Ontology-Based Modeling of Control Logic in Building Automation Systems

Georg Ferdinand Schneider, Pieter Pauwels, and Simone Steiger

***Abstract*—The control logic implemented in building automation systems (BAS) has a significant impact on the overall energy demand of the building. However, information on the control logic, if documented, is often concealed from further data integration and reuse in heterogeneous information silos using disparate data formats. In particular, existing data formats and information models offer limited support to describe control logic explicitly. Ontology-based modeling of the control logic of BAS can potentially result in a versatile source of information for information-driven processes to further increase the performance of technical equipment in a building. Therefore, we present a novel information model, CTRLont, which allows to formally specify the domain of control logic in BAS. We demonstrate the usefulness of the novel information model by using it as a knowledge base for automating rule-based verification of designed control logic in BAS. We successfully apply the methodology to a simple control of an air handling unit and indicate a number of future steps.**

***Index Terms*—Building automation system (BAS), control, ontology, schedule, state graph, state machine.**

I. Introduction

AUTOMATION systems are essential ingredients for the mass production of goods and commodities as well as the automation of technical systems (e.g., buildings) to enable resource efficient operation [1]. In the building sector, building automation systems (BAS) are understood to be a key technology to increase energy efficiency of existing and future buildings [2], [3]. The use of BAS is stipulated by relevant standards (e.g., EN 15232 [4], LEED [5]). Moreover, up to 30% of heating energy savings may be achieved in an office building by deploying properly configured BAS [4]. An overview of control systems for energy and comfort management related to buildings is provided in [3]. Unfortunately, practical implementations of BAS often fail to meet their expectations regarding energy efficiency. Such malfunctioning BAS contribute to the perceived gap between predicted and actual energy demand of a building [6]. In order to establish effective ways of designing, commissioning, and operating energy-efficient BAS, there is a strong need to exchange information regarding its innermost part: the control logic.

Information exchange over the life cycle of BAS, including their control logic, commonly consists of textual descriptions, spreadsheets, and two-dimensional drawings. Such exchanges are suggested for design documentation and information handover after commissioning by relevant standards in the field of BAS [7]–[9]. Consequently, verifying correct operation of the control logic from this unstructured documentation [10], [11] or checking the compliance of BAS with energy efficiency classifications [4] are cumbersome, time-consuming, and costly tasks. Such processes require a structured definition of the as-designed control logic, the bindings defining the logical topology of all control actors in BAS, contextual information on the affiliation of control entities to building elements and equipment, and additional information on inputs and outputs (e.g., measured quantities). In conclusion, the problems in the existing information exchange on control logic in BAS are:

1) Reliance on textual descriptions, spreadsheets, and drawings;
2) Heterogeneous and distributed nature of related information over various formats and information silos specified without definition of common semantics;
3) Absence of a unified manner to provide contextual information from adjacent information domains, e.g., building elements and equipment.

To address the above mentioned problems, several efforts aimed at modelling information related to BAS exist, as reported in Section II. However, none of these efforts fulfill all requirements of an appropriate information model to overcome the summarized problems.

1) High-level description of control actors and related semantics, i.e., inputs, outputs, parameters, the control actor, and its logic and the domain-specific relationships among them, e.g., hierarchy and logical binding;
2) Explicit, formal specification of the control logic itself;
3) Contextual information on inputs and outputs: unit, medium, quantity, affiliation to building elements, and equipment;
4) A modular structure to ensure easy extension with future, novel control logic, and to encourage reuse of existing ontologies in adjacent information domains;

Manuscript received October 31, 2016; revised April 14, 2017, May 24, 2017, and July 3, 2017; accepted August 12, 2017. Date of publication August 23, 2017; date of current version December 1, 2017. This work was supported by the state of Bavaria through the "Bavaria on the move" initiative. Paper no. TII-16-1252. *(Corresponding author: Georg Ferdinand Schneider.)*

G. F. Schneider and S. Steiger are with the Fraunhofer Institute for Building Physics, Nuremberg 90429, Germany (e-mail: georg.schneider@ibp.fraunhofer.de; simone.steiger@ibp.fraunhofer.de).

P. Pauwels is with the Ghent University, Ghent 9000, Belgium (e-mail: pipauwel.pauwels@ugent.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2743221

5) Use of a machine-interpretable format to enable intelligent, automated applications based on this information.

The contribution of this paper resides in presenting a novel information model termed *CTRLont*, which aims to respond to the above five requirements and, thus, allows to integrate explicit models of control logic in a formal and unified manner with adjacent information domains. *CTRLont* allows to specify inputs and outputs of control actors with e.g., units, quantities, etc. Moreover, it is possible to describe control actors and related semantics from the control domain on an abstract level. *CTRLont* supports the integration of explicit descriptions of control logic in a modular way, thus, allowing easy reuse and extensibility. The model is implemented using the Web Ontology Language (OWL) [12], enabling its reuse in intelligent applications through semantic search and reasoning. Additionally, we provide information models to explicitly model control logic defined as Unified Modelling Language (UML) state machines [13], state graphs [14], and schedules.

In the subsequent sections, we, first, analyze related work on information modeling of control logic in BAS (see Section II) with regard to meeting the defined requirements. We then present the novel information model (CTRLont, Section III) and explicit specifications of control logic (see Section IV). Finally, we demonstrate the usefulness of the novel model by automating rule-based verification of designed control logic in a use case (see Section V).

II. Related Work

A. Generic Formats for the Representation of Control Logic

A method to design control logic are UML state machines [13]. UML is a well accepted, standardized modeling language across various domains. A format that can be used to represent and exchange these designs between UML editors is the XML-based data format XML metadata interchange (XMI) [15]. However, because the UML is designed to describe generic applications, certain information uniquely associated to BAS cannot be expressed in a commonly agreed manner. For example, the affiliation to pieces of technical equipment or detailed semantics of input variables is not commonly defined in the UML.

Another format for industrial automation systems is standardized in IEC 62714 [16]. In this standard, the control logic is exchanged using the open PLCopenXML [17] format. The format allows to specify a logical topology as a directed graph and basic data types for interface variables, e.g., integers. The actual control logic is specified by including the implemented source code in one of the five languages from IEC 61131-3 [18]. The PLCopenXML data format covers high-level interface descriptions of control actors, e.g., the logical topology. Limitations exist when a detailed specification of the unit or quantity of an input or output is required. Furthermore, the formal specification of the control logic is hampered when using text-based programming languages from IEC 61131-3 [18], since the format stores the plain source code only.

An effort stemming from the modeling of heterogeneous models of computation resulted in the modeling markup language (MoML) [19], which is implemented in XML and supports an actor-oriented description of systems. The actor-oriented basis of MoML [19] enables the high-level description of control actors and relationships among them, e.g., input is connected to output. However, the modeling language lacks the possibility to express the affiliation of an actor to building elements and equipment.

Finally, an information model for BAS implemented using the EXPRESS data modeling language [20] covers a high-level description of control actors (hierarchy, logical topology) and the integration with adjacent domains of information (network, data points, building elements, and equipment) [21]. However, details on specifying, for example, the unit of an output relies on naming conventions in this approach. This poses serious barriers for interoperability and reuse.

