



Applying Knowledge Graphs as Integrated Semantic Information Model for the Computerized Engineering of Building Automation Systems

Henrik Dibowski^(✉)  and Francesco Massa Gray 

Corporate Research, Robert Bosch GmbH, 71272 Renningen, Germany
henrik.dibowski@de.bosch.com

Abstract. During the life cycle of a smart building, an extensive amount of heterogeneous information is required to plan, construct, operate and maintain the building and its technical systems. Traditionally, there is an information gap between the different phases and stakeholders, leading to information being exchanged, processed and stored in a variety of mostly human-readable documents. This paper shows how a knowledge graph can be established as integrated information model that can provide the required information for all phases in a machine-interpretable way. The knowledge graph describes and connects all relevant information, which allows combining and applying it in a holistic way. This makes the knowledge graph a key enabler for a variety of advanced, computerized engineering tasks, ranging from the planning and design phases over the commissioning and the operation of a building. The computerized engineering of building automation systems (BAS) with an advanced software tool chain is presented as such a use case in more detail. The knowledge graph is based on standard semantic web technologies and builds on existing ontologies, such as the Brick and QUDT ontologies, with various novel extensions presented in this paper. Special attention is given to the rich semantic definition of the entities, such as the equipment and the typically thousands of datapoints in a BAS, which can be achieved as a combination of contextual modeling and semantic tagging.

Keywords: Building automation system · Knowledge graph · Information model · Semantic definition · Semantic tagging · Building controls · Analytics

1 Introduction

Several trends in the building domain, such as the increasing ubiquitousness and interconnectivity of IoT devices [1], an energy or cost efficient operation guaranteed through the well-orchestrated control of heterogeneous technical equipment and systems [2] and the adaptation of the indoor climate to individual needs [3], have led to the concept of “smart buildings”. A central part of a smart building is the building automation system (BAS), which can be understood as the combination of software and hardware required

to operate and monitor the building [4]. BAS can consist of hundreds or thousands of sensors, actuators and control functions, and a complex communication system connecting all these devices into an interoperable, functioning system.

During the life cycle of a smart building, an extensive amount of different kinds of heterogeneous information is required. Traditionally, there is an information gap between the different phases (e.g. planning, installation, operation) and stakeholders, leading to information being exchanged, processed and stored in a variety of mostly human-readable documents of different types. This leads to a large amount of time and effort required to organize and integrate the data, and it makes it very hard or impossible for computers to make use of the information.

In practice, the planning and design of building automation systems is hence a complex, predominantly manual process. BAS planners must typically extract and interpret requirements and information about the building and its technical systems from heterogeneous sources, translate this information into a general or functional plan of the system and finally design and program the complex, highly-connected BAS, including the necessary software and hardware components and the communication between them. This has several disadvantages: The manual data collection is cumbersome and error-prone, certain tasks are repetitive and time consuming, and the manual BAS setup can result in configuration and parameterization issues. Moreover, the current lack of qualified technicians in the building automation field is forecasted to increase in the next years [5].

Empowered by the strength of semantic technologies for describing information and knowledge in a holistic, consistent and machine-interpretable way, this paper presents a semantic model developed for an automated BAS engineering solution. The fundamental concept behind the solution, which is called “BIM2BA”, is to gather digital information through all planning and engineering phases and use it in order to automatically configure and parameterize the BAS, which is up to now a highly manual process that requires skilled engineers. The information can stem from various sources, including a Building Information Model (BIM), a requirement definition tool, digital product catalogues or digital parts lists. By storing all the BAS planning data using semantic technologies, information silos are avoided and errors and inconsistencies can be automatically detected by employing reasoning on the resulting knowledge graph.

This paper specifically focuses on the semantic model and knowledge graph of the BIM2BA solution. For this purpose we built upon existing ontologies, such as Brick [6], and extended them where necessary. Moreover, we explain why we consider semantic tagging to be an enabler for automating complex engineering tasks in the building domain.

2 The BIM2BA Solution and Workflow

The overall objective of the BIM2BA solution is to plan and generate working BAS control networks automatically with only minimal manual effort. The development of the BIM2BA solution was triggered by the current digitalization trend in the building industry. An embodiment of this trend can be found in Building Information Modeling (BIM) [7], a digital planning process for buildings that has been gaining strength in the

past decades. At the core of the BIM process lies the Building Information Model¹, which is typically understood as a machine-readable, digital representation of the building and its subsystems. By ingesting data stemming from a BIM, BIM2BA extracts, interprets and stores the relevant architectural and engineering data automatically.

An overview of BIM2BA’s workflow and software architecture is shown in Fig. 1. BIM2BA consists of a RESTful backend with four microservices, which represent and implement the four phases of the BIM2BA workflow. The knowledge graph is the core element of the BIM2BA backend and stores all relevant information. A browser-based frontend acts as the interface for the user, typically planners and system integrators. On the technical infrastructure level, the backend is powered by a Jena Fuseki triple store, the Jena ontology API and the TopBraid SHACL API.

