 24, 2014.

62. OSGI ALLIANCE. *OSGi Alliance Specifications*. Available from: `http://www.osgi.org/Specifications/HomePage`. Referred on August 24, 2014.

63. MICROSOFT CORPORATION. *Windows Communication Foundation [.NET Framework 4.0]*. Available from: `http://msdn.microsoft.com/en-us/library/dd456779(v=vs.110).aspx`. Referred on August 24, 2014.

64. WORLD WIDE WEB CONSORTIUM. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. Available from: `http://www.w3.org/TR/wsdl20-primer/`. Referred on August 24, 2014.

65. WORLD WIDE WEB CONSORTIUM. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Available from: `http://www.w3.org/TR/soap12-part1/`. Referred on August 24, 2014.

66. INTERNET ENGINEERING TASK FORCE. *RFC 7159 The JavaScript Object Notation (JSON) Data Interchange Format*. Available from: `http://tools.ietf.org/html/rfc7159`. Referred on August 24, 2014.

67. PAPAZOGLOU, Mike P. Service-oriented computing: Concepts, characteristics and directions. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering*. 2003, pp. 3–12.

68. ERL, Thomas. *Service-oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, 2004. Charles F. Goldfarb definitive XML series. ISBN 9780131428980. Available also from: `http://books.google.cz/books?id=9NtQAAAAMAAJ`.

69. ARSANJANI, A. et al. The SOA Manifesto. *SOA Manifesto*. 2009.

70. WIKIPEDIA. *Service-oriented architecture — Wikipedia, The Free Encyclopedia*. 2017. Available also from: `%5Curl%7Bhttps://en.wikipedia.org/w/index.php?title=Service-oriented_architecture&oldid=770220184%7D`. [Online; accessed 19-March-2017].

71. CHAPPELL, David A. *Enterprise Service Bus: Theory in Practice*. O'Reilly Media, 2004. Theory in practice. ISBN 9781449391096. Available also from: `http://books.google.cz/books?id=Uhue3faV2mwC`.

72. INMON, William H. *Building the Data Warehouse*. Wiley, 1992. A Wiley-QED publication. ISBN 9780471569602. Available also from: `http://books.google.cz/books?id=bHP1Wc4CdGEC`.

73. KIMBALL, Ralph. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. Wiley, 1996. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. ISBN 9780471153375. Available also from: `http://books.google.cz/books?id=8IMpAQAAMAAJ`.

74. CHAUDHURI, Surajit; DAYAL, Umeshwar. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.* 1997, vol. 26, no. 1, pp. 65–74. ISSN 0163-5808. Available from DOI: `10.1145/248603.248616`.

75. CHAUDHURI, Surajit; DAYAL, Umeshwar; NARASAYYA, Vivek. An Overview of Business Intelligence Technology. *Commun. ACM*. 2011, vol. 54, no. 8, pp. 88–98. ISSN 0001-0782. Available from DOI: `10.1145/1978542.1978562`.

76. FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From data mining to knowledge discovery in databases. *AI magazine*. 1996, vol. 17, no. 3, pp. 37.

77. WITTEN, Ian H.; FRANK, Eibe. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Elsevier Science, 2005. The Morgan Kaufmann Series in Data Management Systems. ISBN 9780080477022. Available also from: `http://books.google.cz/books?id=QTnOcZJzlUoC`.

78. LUCKHAM, David. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2002. ISBN 9780201727890. Available also from: `http://books.google.cz/books?id=AN1QAAAAMAAJ`.

79. ROKYTA, Jan. *Datový sklad k technologiím budov [Data Warehouse for Building Management]*. 2013. Master thesis. Masaryk University, Faculty of Informatics. Supervised by Tomáš PITNER. Available from `http://is.muni.cz/th/256630/fi_m/`.

80. POPELÍNSKÝ, Lubomír; GLOS, Petr. Towards Detection of Anomalies in Building Management Data. In: *15th IBIMA conference on Knowledge Management and Innovation: A Business Competitive Edge Perspective*. Cairo: IBIMA Publishing, 2010, pp. 664–669. ISBN 978-0-9821489-4-5. Available also from: `http://www.ibima.org/CA2010/index.html`.