B. Representation of Control Logic Using the Web Ontology Language (OWL)

Several examples in the literature use OWL [12] for the representation of BAS, with the aim of formalizing the description of BAS and to allow linking to related information areas. These two key aims are at the core of the Semantic Web idea. A comprehensive introduction to Semantic Web technologies in general may be found in [22], and an overview on the topic in the building domain is provided in [23].

For instance, DogOnt [24] is an ontology to describe domotic environments, such as connected household appliances. For an extended use case the ThinkHome ontology formalizes all relevant information for energy analysis in residential buildings, including ambient weather conditions, building structure and materials, appliances, comfort, energy, processes, and schedules [25]. The smart appliances reference ontology is a consensus of domain experts for smart household appliances that was established after reviewing existing ontologies in the field [26].

Some approaches focus on information integration for energy management. An ontology is presented by [27] for an airport facility. This ontology constitutes a knowledge base for energy management according to ISO 50001 [28]. A similar approach is presented by [29], but with a stronger focus on energy system modeling. A remarkable piece of work on cross-domain data integration using ontologies for energy performance assessment in buildings is presented in [30] ("performance assessment ontology"). Notable work in the direction of defining a domain ontology for BAS (BASont) is documented in [31]–[33]. It is related to cope with heterogeneity and interoperability issues in BAS, resulting from the wide range of data formats and communication protocols used in these systems. Some of the developed ontologies are used to automate the design process of BAS [33].

A wider range of domains is captured by information models designed to represent all information related to a building to enable seamless information exchange over its life cycle [Building Information Modeling (BIM)]. Such models include information on building elements and equipment, geometry, BAS, and process and project management [34]. The Industry Foundation Classes (IFC) is an information model developed by buildingSMART that captures such information. It is implemented in

EXPRESS [20] and has been translated to XSD and OWL [35]. Closely related to the IFC, but with its application in the domain of energy simulation, SimModel [36] aims at bridging the gap between BIM and building energy performance simulation. The SEAS ontologies are a family of upper level ontologies for the engineering domain and provide concepts for data points in BAS on an abstract level [37].

C. Summary

After reviewing the existing data formats and ontologies, none is found to fulfill all five requirements specified in the introduction. The definition of formal semantics in languages other than OWL is possible but limited due to their reliance on XML or EXPRESS technology (see Section II-A above). Existing conceptualizations from ontology-based modeling approaches related to BAS (see Section II-B above) are tailored to the respective application areas, i.e., smart home and appliances, energy efficiency, BIM, and BAS. All models are implemented using OWL, enabling the reuse of information modeled using these ontologies for intelligent applications. The ontologies include concepts to describe building elements and equipment, physical entities of BAS such as devices, sensors, actuators, and virtual entities, e.g., data points, control actors, etc., with a varying degree and resolution. However, none of the reported approaches allow to explicitly model the control logic existing in BAS. Instead, reported approaches aim at capturing control logic using extensive classification schemes where one concept represents one type of control logic [24]–[27], [36] or refers to definitions provided in standards [33]. To fill the existing gap in explicit modeling of control logic in BAS, we propose a novel information model, which will be introduced in the remainder of this paper.

III. CTRLont Ontology

In this section, we describe the architecture of our modeled ontologies that formalize control logic in BAS. The modeling is undertaken using OWL, which offers benefits as it is an expressive formal language with a firm basis in description logics. Furthermore, the usage of OWL allows the integration of the novel model with existing domain models (see Section II-B) and allows the deduction of new insights by semantic search and reasoning. For the design of the CTRLont ontology, we followed a structured approach [38] and we relied on the review of existing conceptualizations documented in Section II. All ontologies are represented using graphs as shown in Fig. 1. The utilized nomenclature and namespaces applied in this work are depicted in Fig. 2. For better readability, we omitted imposed restrictions in the ontology visualizations (cardinality restrictions, universal qualifiers, existential qualifiers, type restrictions).

Central to the CTRLont ontology (see Fig. 1) is the specification of the *sense-process-actuate* pattern, which is recognized in the control domain, e.g., *function profiles* [7], [8], *function blocks* [39], or generic actor-oriented modeling [40]. A `ControlActor` typically acquires information through `Inputs`, processes this information by some `ApplicationLogic` and performs informed actuation via some `Output` [1]. Each `ControlActor` entity is related to an `ApplicationLogic` entity using the relationship `hasApplicationLogic`. The object properties `logicInput`, `logicOutput`, and `logicParameter` are defined to explicitly model the relationships of `ApplicationLogic` to `Input`, `Output`, and `Parameter`. The concept `Parameter` describes time-invariant values and settings of the respective `ControlActor` and can be added according to the specific needs of the respective `ApplicationLogic`.

In any building automation solution, a network of `ControlActor` entities `isConnectedTo` each other via the respective inputs and outputs forming the logical topology of the system. Moreover, a hierarchical relationship exists between entities placed at the field level/ automation level and entities placed at the management level [7]. A management level entity often `supervises` an entity on the automation and field level and a field level entity `isSupervisedBy` a management level entity. This implicit hierarchy needs to be reflected by the model, e.g., to deduct that a control strategy implemented on management level is the root cause for a local control failure.

The concept of a `ControlActor` may be used to specify both open-loop and closed-loop control entities [41]. In the case of closed-loop control entities, specific concepts and relationships need to be introduced in the respective `ApplicationLogic` to describe a setpoint either as an `Input` or a `Parameter`. Further annotation of inputs, outputs, and parameters is important to enable interoperability and exchange of `ControlActor` entities among software and automation systems or the automated design of BAS [33]. In particular, the specification of their `Unit`, `Quantity`, `Medium`, and `SemanticType` (see Fig. 1) is needed. We refrain here from specifying completely new ontologies. Instead, we implement placeholders for the mentioned concepts and suggest reusing existing ontologies, e.g., ontology for units of measure (OM) [42]. Further annotation might be necessary to allow the description of more detailed differences such as a drybulb and a wetbulb air temperature [33]. Therefore, a domain-dependent `SemanticType` can be introduced.

The concepts `Input`, `Output`, and `ControlActor` are related to technical equipment (e.g., air handling unit (AHU) controller) or to building elements such as spaces (room temperature control). Instead of expressing this through the use of taxonomies, we suggest to relate the respective concepts directly to concepts from existing information models from adjacent domains, e.g., BAS [32], BIM [35] and technical systems [37].