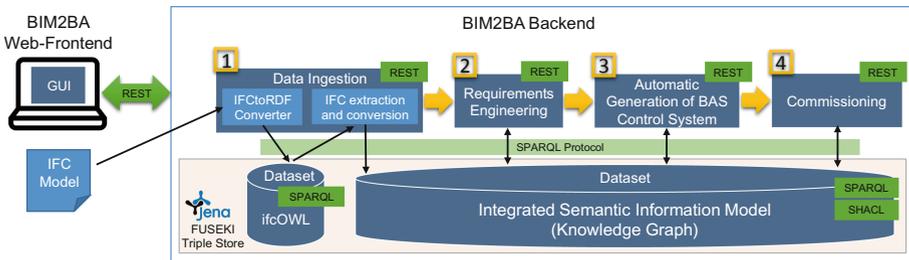


Fig. 1. Workflow and software architecture of the BIM2BA solution

In the following, we explain the four phases of BIM2BA for the automatic generation of a BAS.

1) Data Ingestion

The BIM2BA workflow starts with the ingestion of a BIM file in the IFC format [8]. The IFC file must contain the building geometry and the HVAC system topology, including information regarding the characteristics of the HVAC components.

The IFC file is automatically converted to ifcOWL [9] by using the IFCtoRDF converter software [10], with the resulting triples stored in a separate dataset in the triple store. Since ifcOWL is an automatic one-to-one translation of the complicated IFC EXPRESS schema into RDF, the resulting structures stay complicated and are not straightforward to navigate and search. To overcome these issues and drastically reduce the model complexity (IFC models describe the entire geometry), our solution performs a further model transformation from ifcOWL to our semantic model via a set of transformation rules encoded as SPARQL update queries. This results in a much leaner semantic model, which is optimized for following causalities and for performing semantic search and reasoning.

¹ Also abbreviated as “BIM”.

Additionally, information from other sources can be ingested and stored in the knowledge graph via dedicated ingestion pipelines (not shown in Fig. 1), along with information manually entered with dedicated tools.

2) Requirements Engineering

In addition to the technical data contained in the BIM, the BA planner must define the functional and non-functional requirements for the BAS to be built. For this purpose, he or she can use the requirements engineering web-frontend, which allows the context-sensitive definition of requirements, given all known information about the building and HVAC system. The requirements are stored in the knowledge graph and checked for consistency and completeness with SHACL. An example of the requirements engineering web-frontend can be seen in Fig. 2.

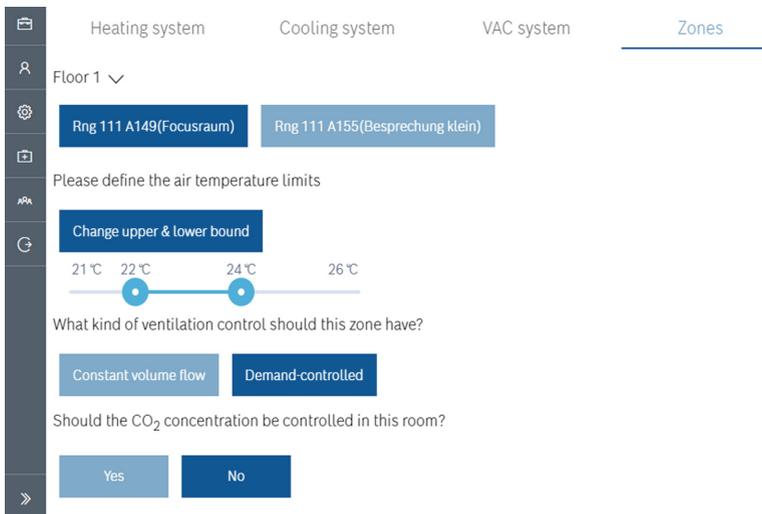


Fig. 2. Example of the requirements engineering web-frontend in the BIM2BA solution

In the simplified example presented there, the requirements comprise the following: the type of control used in each control zone; the parameters of the controllers, such as the upper and lower temperature bounds or the maximum CO₂ concentration; the desired operation schedule of the HVAC system.

3) Automatic Generation of BAS Control System

Based on the information about the building, HVAC system and requirements available in the knowledge graph, the control system of the BAS is automatically generated by the BIM2BA solution and a description of it is added to the knowledge graph. At first, the necessary control functions for each control zone are determined. The control functions are then configured for the specific HVAC system. This includes setting up all required connections for the communication between the control functions and the sensors and actuators. Additionally, the controller's parameters and the actuation ranges

for the different actuators have to be defined. This includes the inclusion of parameters defined by the user during the requirements engineering, e.g. the desired temperature bounds, and the post-processing and transformation of specific user requirements into control parameters. The latter requires transformations and calculations, which can be done directly on the knowledge graph via SHACL rules (see Sect. 5).

4) Commissioning

In a final commissioning step, the generated BAS control system can be deployed to the HVAC system. Tridium Niagara [11] is the currently supported target system.

In summary, the BIM2BA solution allows the user to create a BAS software without requiring advanced knowledge about the inner workings of the control functions nor the target system. A central part of BIM2BA is its rich semantic model, which is described in more detail in Sect. 4.

3 Related Work

Information modeling and ontologies in the building automation domain have been widely addressed in the past years, which resulted in a variety of approaches. Good surveys of that field have already been provided in [12] and [13]. These works show a clear trend in moving from conventional, often proprietary information models, mostly based on text files or XML dialects, to more expressive approaches based on standardized semantic technologies and ontologies.

Apart from solutions based on semantic technologies, there is some recent work that uses OPC/UA as information model for BAS [14], or that stays on a high, technology-independent level [15].