81. KUČERA, Adam. *Komplexní zpracování událostí v systémech pro správu budov [Complex Event Processing in Building Management Systems]*. 2012. Master thesis. Masaryk University, Faculty of Informatics. Supervised by Tomáš PITNER. Available from `http://is.muni.cz/th/255658/fi_m/`.

82. BISHOP, Matthew A. *The Art and Science of Computer Security*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201440997.

83. MENZEL, K; WEISE, M; LIEBICH, T; VALMASEDA, C. Capabilities of IFC 4 for Advanced Building Performance Management. In: *Proceedings of the 2nd Central European Symposium on Building Physics*. Vienna, Austria: Vienna University of Technology - Faculty of Architecture and Regional Planning, 2013, vol. 2013, pp. 467–474. ISBN 978-3-85437-321-6. Available also from: `http://info.tuwien.ac.at/cesbp/Files/CESBP2013_SammelmappeFinal_29.08.13.pdf`.

84. SCHUELKE, Anett et al. A middleware platform for integrated building performance management. In: *Proceedings of the 2nd Central European Symposium on Building Physics*. Vienna, Austria: Vienna University of Technology - Faculty of Architecture and Regional Planning, 2013, vol. 2013, pp. 459–466. ISBN 978-3-85437-321-6. Available also from: `http://info.tuwien.ac.at/cesbp/Files/CESBP2013_SammelmappeFinal_29.08.13.pdf`.

85. WANG, Hongxia; GLUHAK, Alex; MEISSNER, Stefan; TAFAZOLLI, Rahim. Integration of BIM and Live Sensing Information to Monitor Building Energy Performance. In: *Proceedings of the 30th International Conference of CIB W78*. Beijing, China, 2013, pp. 344–352. Available also from: `http://itc.scix.net/cgi-bin/works/Show?w78-2013-paper-146`.

86. PAUWELS, Pieter; TERKAJ, Walter. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*. 2016, vol. 63, pp. 100–133. ISSN 0926-5805. Available from DOI: `http://doi.org/10.1016/j.autcon.2015.12.003`.

87. PAUWELS, Pieter; ROXIN, Ana. SimpleBIM: from full ifcOWL graphs to simplified building graphs. In: CHRISTODOULOU, Symeon; SCHERER, Raimar (eds.). *11th European Conference on Product and Process Modelling*. Limassol, Cyprus: CRC Press, 2016, pp. 11–18. ISBN 978-1-138-03280-4.

88. ZACH, Robert; GLAWISCHNIG, Stefan; HÖNISCH, Michael; APPEL, Regina; MAHDAVI, Ardeshir. MOST: An open-source, vendor and technology independent toolkit for building monitoring, data preprocessing, and visualization. *eWork and eBusiness in Architecture, Engineering and Construction*. 2012, pp. 97–103.

89. CHRISTOPH, Legat. Semantics to the Shop Floor: Towards Ontology Modularization and Reuse in the Automation Domain. In: EDWARD, Boje (ed.) [online]. 2014, pp. 3444–3449 [visited on 2015-05-30]. Available from DOI: `10.3182/20140824-6-ZA-1003.02512`.

90. DANIELE, Laura; HARTOG, Frank den; ROES, Jasper. Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: *Formal Ontologies Meet Industry: 7th International Workshop, FOMI 2015, Berlin, Germany, August 5, 2015, Proceedings*. Ed. by CUEL, Roberta; YOUNG, Robert. Cham: Springer International Publishing, 2015, pp. 100–112. ISBN 978-3-319-21545-7. Available from DOI: `10.1007/978-3-319-21545-7\_9`.

91. FENSEL, Anna; TOMIC, Slobodanka; KUMAR, Vikash; STEFANOVIC, Milan; ALESHIN, Sergey V.; NOVIKOV, Dmitry O. SESAME-S: Semantic Smart Home System for Energy Efficiency. *Informatik-Spektrum*. 2013, vol. 36, no. 1, pp. 46–57. ISSN 1432-122X. Available from DOI: `10.1007/s00287-012-0665-9`.

92. PLOENNIGS, Joern; SCHUMANN, Anika; LÉCUÉ, Freddy. Adapting Semantic Sensor Networks for Smart Building Diagnosis. In: *The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*. Ed. by MIKA, Peter et al. Cham: Springer International Publishing, 2014,

pp. 308–323. ISBN 978-3-319-11915-1. Available from DOI: `10.1007/978-3-319-11915-1_20`.