IV. Explicit Modeling of Control Logic

In this section, we provide examples of ontologies that can be used to explicitly represent control logic in BAS, namely UML state machines [13], state graphs according to VDI3814-6 [14], Boolean expressions apparent in conditions of state machines and state graphs and schedules. The chosen examples reflect a portion of control logic frequently deployed in BAS but are not meant to be exhaustive. The examples represent the lower part

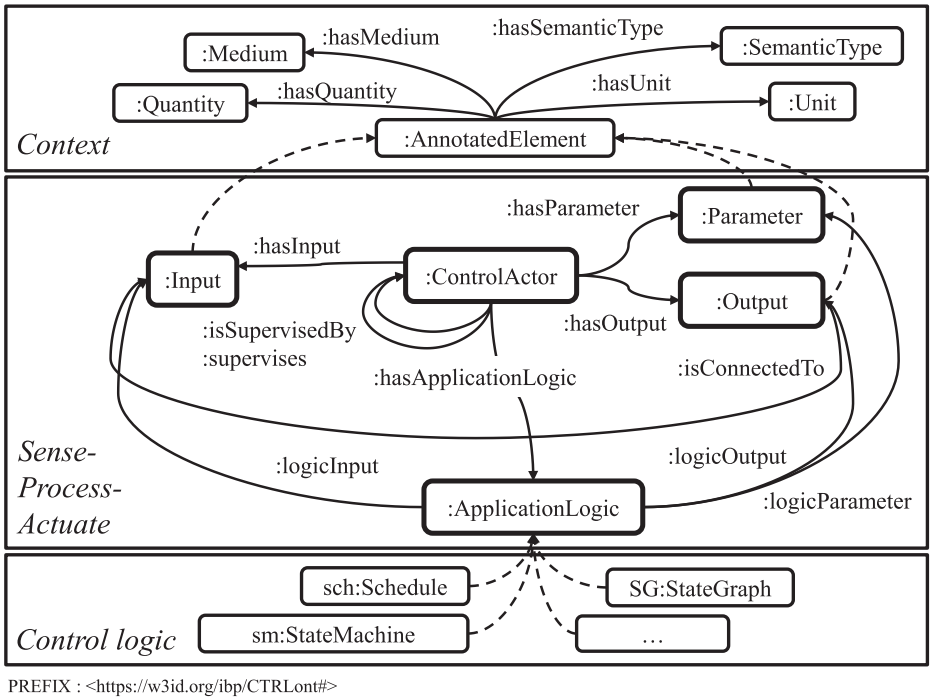

Fig. 1. Concepts and relationships of the CTRLont ontology.

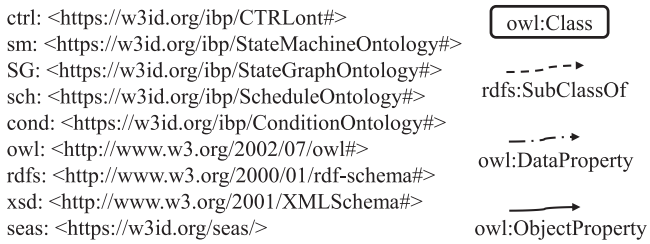

Fig. 2. Overview of namespaces and nomenclature used.

of Fig. 1 (*control logic*), and thus extend the CTRLont ontology specified above.

A. UML State Machine

UML state machines provide a generic way to model control logic [13]. We developed the ontology presented in Fig. 3 by modifying an existing one [43]. Our modifications include the alignment to the latest version of the UML standard [13] and an extension to explicitly include Boolean expressions in guards. As is clear from Fig. 3, `StateMachine` contains `StateMachineElements` such as `State` and `Transition`. `State` may be further specified, e.g., by `Composite`. The transition from one state to another is triggered by an `Event` and a `BooleanExpression` described using the ontology presented in Section IV-C guards the transition between two states. `State` and `Transition` usually have a related `Action`. According to the UML standard, any arbitrary behavior may be specified for an `Action`. Within an `Action`, inputs are transformed to outputs. In this model, we, therefore, define a placeholder concept `Behaviour` that has relationships to `Input`, `Output`, and `Parameter`. Suitable approaches

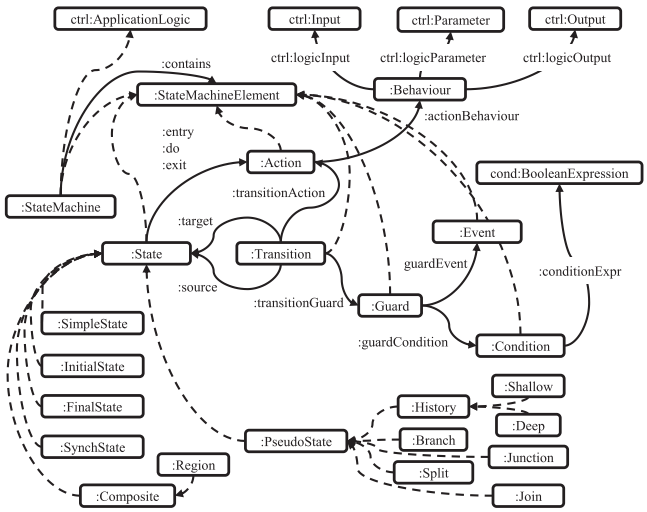

Fig. 3. Concepts and relationships to model a UML state machine. The modeling of `BooleanExpression` in conditions is detailed in Section IV-C.

to specify `Behaviour` are briefly discussed at the end of Section V. Finally, the `StateMachine` concept is specialized from `StateMachineElement`, aiming to comply with the UML definition of nested state machines, i.e., a state machine may contain other state machines.

B. State Graphs

State graphs are defined according to the VDI 3814-6 standard [14] and are supposed to fill a gap in standardized control descriptions by representing "specific logic interlocks" in BAS.

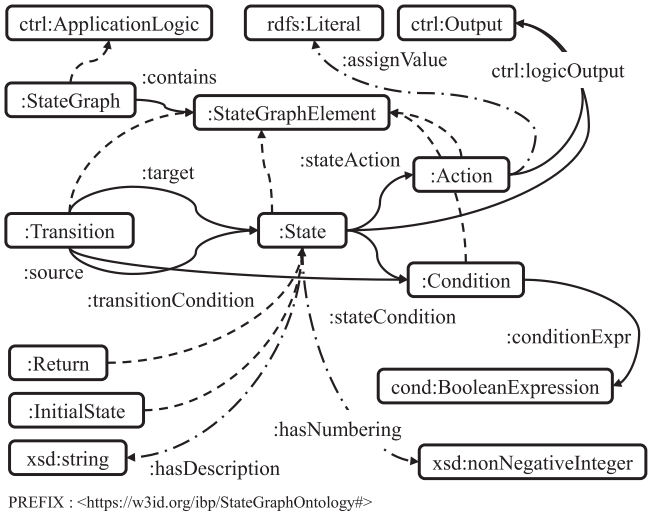

Fig. 4. Concepts and relationships to describe a state graph as specified in the standard VDI 3814-6 [14]. The modeling of `BooleanExpression` in conditions is detailed in Section IV-C.

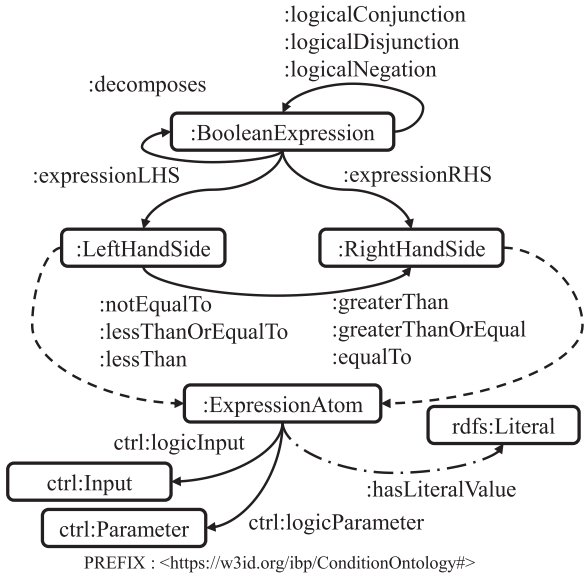

Fig. 5. Concepts and relationships to describe conditional logic expressed as Boolean expressions.