Since we are fully convinced about the strength of semantic technologies, we chose them as technology for realizing the BIM2BA semantic model. Instead of developing yet another ontology or information model from scratch, we analyzed the existing solutions from the surveys [12] and [13] and additionally the Linked Building Data (LBD) ontologies² for suitability. Brick [6], the Haystack 4.0 ontology³ and the LBD ontologies appeared content-wise to be the best candidates with respect to completeness and conciseness. Further evaluation criteria we applied were accuracy, clarity, adaptability and the user community behind. We finally selected the Brick ontologies as the most suitable approach. Brick allows for modeling a building from a structural point of view (topological building), and of its BAS components and datapoints, which is a good fit to what we need to model. Furthermore, there is a broad community of both industrial and academic contributors and supporters behind Brick.

The Haystack 4.0 ontology on the contrary was not as convincing, since, due to its claim of ensuring full backward compatibility to older Haystack versions, inherent issues from Haystack from a sound and good modeling perspective kept on existing in it. Such negative aspects are the strongly typed relations, poorly defined tags and the low expressivity. Nevertheless, the Haystack-typical tagging mechanism, which found

² <https://w3c-lbd-cg.github.io/lbd/UseCasesAndRequirements/>.

³ <https://project-haystack.dev/doc/docHaystack/Rdf>.

its way into Brick as well, inspired us for our semantic modeling approach as a simple but powerful way of defining semantics and enabling semantic search (see Sect. 4.6). Despite the good intentions behind the emerging LBD ontology and initiative, it was not ready for being used at the time, but meanwhile it is worth reconsidering it.

We made several extensions to Brick for customizing it to the needs of the BIM2BA solution (see Sect. 4). The main differentiation of the BIM2BA semantic model, described in this paper, and other existing solutions is the intended coverage of the whole building life cycle in a holistic knowledge graph, including requirements, and the way how a rich semantic definition of BAS entities is achieved by a combination of contextual modeling and semantic tagging. This will be explained in the next section.

4 Integrated Semantic Information Model

The semantic model of the BIM2BA solution acts as an integrated information model that can cover all life cycle phases of a building, making the information combinable and applicable in a holistic way. As the information backbone, it is positioned at the core of the BIM2BA software architecture, as can be seen in Fig. 1. It can contain all the required information, including the building geometry, interior architectural layout, technical details of the HVAC system, the controllers and their functional description, the datapoints, parameters and the BAS requirements etc.

As was mentioned in the last chapter, we chose the Brick ontologies as foundation for the semantic model. We made several extensions to Brick to customize it to the needs of the BIM2BA solution. For this purpose, we used the professional ontology IDE TopBraid Composer. The customizations extend the coverage of Brick from the operation phase of buildings towards the planning and engineering phase, by adding capabilities for modeling and storing requirements for the BAS to be built, and for expressing the functionality of the control network.

The following sections describe the semantic model of BIM2BA in detail, from its structure over extensions made to Brick up to different use cases that were addressed. The ontology examples will show excerpts from a semantic model of a building at the Bosch site in Renningen, Germany, which served as one of the demonstrator buildings for the BIM2BA solution.

4.1 Ontology Layer Architecture

Semantic technologies provide powerful, flexible means of reusing and extending existing vocabularies. Ontologies can be (re-)used by simply importing them into a model, and they can be extended by subclassing, by defining (sub-)properties and by enriching them with further axioms and constraints. As mentioned in the previous section, the Brick ontologies were one starting point for the BIM2BA semantic model.

Apart from Brick, two more groups of ontologies were (re-)used: the QUDT (Quantities, Units, Dimensions and Types) ontologies⁴ and the Open PHACTS Units

⁴ <http://www.qudt.org/release2/qudt-catalog.html>.

ontology^{5,6} (OPS). QUDT defines an extensive list of units, quantities, dimensions and types of more than a thousand in number, and is a de-facto standard for expressing units and currencies. Still, not all units are defined in QUDT, which is why OPS is also employed. OPS uses the QUDT ontology vocabulary and defines additional units, for example parts per million (ppm), which was needed by BIM2BA to define CO₂ concentrations.

The integration and extension of these ontologies resulted in a layered ontology model, which is shown in Fig. 3. Arrows from one ontology block (the source) to another (the target) represent an import relationship between both (groups of) ontologies, i.e. all triples defined in the target model are imported into the source model. Those arrows labeled with “uses” import and leave the models as they are, whereas arrows labeled with “extends” import, extend and hence enrich the models. Figure 3 shows also the namespace prefixes used in all following examples for the respective ontologies.

The Brick Extensions ontology presented in this paper combines all aforementioned ontologies and customizes the Brick ontologies in order to address aspects that were not adequately supported, but needed (see Sect. 4.5).

The bottom layer comprises the building instance ontologies, each of which models and represents a specific building of the real world, and each of which uses a part of the ontology vocabulary defined in the upper ontology layers. Some insights on these vocabularies and several example knowledge graphs will be explained in the following.

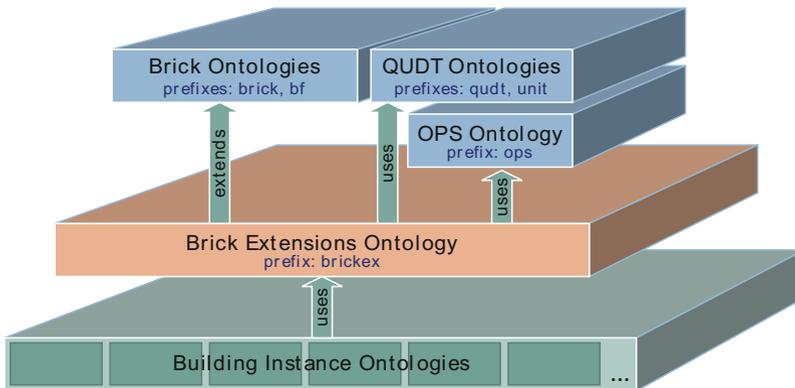


Fig. 3. BIM2BA ontology layer architecture

4.2 Use Case: Modeling of the Hierarchical Building Structure

Brick defines the necessary concepts and object properties to define a structural building model. This is shown in Fig. 4 for the Bosch site in Renningen⁷. There, the concepts `brick:Building`, `brick:Floor`, `brick:Room` and `brick:HVAC_Zone`

⁵ <http://www.openphacts.org/specs/2013/WD-units-20130913/>.