93. PLOENNIGS, Joern; SCHUMANN, Anika; LECUE, Freddy. Extending Semantic Sensor Networks for Automatically Tackling Smart Building Problems. In: *Proceedings of the Twenty-first European Conference on Artificial Intelligence*. Prague, Czech Republic: IOS Press, 2014, pp. 1211–1212. ECAI'14. ISBN 978-1-61499-418-3. Available from DOI: `10.3233/978-1-61499-419-0-1211`.

94. MEHDI, Muntazir; SAHAY, Ratnesh; DERGUECH, Wassim; CURRY, Edward. On-the-fly Generation of Multidimensional Data Cubes for Web of Things. In: *Proceedings of the 17th International Database Engineering &#38; Applications Symposium* [online]. New York, NY, USA: ACM, 2013, pp. 28–37 [visited on 2015-05-12]. IDEAS '13. ISBN 978-1-4503-2025-2. Available from DOI: `10.1145/2513591.2513655`.

95. CURRY, E.; HASAN, S.; O'RIAIN, S. Enterprise energy management using a linked dataspace for Energy Intelligence. In: *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*. 2012, pp. 1–6.

96. CURRY, Edward; ODONNELL, James; CORRY, Edward; HASAN, Souleiman; KEANE, Marcus; ORIAIN, Seán. Linking building data in the cloud: Integrating cross-domain building data using linked data. *Advanced Engineering Informatics* [online]. 2013, vol. 27, no. 2, pp. 206–219 [visited on 2015-05-30]. ISSN 1474-0346. Available from DOI: `10.1016/j.aei.2012.10.003`.

97. HASAN, Souleiman; CURRY, Edward; BANDUK, Mauricio; O'RIAIN, Sean. Toward Situation Awareness for the Semantic Sensor Web: Complex Event Processing with Dynamic Linked Data Enrichment. In: *Proceedings of the 4th International Conference on Semantic Sensor Networks - Volume 839*. Bonn, Germany: CEUR-WS.org, 2011, pp. 69–82. SSN'11. Available also from: `http://dl.acm.org/citation.cfm?id=2887659.2887665`.

98. GAO, Jingkun; PLOENNIGS, Joern; BERGES, Mario. A Data-driven Meta-data Inference Framework for Building Automation Systems. In: *Proceedings of the 2Nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. Seoul, South Korea: ACM, 2015, pp. 23–32. BuildSys '15. ISBN 978-1-4503-3981-0. Available from DOI: `10.1145/2821650.2821670`.

99. DIBOWSKI, Henrik; KABITZSCH, Klaus. Ontology-based Device Descriptions and Device Repository for Building Automation Devices. *EURASIP J. Embedded Syst.* [online]. 2011, vol. 2011, pp. 3:1–3:17 [visited on 2015-09-15]. ISSN 1687-3955. Available from DOI: `10.1155/2011/623461`.

100. PLOENNIGS, J.; HENSEL, B.; DIBOWSKI, H.; KABITZSCH, K. BASont - A modular, adaptive building automation system ontology. In: *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*. 2012, pp. 4827–4833. Available from DOI: `10.1109/IECON.2012.6389583`.

101. RUNDE, S.; DIBOWSKI, H.; FAY, A.; KABITZSCH, K. A semantic Requirement Ontology for the engineering of building automation systems by means of OWL. In: *2009 IEEE Conference on Emerging Technologies Factory Automation*. 2009, pp. 1–8. ISSN 1946-0740. Available from DOI: `10.1109/ETFA.2009.5346991`.

102. DIBOWSKI, Henrik; KABITZSCH, Klaus. Ontology-Based Device Descriptions and Device Repository for Building Automation Devices. *EURASIP Journal on Embedded Systems*. 2010, vol. 2011, no. 1, pp. 623461. ISSN 1687-3963. Available from DOI: `10.1155/2011/623461`.

103. RUNDE, S.; FAY, A. Software Support for Building Automation Requirements Engineering – An Application of Semantic Web Technologies in Automation. *IEEE Transactions on Industrial Informatics*. 2011, vol. 7, no. 4, pp. 723–730. ISSN 1551-3203. Available from DOI: `10.1109/TII.2011.2166784`.