They are used to provide a graphical representation of control logic that can complement textual descriptions. We designed an ontology for state graphs as illustrated in Fig. 4, while taking into account the previously defined ontology (see Fig. 3). A difference between state graphs and state machines is that conditions triggering a transition from one state to another only relate to a state. The designed ontology reflects this detail by relating the concepts `Action` and `Condition` directly to `State`. As defined in the standard, a value is assigned to an output (`assignValue`) immediately when a state and its associated actions are active. An `Output` may be related to `State` to represent a virtual datapoint indicating state activity. The standard defines, separately from state graphs, function blocks that may be used to translate the assigned value, possibly involving the translation of inputs to outputs. As this function block forms a separate `ApplicationLogic`, it is not specified here.

If more than one transition is targeting a state, conditions need to link to the respective transitions using the `transitionCondition` object property. Instances of `Input` and `Condition` are related as described in Section IV-C.

C. Capturing Conditional Logic

The description of conditional logic is a central element to link concepts from state-based control logic such as UML state machines and state graphs to inputs of a control actor. For example, the transition from one state to another state in a UML state machine may be prohibited by a guard, which specifies a condition $y <= 1$. In data formats like XMI or MoML, this information is stored as a plain string, omitting the actual semantics of this piece of information (see Section II). The ontology presented in Fig. 5 on the contrary allows to formalize the conditions apparent in transitions of state-based control logic. We assume that these pieces of conditional logic are expressed as Boolean expressions, which may be concatenated and nested using logical conjunctions (AND), logical disjunctions (OR), and logical negations (NOT). The body of a Boolean expression usually incorporates a left-hand and a right-hand side of the expression. Left-hand side and right-hand side are in a relationship, which captures the information to compare both, e.g., `equalTo`. The affiliation to an `Input` or `Parameter` can be expressed using the `logicInput` and `logicParameter` property, respectively.

D. Schedule

Schedules are a frequently occurring control logic within BAS. They provide a straight-forward way to reduce the energy demand of a building by simply aligning equipment operation with occupation periods. An ontology to describe schedules in a generic manner is presented in Fig. 6.

A schedule is modeled as a set of consecutive intervals, which are defined with their start and end points and a mathematical function describing the relationship in-between. We assume that only polynomial functions occur. The explicit relationship to `Input` and `Output` of the respective `Schedule` is established. Moreover, the concept `Periodicity` allows to specify whether a schedule is repeated on a daily, weekly, or yearly basis.

V. Use Case

In the following section, we underline the usefulness of ontology-based, explicit modeling of control logic by automating a rule-based verification method of designed control logic in BAS presented by [10], [11].

A. Automated Rule-Based Verification of Designed Control Logic in BAS

It is nearly impossible for control logic designers to verify whether the as-designed control logic was implemented as specified after commissioning. This is caused mainly by the

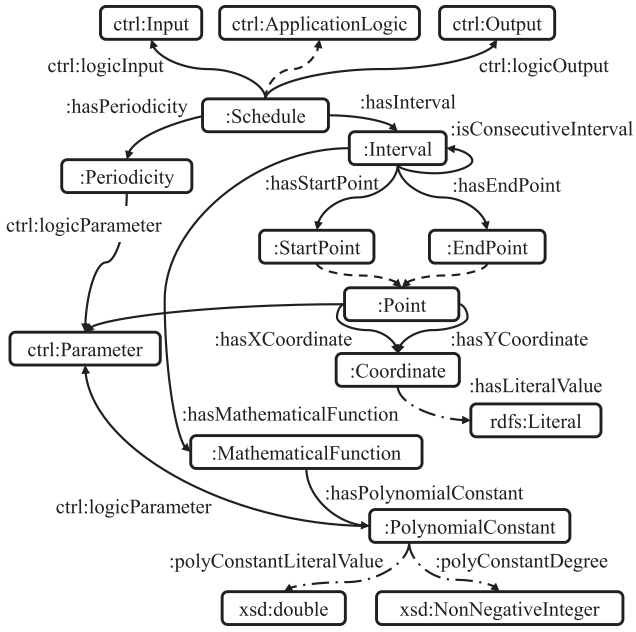

Fig. 6. Concepts and relationships to describe a schedule.

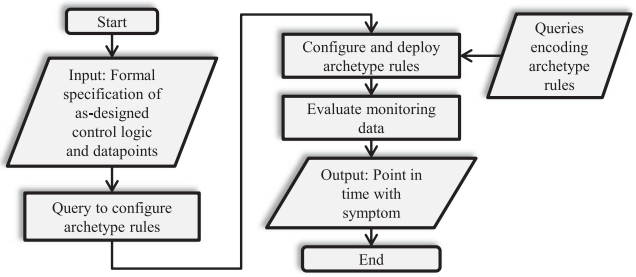

Fig. 7. Flow chart for automated rule-based verification of designed control logic in BAS.

following reasons: the monitoring of data through seasons and operation modes is not yet available and, usually, the source code of the implementation is concealed. A methodology that addresses this problem is introduced in [10]. The approach relies on the analysis of existing documentation of a building automation solution by domain experts, who then define verification rules to evaluate monitoring data. Clearly, manually analyzing this documentation is highly time-consuming, because documentation is spread across various data formats including textual descriptions, spreadsheets, and drawings. Instead of undertaking this labour-intensive task, we propose to follow an automated process as depicted in Fig. 7.

First, data describing the control logic design and respective data points are extracted and formalized from state-of-the-art design tools for BAS. The results are stored in a knowledge base. Archetype rules derived from expert knowledge on how to verify a certain control logic type are encoded as parameterized queries. Next, the necessary information to configure instances of the parameterized queries encoding the archetype rules is retrieved using a fixed query to the knowledge base. Monitoring