⁶ <https://github.com/openphacts/jqudt/blob/master/src/main/resources/onto/ops.ttl>.

⁷ This and all following figures were created with TopBraid Composer.

are used to model the building Rng111 with its first floor and two rooms in it. The `bf:hasPart` object property defines the hierarchical containment structure, i.e. building contains floor, floor contains rooms etc. Properties of the rooms, such as area and volume, are modeled as specific static types of properties, which is one of the extensions made to Brick (see Sect. 4.5). They are attached to the rooms via the Brick object property `bf:hasPoint`.

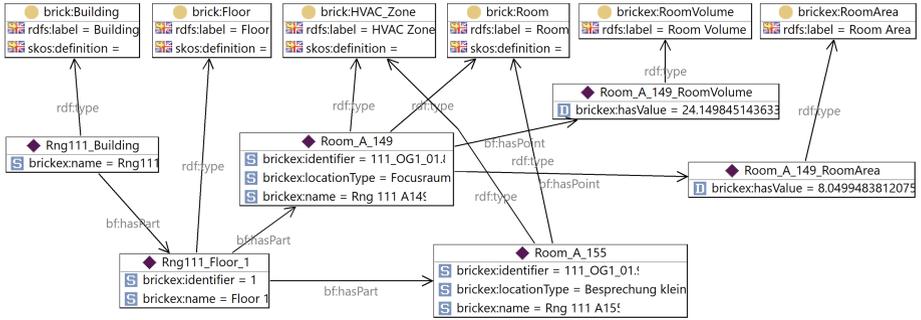


Fig. 4. Structural building model example with some static properties (See footnote 7)

4.3 Use Case: Modeling of the HVAC System and Energy Flows

Brick furthermore enables the modeling of HVAC plants and equipment and their up- and downstream relationships. Figure 5 shows that for a section of the BAS in building Rng111: An air handling unit supplies two downstream VAV⁸ boxes, which supply the rooms they are located in. The supply relationships are modeled with the Brick object property `bf:feeds`, which represents the supply of material and energy (here: heated or cooled air) from an upstream to a downstream plant or building element.

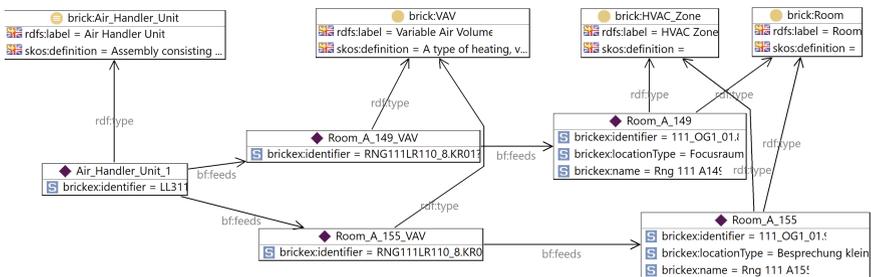


Fig. 5. Model of the material and energy flow from plants to zones

4.4 Use Case: Modeling of Datapoints

Datapoints, such as inputs and outputs of sensors, controllers or actuators in the building, can be represented as instances of the various specializing subclasses of

⁸ Variable Air Volume, a type of ventilating and air-conditioning system.

brick:Point. They are associated with an HVAC plant, equipment or building element via bf:hasPoint object property. Figure 6 shows an example of three modeled datapoints that are associated with a room. Two of them are sensor datapoints of type brick:Return_Air_Temperature_Sensor and brick:Return_Air_CO2_Sensor. They have a BACnet address and identifier assigned via two new datatype properties brickex:bacnetAddress and brickex:identifier. The third datapoint is a parameter of the room itself and defines its maximum possible supply airflow as 240.5 m³/h. All three datapoints have a unit of measurement from either the QUDT ontologies or the OPS ontology attached (see Sect. 4.1).