104. BUTZIN, B.; GOLATOWSKI, F.; NIEDERMEIER, C.; VICARI, N.; WUCHNER, E. A model based development approach for building automation systems. In: *2014 IEEE Emerging Technology and Factory Automation (ETFA)*. 2014, pp. 1–6. Available from DOI: `10.1109/ETFA.2014.7005365`.

105. VICARI, N.; WUCHNER, E.; BRÖRING, A.; NIEDERMEIER, C. Engineering and operation made easy - a semantics and service oriented approach to building automation. In: *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*. 2015, pp. 1–8. ISSN 1946-0740. Available from DOI: `10.1109/ETFA.2015.7301657`.

106. HAN, S. N.; LEE, G. M.; CRESPI, N. Semantic Context-Aware Service Composition for Building Automation System. *IEEE Transactions on Industrial Informatics*. 2014, vol. 10, no. 1, pp. 752–761. ISSN 1551-3203. Available from DOI: `10.1109/TII.2013.2252356`.

107. JUNG, M.; WEIDINGER, J.; KASTNER, W.; OLIVIERI, A. Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture. In: *2013 27th International Conference on Advanced Information Networking and Applications Workshops*. 2013, pp. 1361–1367. Available from DOI: `10.1109/WAINA.2013.200`.

108. RUTA, M.; SCIOSCIA, F.; SCIASCIO, E. Di; LOSETO, G. Semantic-Based Enhancement of ISO/IEC 14543-3 EIB/KNX Standard for Building Automation. *IEEE Transactions on Industrial Informatics*. 2011, vol. 7, no. 4, pp. 731–739. ISSN 1551-3203. Available from DOI: `10.1109/TII.2011.2166792`.

109. BOVET, G.; HENNEBERT, J. Distributed Semantic Discovery for Web-of-Things Enabled Smart Buildings. In: *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. 2014, pp. 1–5. Available from DOI: `10.1109/NTMS.2014.6814015`.

110. BOVET, Gérôme; RIDI, Antonio; HENNEBERT, Jean. Toward Web Enhanced Building Automation Systems. In: BESSIS, Nik; DOBRE, Ciprian (eds.). *Big Data and Internet of Things: A Roadmap for Smart Environments* [online]. Cham: Springer International Publishing, 2014, vol. 546, pp. 259–283 [visited on 2015-05-16]. ISBN 978-3-319-05028-7 978-3-319-05029-4. Available from: `http://link.springer.com/10.1007/978-3-319-05029-4_11`.

111. SCHACHINGER, D.; KASTNER, W. Semantics for smart control of building automation. In: *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*. 2016, pp. 1073–1078. Available from DOI: `10.1109/ISIE.2016.7745041`.

112. REINISCH, C.; GRANZER, W.; PRAUS, F.; KASTNER, W. Integration of heterogeneous building automation systems using ontologies. In: *34th Annual Conference of IEEE Industrial Electronics, 2008. IECON 2008*. 2008, pp. 2736–2741. Available from DOI: `10.1109/IECON.2008.4758391`.

113. CAFFAREL, Jaime; JIE, Song; OLLOQUI, Jorge; MARTÍNEZ, Rocío; SANTAMARÍA, Asunción. Implementation of a Building Automation System Based on Semantic Modeling. *Journal of Universal Computer Science*. 2013, vol. 19, no. 17, pp. 2543–2558. Available also from: `http://jucs.org/jucs_19_17/implementation_of_a_building/jucs_19_17_2543_2558_caffarel.pdf`.

114. BONINO, D.; CASTELLINA, E.; CORNO, F. The DOG gateway: enabling ontology-based intelligent domotic environments. *IEEE Transactions on Consumer Electronics*. 2008, vol. 54, no. 4, pp. 1656–1664. ISSN 0098-3063. Available from DOI: `10.1109/TCE.2008.4711217`.

115. ANDRUSHEVICH, A.; STAUB, M.; KISTLER, R.; KLAPPROTH, A. Towards semantic buildings: Goal-driven approach for building automation service allocation and control. In: *2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*. 2010, pp. 1–6. Available from DOI: `10.1109/ETFA.2010.5641325`.