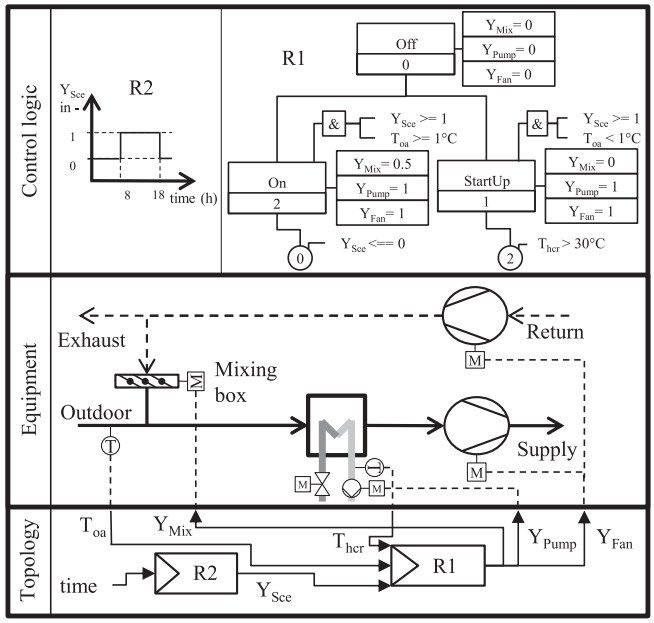

Fig. 8. Schema [7] of implemented use case. Y_{Sce}, Y_{Mix}, Y_{Fan}, Y_{Pump}—Normalized output signal from schedule, mixing box damper flap, fan and pump, T_{hcr}—Heating coil return water temperature, T_{oa}—Outdoor air temperature.

data are evaluated by deploying the configured queries. In this approach, a deviation at a point in time from the designed control logic is interpreted as a symptom of a fault. The results are stored and visualized for fault isolation and post-processing, e.g., by calculating the weighted operational quality [10].

B. Implementation and Results

The method described above has been implemented using Python programming language and is deployed in a test case for a fictional AHU. First, we describe the setting of the test case; then we present instances populating the above-mentioned ontologies and queries to implement the method following the procedure as outlined in Fig. 7. Finally, we present results from applying the method to simulated data.

The designed, rather simple control consists of two control actors, a schedule and a state graph. The topology of the control logic, its connections to inputs and outputs and drawings of the control logic are illustrated in Fig. 8. The schedule restricts operation time of AHU fans and pumps to occupied times between 8 am and 6 pm. A state graph is modeled to actually operate the AHU within states `Off`, `StartUp`, and `On`. `Return` states illustrated as circles act as placeholders and redirect to the respective states when the associated Boolean expressions evaluate to true. To prevent damage to the heating coil through freezing (outdoor air temperatures T_{oa} below 1° C), the control logic operates the AHU by recirculating return air $Y_{\mathrm{Mix}} := 0$ at `StartUp` only. Normal operation (`On`) with a fixed opening of the mixing box damper Y_{Mix} starts when the heating coil water return temperature T_{hcr} reaches 30 °C.

Fig. 9 shows an excerpt of instances of the designed control logic presented in Fig. 8 that are stored in a knowledge base. Note, here we use SEAS ontologies to model data points of BAS

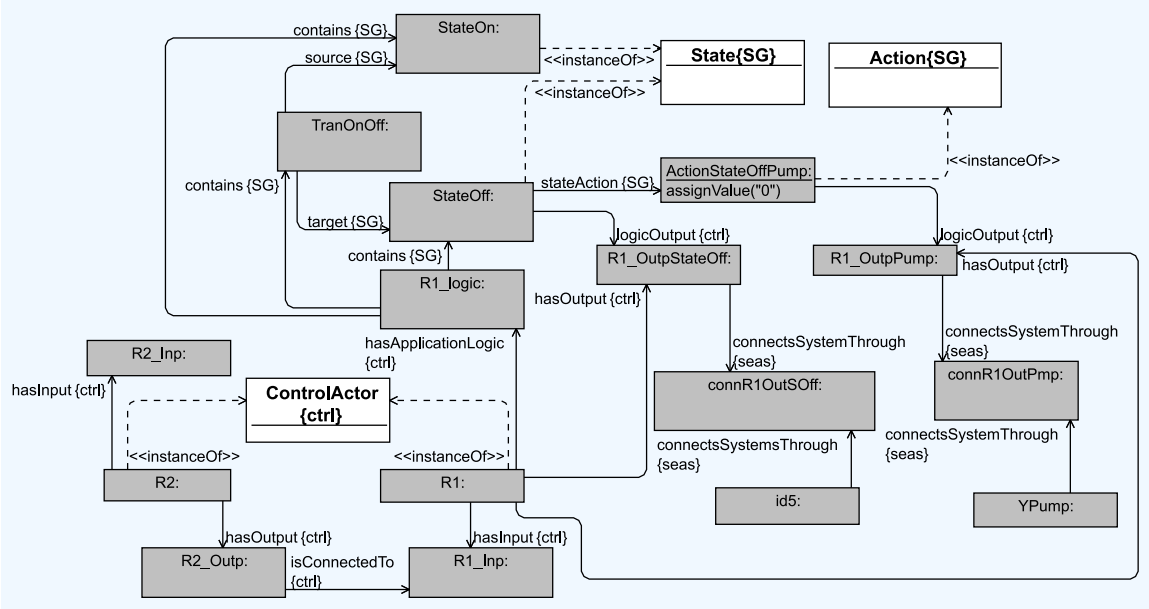

Fig. 9. Excerpt of instances used in test case (see Fig. 8), visualized using OWLGrEd tool and notation [45]. `id5`—Virtual data point of state `Off`.

TABLE I
EXAMPLE INFORMATION RETRIEVED FROM KNOWLEDGE BASE FOR STATE `Off`. VARIABLES AS DENOTED IN CODE 1.

vState	vDPState	vDPMeas	vValue
:StateOff	:id5	:YMix	'0'
:StateOff	:id5	:YFan	'0'
:StateOff	:id5	:YPump	'0'

Code 1: Query to retrieve necessary information to configure parameterized query listed in Code 2.

```
PREFIX [...]
SELECT ?vState ?vDPState ?vDPMeas
?vValue
WHERE
{?vSG rdf:type SG:StateGraph ;
      SG:contains vState .
?vState rdf:type SG:State .
?vState SG:stateAction ?vAction .
?vAction rdf:type SG:Action .
?vAction SG:assignValue ?vValue .
?vAction ctrl:logicOutput ?vOutput .
?vOutput rdf:type ctrl:Output .
?vOutput seas:connectsSystemThrough
?vConn .
?vDPMeas seas:connectsSystemThrough
?vConn .
?vDPMeas rdf:type seas:Communication
  ConnectionPoint .
?vState ctrl:logicOutput ?vOutputS .
?vOutputS rdf:type ctrl:Output .
?vOutputS seas:connectsSystemThrough
?vConnS .
?vDPState seas:connectsSystemThrough
?vConnS .
?vDPState rdf:type seas:Communication
  ConnectionPoint .}
```

on an abstract level [37]. This information is retrieved to configure parameterized queries implementing archetype rules. For example, the SPARQL query in Code 1 can be used to configure instances of the parameterized query listed in Code 2 (also implemented in SPARQL [44]). Results of the query listed in Code 1 for state `Off` are presented in Table I. The following archetype rule can be defined from expert knowledge: If a state is active, the associated actions are supposed to be executed [10]. Consequently, monitoring data can be queried for occurrences where the actual value differs from the respective value obtained from the design specification. For the presented use case a parameterized query realizing this is given by Code 2. Similarly, archetype rule and query can be formulated for a schedule.