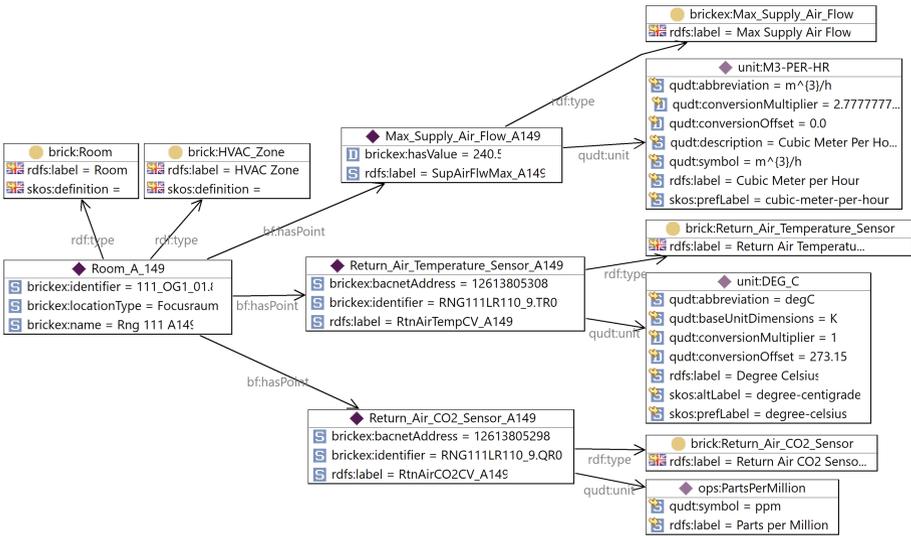


Fig. 6. Model of datapoints and their units, values and BACnet addresses

4.5 Customizations and Extensions Made to Brick

The Brick ontologies were one starting point for the semantic model for buildings, as described in the previous section. Brick was customized by our team in order to address aspects that were not adequately supported. The customization was done by subclassing and by defining entirely new class trees and properties.

Static Properties

An important concept that was missing in Brick is the concept of static properties. Unlike time series based datapoints, such as sensor values, commands or alarms (which are widely addressed by Brick), static properties do not change over time. Therefore we added a new subclass **brickex:StaticValue** to the **brick:Point** class, as can be seen in Fig. 7. By defining new subclasses of **brickex:StaticValue**, such as **brickex:RoomArea**, **brickex:RoomVolume**, new types of static properties can

now be defined and used, and they are semantically distinguishable from conventional datapoints. The usage of these added classes was shown already in Fig. 4.

Datatype Properties

Brick itself does not define any datatype properties. We had to import and define a couple of datatype properties for modeling required attributes, such as names, identifier, BACnet addresses etc.

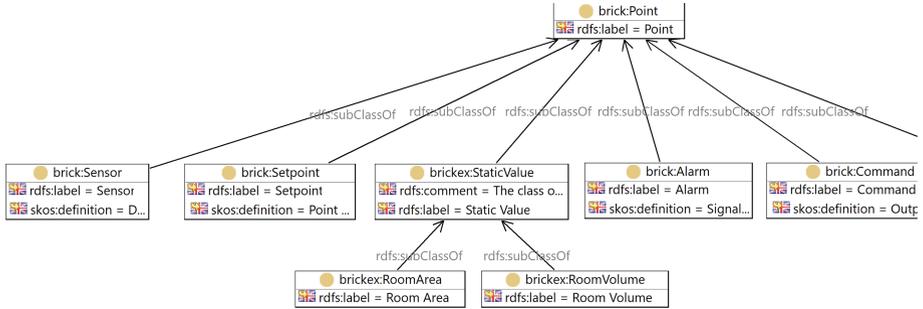


Fig. 7. New concept brickex:StaticValue and subclasses for modeling static properties

Control Functions

Brick is also missing concepts for control functions, i.e. classes for expressing the functionality of field and automation devices. We added such concepts by introducing a new class brickex:ControlFunction and a tree of subclasses underneath it, as displayed in Fig. 8. This class hierarchy is not exhaustive, but extensible as needed.

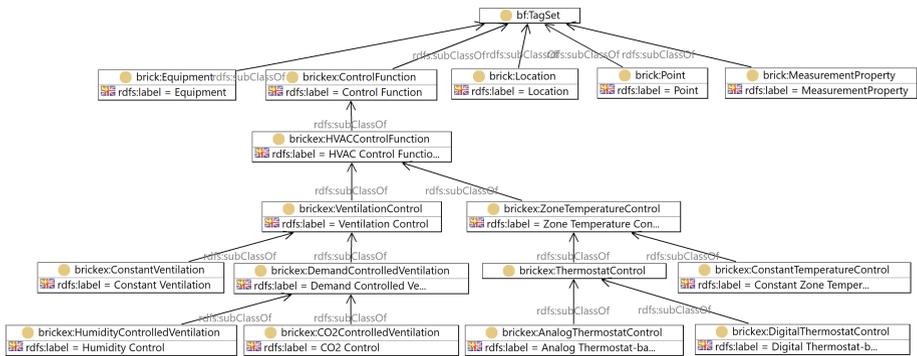


Fig. 8. New class hierarchy for modeling control functions (extract)

New Datapoint Subclasses and Tags

Additional extensions comprise new and more specific datapoint subclasses (brick:Point) for some missing types of datapoints, and new tags (e.g. Area, Constant, StaticValue) needed for tagging some of the new classes.

Requirements Model

Furthermore, a comprehensive requirements model, shown in Fig. 9, was developed. It comprises classes and properties for modeling requirements of a BAS that is to be planned and built. A requirement (class `brickex:Requirement`) can define a control function as required feature to be implemented (object property `brickex:requiredFeature`). Optionally, it can define one or more parameter values relevant for the control function to be realized. Related requirements can be bundled into requirement sets (class `brickex:RequirementSet`), and each requirement set can be attached to either a building element or an HVAC plant/equipment.

The requirements model allows for expressing requirements such as the following: The room “Rng111 A155” should be equipped with a CO₂- and humidity-controlled ventilation with an upper CO₂ concentration limit of 800 ppm. The room temperature should be controlled by a digital thermostat, with the lower and upper temperature bounds being 18 and 23 °C, respectively.