116. MAHDAVI, Ardeshir; TAHERI, Mahnameh. An ontology for building monitoring. *Journal of Building Performance Simulation*. 2016, pp. 1–10. Available from DOI: `10.1080/19401493.2016.1243730`.

117. DIBLEY, M.J.; LI, H.; MILES, J.C.; REZGUI, Y. Towards intelligent agent based software for building related decision support. *Advanced Engineering Informatics*. 2011, vol. 25, no. 2, pp. 311–329. ISSN 1474-0346. Available from DOI: `http://doi.org/10.1016/j.aei.2010.11.002`. Information mining and retrieval in design.

118. DIBLEY, Michael; LI, Haijiang; REZGUI, Yacine; MILES, John. An integrated framework utilising software agent reasoning and ontology models for sensor based building monitoring. *Journal of Civil Engineering and Management*. 2015, vol. 21, no. 3, pp. 356–375. Available from DOI: `10.3846/13923730.2014.890645`.

119. MOUSAVI, Arash; VYATKIN, Valeriy. Energy Efficient Agent Function Block: A semantic agent approach to IEC 61499 function blocks in energy efficient building automation systems. *Automation in Construction*. 2015, vol. 54, pp. 127–142. ISSN 0926-5805. Available from DOI: `http://dx.doi.org/10.1016/j.autcon.2015.03.007`.

120. RUTA, M.; SCIOSCIA, F.; LOSETO, G.; SCIASCIO, E. Di. Semantic-Based Resource Discovery and Orchestration in Home and Building Automation: A Multi-Agent Approach. *IEEE Transactions on Industrial Informatics*. 2014, vol. 10, no. 1, pp. 730–741. ISSN 1551-3203. Available from DOI: `10.1109/TII.2013.2273433`.

121. DIBOWSKI, H.; VASS, J.; HOLUB, O.; ROJÍČEK, J. Automatic setup of fault detection algorithms in building and home automation. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016, pp. 1–6. Available from DOI: `10.1109/ETFA.2016.7733622`.

122. DIBOWSKI, H.; HOLUB, O.; ROJÍCEK, J. Knowledge-Based Fault Propagation in Building Automation Systems. In: *2016 International Conference on Systems Informatics, Modelling and Simulation (SIMS)*. 2016, pp. 124–132. Available from DOI: `10.1109/SIMS.2016.22`.

123. CAMACHO, Rui; CARREIRA, Paulo; LYNCE, Inês; RESENDES, Sílvia. An ontology-based approach to conflict resolution in Home and Building Automation Systems. *Expert Systems with Applications*. 2014, vol. 41, no. 14, pp. 6161–6173. ISSN 0957-4174. Available from DOI: `http://dx.doi.org/10.1016/j.eswa.2014.04.017`.

124. RODRIGUEZ-MURO, Mariano; KONTCHAKOV, Roman; ZAKHARYASCHEV, Michael. Ontology-based data access: Ontop of databases. In: *International Semantic Web Conference*. 2013, pp. 558–573.

125. SCHLEGEL, Daniel R; BONA, Jonathan P; ELKIN, Peter L. Comparing Small Graph Retrieval Performance for Ontology Concepts in Medical Texts. In: *VLDB Workshop on Big Graphs Online Querying*. 2015, pp. 32–44.

126. KILINTZIS, Vassilis; BEREDIMAS, Nikolaos; CHOUVARDA, Ioanna. Evaluation of the performance of open-source RDBMS and triplestores for storing medical data over a web service. In: *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. 2014, pp. 4499–4502.

A List of publications

This appendix provides comprehensive list of publications authored or co-authored by the author during his doctoral study and research. Each publication is accompanied by evaluation of significance if applicable (inclusion in Web of Science catalog, CORE ranking, impact factor,...) and level of author's contribution. The publications are ordered chronologically.

KRIKSCIUNIENE, Dalia, Tomáš PITNER, Adam KUČERA a Virgilijus SAKALAUSKAS. *Sensor Network Analytics for Intelligent Facility Management*. In George A. Tsihrintzis, Maria Virvou, Toyohide Watanabe, Lakhmi C. Jain, Robert J. Howlett. Proceedings of the 6th International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS2013). Amsterdam: IOS Press, 2013. pp. 212-221, 10 p. ISBN 978-1-61499-261-5. doi:10.3233/978-1-61499-262-2-212.