Here, we chose to store the monitoring data obtained from simulations in RDF format [46] using a straightforward transformation method that models each sampled measurement as an individual of `PrimaryKey`. Readings are associated to each of these individuals using an `owl:ObjectProperty` (e.g., `tb:YPump`), which has the unique identifier of the reading as its URI. We use an arbitrary prefix in this case (`tb:`). Strings enclosed by $ are placeholders and replaced in configured instances of the queries.

In order to deploy all configured queries and to follow the notion of *operation state space* (OSS) as introduced by [10], all instances of queries are nested into one query using `UNION` to concatenate (Code 3). The subquery nesting ensures efficient execution of the query by the SPARQL endpoint. This query is then executed against monitoring data to filter points in time with symptomatic behavior.

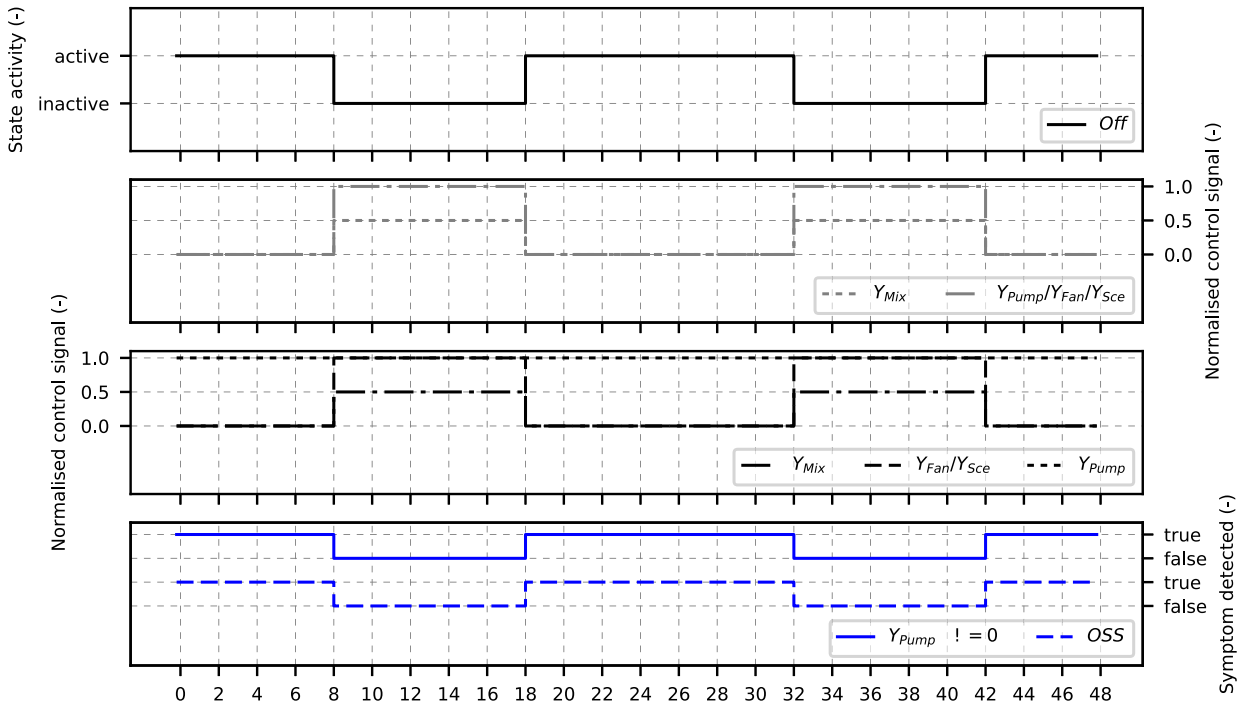

Fig. 10. Results from evaluating simulated data for 48 hrs. Enumeration from top to bottom: 1) Signal indicating if state `Off` is active; 2) Correct, as-designed behavior for comparison; 3) Evaluated overall behavior with faulty control signal for Y_{Pump}, 4) Interpretation of the results obtained from running an instance of the query reported in Code 2 (Y_{Pump} $!= 0$) and a combined one as reported in Code 3 (OSS); If the queries return one or more individuals, the respective points in time are interpreted to be symptomatic (true), else not (false). Y_{Sce}, Y_{Mix}, Y_{Fan}, Y_{Pump}—Normalized output signal from schedule, mixing box damper flap, fan and pump, *OSS*—result of evaluated operation state space.

Code 2: Parameterized SPARQL query to filter points in time with faulty state graph behavior.

```
PREFIX [...]
SELECT ?pk ?timeValue
WHERE
{?pk a tb:PrimaryKey .
 ?pk tb:time ?timeValue .
 ?pk tb:$vDPMeas$ ?yOutpValue .
 ?pk tb:$vDPState$ ?yStateValue .
 FILTER( ?yStateValue = "1.0" &&
       ?yOutpValue != "$vValue$" ) .}
ORDER BY ?timeValue
```

Code 3: Composed query to evaluate monitoring data.

```
PREFIX [...]
SELECT ?pk ?timeValue
WHERE
{
        {# Configured query 1}
UNION
        {# Configured query 2}
UNION
        {#...}
}
```

We evaluate our implementation using data obtained from simulating a model of the controlled AHU (see Fig. 8), which we implemented using simulation models from the `Buildings` library [47]. We introduce a common fault in BAS where problems with some piece of technical equipment occurred and automated control is manually turned off for repairs and troubleshooting but is not turned on again. To emulate this behavior we set the normalized pump control signal (Y_{Pump}) to 1.0 such that the pump is operating continuously. Simulation results and results from deploying the automated verification system are depicted in Fig. 10.

In the following discussion, we enumerate subfigures in Fig. 10 from top to bottom. Assuming state `Off` is active (1) the normalized control signal of the pump Y_{Pump} is supposed to be zero (2). However, in the faulty data the signal stays at 1.0 (3). Points in time are filtered by standalone instances of the query given in Code 2 or all instances aggregated in Code 3, interpreted, and reported by the system (4).

C. Discussion

Several advantages can be identified from applying the ontology-based modeling approach in the described domain. The generic modeling approach allows to define and express relationships between disparate information domains including building elements and equipment, explicit models of control logic, static information on BAS, and monitoring data. Furthermore, this method allows to model control logic in a more detailed and structured manner. For instance, it is now possible to explicitly model the Boolean conditions apparent in state machines and state graphs and define their

semantics. A relationship can now be established between a physical sensor placed in a room of a building, its associated virtual data point and the control logic evaluating its value. Information in this domain can then be used in intelligent applications, as presented in Section V-A for automated rule-based verification.

The presented use case enables detection of faults caused from control logic failure. However, to find suitable remedies, its root cause needs to be identified (fault isolation) [48]. Promising work supporting this has been presented in [49]. The ontologies presented in this paper may extend the explanation space of the described methodology, e.g., via backward fault propagation to identify the root cause of a fault. Furthermore, the effect of a control logic failure on related building elements and equipment could be estimated through forward fault propagation, where the control logic misbehavior is identified using the proposed methodology presented in this paper. Of course, expert knowledge will be needed for defining the causality and fault propagation relationships.