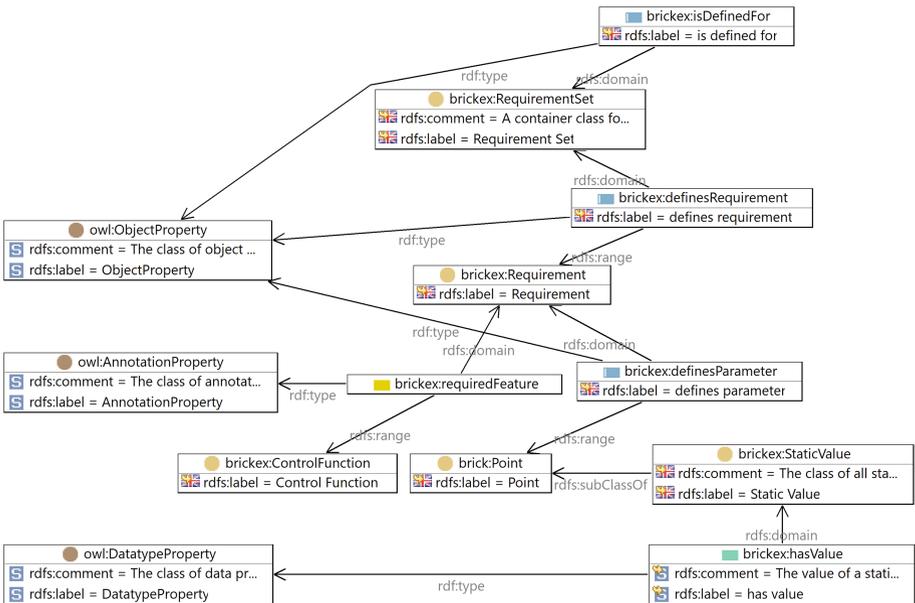


Fig. 9. Requirements meta-model

Brick Criticism and Recommendations

We recommend further improvements to Brick, especially regarding its usability and human interpretability. What is completely missing in Brick are human readable labels (`rdfs:label`) and descriptions (`rdfs:comment`) of the defined classes and object properties, which compromises clarity. Furthermore, there are several occurrences of duplicate class definitions that should be resolved.

Brick at its current state abstracts from the specific hardware of devices and sensors, as well as from their specific software functionality, as it mainly focuses on the modeling

of the HVAC equipment and their datapoints. If that was required, such a device layer and relevant properties would need to be added to Brick.

4.6 Machine-Interpretable Semantic Definition

Semantics is the philosophical and linguistic study of meaning in language, be it natural language or computational languages, such as programming languages and formal logics. It is concerned with the meaning of words or symbols and what they stand for in reality. In this section, we want to focus on the semantics of human readable words that are stored and used in a computer system as names or labels for real-world entities.

Consider the following example: The strings “RtnAirCO2CV_A149”, “SupAirFlwMax_A149” and “RtnAirTempCV_A149” are examples of datapoint names (`rdfs:labels` in Fig. 6). Such natural-language based texts carry some implicit semantics, which is interpretable by a person reading it, albeit this much depends on the person’s background and contextual knowledge he or she has. The meaning of such labels is however not (directly) interpretable for machines, as it requires natural language processing and knowledge about the domain and context.

Ontologies and knowledge graphs are an adequate means to model knowledge and context in a machine-interpretable way, and they can define the semantics of symbols. One way of capturing semantics is by applying rich logical formalisms that describe entities and their semantics formally, such as with description logics. However, this can become extremely complex, and may still not be able to capture the complete semantics. A more manageable, yet powerful alternative way of defining semantics is contextual modeling by means of a rich, interconnected knowledge graph. All relevant entities are to be modeled therein with all relationships and property values that are relevant for understanding their meaning, in the extent that is required for the particular use cases.

Yet, some semantic aspects of certain entities cannot be fully captured by modeling the context alone. The meaning of the various types of datapoints, for example, be it the current value of a return air temperature or a chilled water supply temperature setpoint etc., cannot be adequately expressed by modeling the surroundings of the datapoints, such as the equipment they belong to. In addition to the contextual modeling, the solution for capturing the particular semantics of such entities is semantic tagging.

Semantic tagging is a concept that first appeared in the BAS domain in Project Haystack, and it is also supported by Brick. Semantic tags are the underlying, elementary building blocks that ideally cannot be further split down into smaller semantic units. In Brick, semantic tags are defined as direct subclasses of `bf:Tag` class, and there are 313 of them predefined, from A like `Acceleration` to Z like `Zone`. Based on the requirements of the BIM2BA use case, we added several new tags to Brick.

Figure 10 shows the semantic tagging approach of Brick on the three datapoints from Fig. 6. While Fig. 6 shows the surrounding knowledge graph, i.e. contextual knowledge, of the datapoints, Fig. 10 shows their semantic tags. The tags are attached via `bf:usesTag` annotation properties to the classes of the datapoints. By that, instantiating a class means that all the tags of the class are applicable to their instances. The combination of all tags of an entity then describes the semantics of the entity. The semantics of the datapoint “RtnAirTempCV_A149” (instance `Room_A_149_Return_Air_Temperature_Sensor`), for example, is Return