- Contribution: 30%
- Significance: Indexed in Web of Science Core Collection

KUČERA, Adam, Petr GLOS a Tomáš PITNER. *Fault detection in building management system networks*. In Slanina, Zdeněk. 12th IFAC Conference on Programmable Devices and Embedded Systems, PDeS 2013. International Federation of Automatic Control, 2013. pp. 416-421, 6 p. ISBN 978-3-902823-53-3. doi:10.3182/20130925-3-CZ-3023.00027.

- Contribution: 90%
- Significance: Indexed in Scopus

KUČERA, Adam a Tomáš PITNER. *Intelligent Facility Management for Sustainability and Risk Management*. In Hřebíček, J.; Schimak, G.; Kubásek, M.; Rizzoli, A.E.. Environmental Software Systems. Fostering Information Sharing. Berlin Heidelberg: Springer, 2013. pp. 608-617, 10 p. ISBN 978-3-642-41150-2. doi:10.1007/978-3-642-41151-9_57.

- Contribution: 80%
- Significance: Indexed in Web of Science Core Collection

KRIKSCIUNIENE, Dalia, Tomáš PITNER, Adam KUČERA a Virgilijus SAKALAUSKAS. *Data Analysis in the Intelligent Building Environment*. International Journal of Computer Science & Applications, Kolhapur, India: Technomathematics Research Foundation, 2014, Volume 11, Issue 1, pp. 1-17. ISSN 0972-9038.

- Contribution: 40%
- Significance: Indexed in Scopus

ASFAND-E-YAR, Muhammad, Adam KUČERA a Tomáš PITNER. *Semantic Web Technology for Building Information Model*. In Andreas Holzinger, Thérèse Libourel, Leszek A. Maciaszek, Stephen J. Mellor. Proceedings of the 9th International Conference on Software Engineering and Applications, Vienna, Austria. SciTePress, 2014. pp. 109-116, 8 p. ISBN 978-989-758-036-9. doi:10.5220/0004999201090116.

- Contribution: 45%
- Significance: CORE 2014 rank B

ASFAND-E-YAR, Muhammad, Adam KUČERA a Tomáš PITNER. *Smart buildings: Semantic web technology for building information model and building management system*. In 2014 International Conference on Data and Software Engineering (ICODSE). IEEE, 2014. pp. 1-6, 6 p. ISBN 978-1-4799-7996-7. doi:10.1109/ICODSE.2014.7062671.

- Contribution: 45%
- Significance: Indexed in Web of Science Core Collection

KUČERA, Adam a Tomáš PITNER. *Towards Flexible Intelligent Building Data Analysis*. In Jiří Hřebíček, Jan Ministr, Tomáš Pitner. 11th Summer School of Applied Informatics Proceedings. 2014. pp. 77-83, 7 p. ISBN 978-80-85763-87-4.

- Contribution: 90%

KUČERA, Adam; PITNER, Tomáš. *Semantic BMS: Ontology for Analysis of Building Automation Systems Data*. In: DOCEIS 2016: Technological Innovation for Cyber-Physical Systems. 2016, vol.470/2016. IFIP Advances in Information and Communication Technology. pp. 46-53, 8 p., ISBN 978-3-319-31164-7. ISSN 1868-4238.

- Contribution: 90%

- Significance: Indexed in Web of Science Core Collection

KUČERA, Adam; PITNER, Tomáš. *Semantic BMS: Ontology for Analysis of Building Operation Efficiency*. In: Conference proceedings of the International Symposium on Environmental Software Systems (ISESS) 2017 proceedings. To be published.
- Contribution: 90%
- Significance: Not published yet (expected indexing in WOS)

A.1 Publications for industry practitioners – in Czech

Additionally, the author published two articles in Czech industry journal on building automation systems.

KUČERA, Adam. *BMS na Masarykově univerzitě*. TZB-info, Praha: Topinfo, 2014, Vol. 16, Issue 16, pp. 1-4. ISSN 1801-4399.

KUČERA, Adam. *Zajištění spolehlivosti BMS*. TZB-info, Praha: Topinfo, 2014, Vol. 16, Issue 27, pp. 1-4. ISSN 1801-4399.