When modeling UML state machines, the actual relationship between inputs and outputs is established by the respective behavior executed when an action is active. As indicated, this can be close to anything. This degree of freedom poses limitations to the modeling approach. Further research is required to find suitable modes of description for behavior, which may range from mathematical expressions potentially encoded via OpenMath [50] to UML activity diagrams [51].

VI. Conclusion

In this paper, we present a novel information model, *CTRLont*, which forms the basis to formalize control logic information in BAS. The concepts and relationships defined allow to integrate an explicit specification of the control logic and adjacent information areas, i.e., BAS, BIM models, and BMS. We present models to formally specify state-based control logic, namely UML state machines, state graphs and schedules.

We demonstrate the usefulness of the novel model by using it as a knowledge base for automating rule-based verification of designed control logic, where rules are configured dynamically and applied to monitoring data obtained from an instance of virtual BMS. We successfully apply the methodology to an AHU control test case. Using this methodology, the amount of time spent on reading textual descriptions of control logic in BAS can be reduced and engineers and facility managers can spend more time on isolating the root cause of faults. We assume here that the designed ontologies can be automatically populated from authoring tools used by control logic designers (exporters/adapters).

The presented ontology can serve as a catalyst in establishing a shared common understanding of control logic in the automation domain. It enables novel intelligent methods to improve building operation in the future, e.g., tailored rule-based verification of control logic in BAS or extend the explanation space of knowledge-based methods for detecting the root cause of faults in buildings [49]. For example, by forward fault propagation estimating the effect of a control logic failure or in backward fault propagation to identify control logic failure as (one of) the root cause(s).

A number of challenges still exists regarding the proposed ontology-based modeling of control logic:

1) Complexity of deployed control logic:
   In general, control logic in BAS may range from a simple PID-controller to advanced predictive and adaptive controllers applied to various processes in the building [3]. This complexity poses barriers to what extent a consensus can be agreed on in terms of ontology for such advanced controllers.
2) Intellectual property:
   In the automated approach reported above, the execution of the rules requires the information of the current active state in a state-based controller as an input. In practical implementations of these controllers, this information is often concealed due to intellectual property (IP) reasons imposed by the company supplying the device. This coincides with complexity as IP concerns typically increase the more advanced a control logic is. Suitable workarounds to automatically identify the operation states need to be developed, e.g., evaluating the pressure difference measured in the supply duct of an AHU to determine its operation as suggested by [10].

Most of these challenges are also present in non-ontology-based approaches. A number of future research paths can be outlined as well:

1) Ontology linking:
   The existing implementation of the presented ontologies is performed from scratch to ensure usability and semantic correctness. However, to benefit most from using Semantic Web technologies the integration with existing ontologies in the domain needs to be stipulated, e.g., BASont [32], ifcOWL [35], SEAS [37], OM [42], OpenMath [50], and SSN [52].
2) Model-based information exchange in BAS:
   Current tools for designing a building automation solution store the design information in various formats. An effort is needed to enable a standardized model-based information exchange over the life cycle of BAS.
3) Formalizing existing control logic documentation:
   New ways to formalize data on control logic from existing documentation need to be found. One path relates to the extension and refinement of existing data formats, e.g., PLCOpenXML [17]. Modeling work undertaken by [53] may be a good starting point. A second path may address current documentation based on spreadsheets and drawings by extracting this information through the use of semiautomatic processes [54].

References

[1] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, "The evolution of factory and building automation," *IEEE Ind. Electron. Mag.*, vol. 5, no. 3, pp. 35–48, Sep. 2011.

[2] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication systems for building automation and control," *Proc. IEEE*, vol. 93, no. 6, pp. 1178–1203, Jun. 2005.

[3] A. Dounis and C. Caraiscos, "Advanced control systems engineering for energy and comfort management in a building environment A review," *Renewable Sustain. Energy Rev.*, vol. 13, no. 67, pp. 1246–1261, 2009.

[4] *Energy Performance of Buildings Impact of Building Automation, Controls and Building Management*, EN 15232 Standard, 2013.

[5] "LEED v4 for building operations and maintenance," US Green Building Council, Tech. Rep. 4, 2016.

[6] P. De Wilde, "The gap between predicted and measured energy performance of buildings: A framework for investigation," *Autom. Construction*, vol. 41, pp. 40–49, 2014.

[7] *Building Automation and Control Systems (BACS)*, ISO 16484 Standard, 2011.

[8] *Building Automation and Control Systems (BACS) Fundamentals for Room Control*, VDI 3813-1 Standard, 2011.

[9] *Building Automation and Control Systems (BACS) System Basics*, VDI 3814-1 Standard, 2013.

[10] S. Plesser, "Aktive Funktionsbeschreibungen zur Planung und Überwachung des Betriebs von Gebäuden und Anlagen," Ph.D. dissertation, TU Braunschweig, Braunschweig, Germany, 2013.

[11] S. Plesser, M. N. Fisch, C. Pinkernell, T. Kurpick, and B. Rumpe, "The energy Navigator-A web based platform for functional quality mangement in buildings," in *Proc. Int. Conf. Enhanced Bldg. Oper., Kuwait City, Kuwait*, 2010.

[12] W3C, "Web ontology language—OWL," 2016. [Online]. Available: http://www.w3.org/2001/sw/wiki/OWL

[13] *OMG Unified Modeling Language (OMG UML)*, Object Management Group, 2015.

[14] *Building Automation and Control Systems (BACS) Graphical Description of Logic Control Tasks*, VDI 3814-6 Standard, 2013.

[15] *XML Metadata Interchange (XMI) Specification*, Object Management Group, 2015.

[16] *Engineering Data Exchange Format for Use in Industrial Automation Systems Engineering—Automation Markup Language—Part 1: Architecture and General Requirements*, IEC 62714-1 Standard, 2014.

[17] *XML Formats for IEC 61131-3*, PLCopen Standard, 2009.

[18] *Programmable Controllers—Part 3: Programming Languages*, IEC 61131-3 Standard, 2014.

[19] E. A. Lee and S. Neuendorffer, "MoML—A Modeling Markup Language in XML—Version 0.4," Univ. of California, Berkeley, CA, USA, Tech. Rep. ERL/UCB M 00/12, 2000.

[20] *Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 11*, ISO 10303-11 Standard, 2004.

[21] J. Schein, "An information model for building automation systems," *Autom. Construction*, vol. 16, no. 2, pp. 125–139, 2007.

[22] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web technologies*. Boca Raton, FL, USA: CRC Press, 2010.

[23] P. Pauwels, S. Zhang, and Y.-C. Lee, "Semantic web technologies in AEC industry: A literature overview," *Autom. Construction*, vol. 73, pp. 145–165, 2017.

[24] D. Bonino and F. Corno, "DogOnt—Ontology modeling for intelligent domotic environments," in *Proc. Int. Semantic Web Conf. Lec. Not. iComp. Sci.*, vol. 5318, Karlsruhe, Germany, 2008, pp. 790–803.

[25] C. Reinisch, M. J. Kofler, F. Iglesias, and W. Kastner, "ThinkHome energy efficiency in future smart homes," *EURASIP J. Embedded Syst.*, no. 104617, pp. 1–18, 2011.