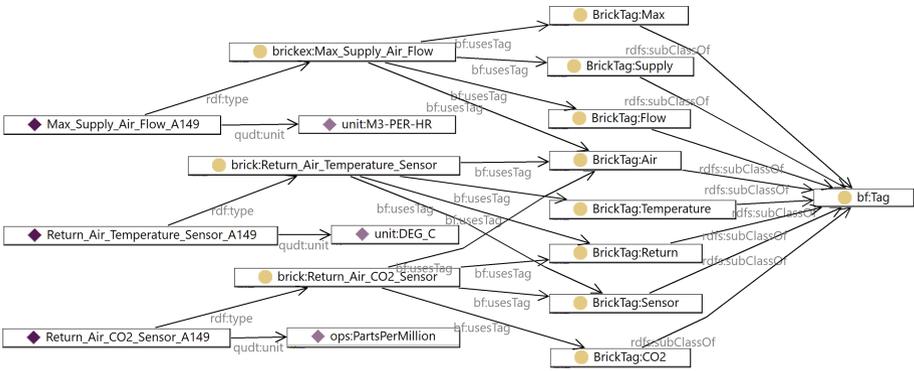


Fig. 10. Semantic definition of datatypes with semantic tags and units

AND Air AND Temperature AND Sensor. Based on the tags, the datapoint is a sensor datapoint that measures a return air temperature. Semantic tagging is applied in the same way to semantically describe HVAC plants and equipment, or any other type of entities.

Semantic tagging is superior to the conventional definition of (large) class hierarchies, whenever the classes are defined by combining atomic concepts to more complex ones. The datatypes of BAS are such an example, where several tags from multiple dimensions are combined to express the overall semantics of datapoints (see examples from Fig. 10). Expressing the entirety of possible datapoint types in a class hierarchy would result in a highly complex, and very likely never complete taxonomy, along with thousands of `rdfs:subClassOf` relations and multiple inheritance. Semantic tagging however does not require to predefine all possible classes, i.e. combinations of tags (despite Brick does so, which is in our opinion the wrong approach). Instead, a set of tags should be in place that allows for selecting and combining the required ones.

We herewith propose, as an additional improvement of Brick and in general, to organize the tags into orthogonal tag categories (i.e. dimensions), such as measurement (tags “temperature”, “pressure”, “mass flow” etc.), material (tags “air”, “water” etc.), control (tags “current value”, “setpoint”, “command” etc.), plant (tags “VAV”, “air handler unit” etc.) and so on. Consistency rules can then be imposed on the tags, such as the rule that an entity can be tagged with maximum one tag per category, or rules that constrain the combination of specific tags from different dimensions etc.

Besides the tags and the contextual model, the unit of measurement adds another dimension to the semantic definition of a datapoint. Physical units such as degree centigrade (°C) and cubic meter per hour (m³/h), or pseudo-units such as part per million (ppm) comprise certain semantics on their own, namely that it is a temperature, a volume flow or a concentration of some substance.

5 Semantic Model as Key-Enabler for Automating Engineering Tasks and Advanced Features

A rich semantic model of a BAS, forming an interconnected knowledge graph, can provide many advantages. The knowledge graph is a key enabler for the computerized automation of a variety of engineering tasks that previously could only be realized by qualified engineers. In the following, different use cases are explained.

Contextual modeling and semantic tagging enables semantic search, i.e. the search for entities based on their semantic definition, instead of a primitive string matching. It is straightforward to write SPARQL queries that search for all entities related to air temperature by defining a SPARQL graph pattern that searches for all entities that have both the tag `Air` and `Temperature` attached. This simple but powerful mechanism allows for searching for all temperature-related setpoints, for all hot-water-related plants and equipment, for all datapoints that are not alarms and many more scenarios.

Semantic search can simplify and automate the task of finding the equipment and datapoints of interest in a building. Currently it is a complicated task to find the required datapoints amongst couple of thousands in a building by using a string search on their names. The hits are often wrong (false positives) or many datapoints are not matched by the search (incomplete results), so that an engineer has to try different terms to improve the search results, but finally still has to go through a list of datapoint names, interpret their meaning and make the right selection. Semantic search dramatically improves that situation by returning exact matches and complete search results, at the push of a button. It enables software and algorithms to take over this task of finding and selecting the right equipment and datapoints, and by that releases the engineers from this repetitive and laborious task. That is a key enabler for several advanced features, such as building management dashboards that are composed and visualized automatically, or building analytics (e.g. fault detection, predictive maintenance) that are self-enabled, i.e. get configured and commissioned completely automatically [16]. Even virtual sensors that compute unavailable measurements virtually from other available data, can be created automatically from such a knowledge graph [17].

In the BIM2BA use case, the knowledge graph is the key enabler for automating the planning and engineering of BAS. The knowledge graph provides all required information and makes it accessible and retrievable within one repository, with one query language (SPARQL). Information is no longer kept in separated silos, without a semantic definition, but it is totally integrated, connected and has a rich semantics. That enables computers to query and process the information, make sense out of it and automate important engineering tasks. By storing the BAS requirements in the same knowledge graph, with the same concepts, requirements can be mapped directly to matching equipment, devices etc., which were described with the same ontologies. All that relieves engineers to process the requirement documents, study product catalogs and specifications and match requirements to suitable control structures and equipment.

Queries and reasoning allow for performing operations and computations directly in the knowledge graph. For a constant volume flow control to be realized for a zone, for example, knowing the desired hourly air change rate⁹ (requirement entered by the user)

⁹ The amount of time the air in a zone is completely replaced.

and the volume of the control zone (information from BIM), a SPARQL Insert query or SHACL rule can calculate and materialize the equivalent volume flow in m^3/h . It can then be used as setpoint parameter for the controller, and it also defines the required minimum volume flow of a ventilation damper to be chosen. Such computations can automatically run in the background and expand the knowledge graph by additional information, which otherwise had to be calculated and provided by engineers.