[26] L. Daniele, F. den Hartog, and J. Roes, "Created in close interaction with the industry: The smart appliances REFerence (SAREF) ontology," in *Proc. Ind. Workshop Formal Ontologies Meet Ind.*, 2015, vol. 225, pp. 100–112.

[27] N. M. Tomašević, M. v. Batić, L. M. Blanes, M. M. Keane, and S. Vraneš, "Ontology-based facility data model for energy management," *Adv. Eng. Inf.*, vol. 29, no. 4, pp. 971–984, 2015.

[28] *Energy Management Systems Requirements With Guidance for Use*, ISO 50001 Standard, 2011.

[29] J. Kaiser and P. Stenzel, "eeEmbedded D4.2: Energy System Information Model - ESIM," *eeEmbedded Consortium, Brussels, Belgium*, 2015.

[30] E. Corry, P. Pauwels, S. Hu, M. Keane, and J. O'Donnell, "A performance assessment ontology for the environmental and energy management of buildings," *Autom. Construction*, vol. 57, pp. 249–259, 2015.

[31] C. Reinisch, W. Granzer, F. Praus, and W. Kastner, "Integration of heterogeneous building automation systems using ontologies," in *Proc. 34th Ann. Conf. IEEE Ind. Electron.*, Orlando, FL, USA, 2008, pp. 2736–2741.

[32] J. Ploennigs, B. Hensel, H. Dibowski, and K. Kabitzsch, "BASont—A modular, adaptive building automation system ontology," in *Proc. 38th Ann. Conf. IEEE Ind. Electron. Soc.*, Montreal, Canada, 2012, pp. 4827–4833.

[33] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated design of building automation systems," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3606–3613, Nov. 2010.

[34] G. F. Schneider, A. Bougain, P. S. Noisten, and M. Mitterhofer, "Information requirement definition for BIM: A life cycle perspective," in *Proc. Eur. Conf. Product Process Model., Limassol, Cyprus*, 2016.

[35] P. Pauwels and W. Terkaj, " EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology," *Autom. Constr.*, vol. 63, pp. 100–133, 2016.

[36] J. O'Donnell, R. See, C. Rose, T. Maile, V. Bazjanac, and P. Haves, "SimModel: A domain data model for whole building energy simulation," in *Proc. IBPSA Building Simul.*, Sydney, Australia, 2011, pp. 382–389.

[37] M. Lefrançois, J. Kalaoja, T. Ghariani, and A. Zimmermann, "D2.2: The SEAS knowledge model," ITEA2 SEAS, Brussels, Belgium, 2017.

[38] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," Stanford Univesity, Stanford, CA, USA, 2001.

[39] *Function Blocks—Part 1: Architecture*, IEC 61499-1 Standard, 2012.

[40] J. Eker *et al.*, "Taming heterogeneity—the Ptolemy approach," *Proc. IEEE*, vol. 91, no. 1, pp. 127–144, Jan. 2003.

[41] *International Electrotechnical Vocabulary—Part 351: Control Technology*, IEC 60050-351 Standard, 2013.

[42] H. Rijgersberg, M. van Assem, D. Willems, M. Wigham, J. Broekstra, and J. Top, "Ontology of units of measure (OM)," 2016. [Online]. Available: http://www.wurvoc.org/vocabularies/om-1.8/

[43] P. Dolog, "Model-driven navigation design for semantic web applications with the UML-Guide," in *Proc. Int. Conf. Web Eng.*, 2004, pp. 75–86.

[44] W3C, "SPARQL query language for RDF," 2016. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/

[45] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, and A. Sprogis, "OWLGrEd: A UML style graphical editor for OWL," in *Proc. Workshop Ont. Rep. Eds. Semantic Web*, Hersonissos, Greece, 2010, pp. 1–5.

[46] W3C, "Resource Description Framework—RDF," 2016. [Online]. Available: http://www.w3.org/RDF/

[47] M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica Buildings Library," *J. Building Perform. Simul.*, vol. 7, no. 4, pp. 253–270, 2014.

[48] S. Katipamula and M. R. Brambley, "Review article: Methods for fault detection, diagnostics, and prognostics for building systems—A review, part I," *HVAC&R Res.*, vol. 11, no. 1, pp. 3–25, 2005.

[49] H. Dibowski, O. Holub, and J. Rojíček, "Knowledge-based fault propagation in building automation systems," in *Proc. Int. Conf. Syst. Informat., Model. Simul.*, Riga, Latvia, 2016, pp. 124–132.

[50] K. Wenzel and H. Reinhardt, "Mathematical computations for linked data applications with OpenMath," in *Proc. Conf. Intell. Comput. Math.*, Bremen, Germany, 2012, pp. 38–48.

[51] I. Grobelna, M. Grobelny, and M. Adamski, "Model checking of UML activity diagrams in logic controllers design," in *Proc. DepCoS-RELCOMEX, Brunów, Poland*, 2014, pp. 233–242.

[52] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics*, vol. 17, pp. 25–32, 2012.

[53] T. Horn and J. Ebert, "Ein Referenzschema für die Sprachen der IEC 61131," Univ. Koblenz-Landau, Koblenz, Germany, Tech. Rep. no. 13, 2008.

[54] P. Häfner, V. Häfner, H. Wicaksono, and J. Ovtcharova, "Semi-automated ontology population from building construction drawings," in *Proc. Int. Conf. Knowl. Eng. Ontology Develop.*, Vilamoura, Portugal, 2013, pp. 1–8.

Georg Ferdinand Schneider received the B.Sc. degree in mechanical engineering, M.Sc. degree in energy engineering, and M.Sc. degree in economics all from RWTH Aachen University, Aachen, Germany, in 2011, 2013, and 2014, respectively. He is currently working toward the Ph.D. degree in mechanical engineering at Karlsruhe Institute of Technology, Karlsruhe, Germany.

He is currently a Research Associate at the Fraunhofer IBP, Nuremberg, Germany. His research interests include formal modeling of automation systems and control logic as well as modeling and simulation of technical systems.

Mr. Schneider is a member of the W3C Linked Building Data Community Group.

Pieter Pauwels received the Master's degree and the Ph.D. degree in engineering—architecture, both from Ghent University, Ghent, Belgium, in 2008 and 2012.

He is an Assistant Professor and Postdoctoral Researcher in the Department of Architecture and Urban Planning, Ghent University. His work and interests are in information system support for the building life-cycle. He currently works on topics affiliated to the intersection of Building Information Modelling and Semantic Web technologies.

Dr. Pauwels is an active member of buildingSMART, more precisely as the Co-Chair of the Linked Data Working Group at buildingSMART Int'l. Furthermore, he Co-Chairs the Linked Building Data Community Group at the World Wide Web Consortium.

Simone Steiger received the Diploma degree in architecture from Technical University Munich, Munich, Germany, in 2004, and the M.Eng. degree in building services engineering from Munich University of Applied Sciences, Munich, Germany, in 2007.

She is currently a full-time Researcher and Project Coordinator at the Fraunhofer IBP, Nuremberg, Germany. Her work and research interests include control algorithms and operational strategies for mechanical and natural ventilation in buildings.