Reasoning is another key benefit of knowledge graphs. A reasoner can process axioms and rules on the knowledge graph and derive new information that enriches it. That has been applied and patented for a rule-based fault propagation and root cause analysis for BAS [18]. It is based on a set of rules (e.g. SHACL rules) that formalize the causalities of how faults can physically (via material flow) or logically (via control network) propagate in the building and affect other equipment, zones and datapoints.

6 Conclusion

In this paper we presented the semantic model developed for the BIM2BA solution, a software for the automated engineering of building automation systems (BAS). Creating a BAS is typically a highly manual task requiring the extraction and combination of information from heterogeneous sources from different phases in the building's lifecycle. By harmonizing, combining and integrating BAS information into a rich, interconnected knowledge graph, all information is made available in one repository and usable in a holistic way. This overcomes information silos and enables semantic search and reasoning over the complete set of triples.

As a basis for the semantic model we used the Brick ontologies and extended them by different aspects, such as capabilities for modeling BAS requirements and the functionality of the control network. The customizations extend the coverage of Brick from the operation phase of BAS towards the planning and engineering phase. Furthermore, we found certain issues with Brick, which we clarified and recommend to improve.

The resulting knowledge graph is a key-enabler for the automated engineering of BAS, which was realized with the BIM2BA solution, as well as for a variety of other advanced functionalities, such as automatically enabled fault detection and analytics. The basis for such advanced use cases is the rich semantic definition of entities, achieved with a combination of contextual modelling in the knowledge graph and semantic tagging. This enables the precise retrieval of datapoints and other BAS entities of interest with semantic search. Furthermore, it supports rule-based inferences on the knowledge graph (SHACL rules), such as the creation and calculations of BAS parameters, the propagation of faults, or plausibility and consistency checks.

In summary, the developed semantic model provides a universally applicable, formal vocabulary for the building automation domain. It has proven to be suitable for automating the engineering of BAS, as well as for realizing automatically enabled and advanced analytics, which can lead to a strong reduction in cost and time and to an increased energy efficiency of the buildings.

References

1. Gupta, A., Tsai, T., Rueb, D., Yamaji, M., Middleton, P.: Forecast: Internet of Things—Endpoints and Associated Services, Worldwide. Gartner Research (2017)
2. Henze, G.P., Kalz, D.E., Liu, S., Felsmann, C.: Experimental analysis of model-based predictive optimal control for active and passive building thermal storage inventor. *HVAC&R Res.* **11**(2), 189–213 (2005)
3. Feldmeier, M., Paradiso, J.A.: Personalized HVAC control system. In: *IEEE Internet of Things (IOT)*, Tokyo, Japan (2010)
4. ISO 16484-2: Building Automation and Control Systems (BACS) – Part 2: Hardware. International Organization for Standardization (2004)
5. Seefeldt, F., Rau, D., Hoch, M.: Fachkräftebedarf für die Energiewende in Gebäuden. Prognos, Study commissioned by VdZ: Forum für Energieeffizienz in der Gebäudetechnik e.V. (2018)
6. Balaji, B., et al.: Brick: towards a unified metadata schema for buildings. In: *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, Stanford, California (2016)
7. buildingSMART International. <https://www.buildingsmart.org/>. Accessed 28 Oct 2019
8. Industry Foundation Classes – An Introduction (buildingSMART). <https://technical.buildingsmart.org/standards/ifc/>. Accessed 12 March 2020
9. Beetz, J., van Leeuwen, J., de Vries, B.: IfcOWL: A Case of Transforming EXPRESS Schemas into Ontologies. *Artif. Intell. Eng. Des. Anal. Manuf.* **23**(1), 89–101 (2009)
10. Pauwels, P.: IFctoRDF. GitHub. <https://github.com/pipauwel/IFctoRDF>. Accessed 27 Nov 2019
11. Tridium Inc. <https://www.tridium.com/en/products-services>. Accessed 10 Dec 2019
12. Butzin, B., Golasowski, F., Timmermann, D.: A survey on information modeling and ontologies in building automation. In: *43rd Annual Conference of the IEEE Industrial Electronics Society (IECON 2017)*, Beijing, China (2017)
13. Dibowski, H., Ploennigs, P., Wollschlaeger, M.: Semantic device and system modeling for automation systems and sensor networks. *IEEE Trans. Ind. Informat.* **14**(4), 1298–1311 (2018)
14. Fernbach, A., Kastner, W.: Semi-automated engineering in building automation systems and management integration. In: *26th IEEE International Symposium on Industrial Electronics (ISIE)*, Edinburgh, UK (2017)
15. Lehmann, M., Andreas, J., Mai, T. L., Kabitzsch, K.: Towards a comprehensive life cycle approach of building automation systems. In: *26th IEEE International Symposium on Industrial Electronics (ISIE)*, Edinburgh, UK (2017)
16. Dibowski, H., Vass, J., Holub, O., Rojicek, J.: Automatic setup of fault detection algorithms in building and home automation. In: *21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016)*, Berlin, Germany (2016)
17. Dibowski, H., Holub, O., Rojicek, J.: Ontology-based automatic setup of virtual sensors in building automation systems. In: *International Congress on Ultra Modern Telecommunications & Control Systems (ICUMT 2016)*, Lisbon, Portugal (2016)
18. Dibowski, H., Holub, O., Rojicek, J.: Knowledge-based fault propagation in building automation systems. In: *Second International Conference on System Informatics, Modeling and Simulation (SIMS 2016)*, Riga, Latvia (2016)